Lecture Notes in Computer Science 2660

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Peter M.A. Sloot David Abramson
Alexander V. Bogdanov Jack J. Dongarra
Albert Y. Zomaya Yuriy E. Gorbachev (Eds.)

Computational Science – ICCS 2003

International Conference
Melbourne, Australia and St. Petersburg, Russia
June 2-4, 2003
Proceedings, Part IV

Springer

Volume Editors

Peter M.A. Sloot
University of Amsterdam, Informatics Institute, Section of Computational Science
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: sloot@science.uva.nl

David Abramson
Monash University, School of Computer Science and Software Engineering
Wellington Road, Clayton, VIC 3800, Australia
E-mail: davida@csse.monash.edu.au

Alexander V. Bogdanov
Yuriy E. Gorbachev
Institute for High-Performance Computing and Information Systems
Fontanka emb. 6, St. Petersburg 191187, Russia
E-mail: {bogdanov,gorbachev}@hm.csa.ru

Jack J. Dongarra
University of Tennessee and Oak Ridge National Laboratory, Computer Science Dept.
1122 Volunteer Blvd., Knoxville, TN 37996-3450, USA
E-mail: dongarra@cs.utk.edu

Albert Y. Zomaya
The University of Sydney, School of Information Technologies, CISCO Systems
Madsen Building F09, Sydney, NSW 2006, Australia
E-mail: zomaya@it.usyd.edu.au

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): D, F, G, H, I, J, C.2-3

ISSN 0302-9743
ISBN 3-540-40197-0 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2003
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin GmbH
Printed on acid-free paper SPIN: 10931189 06/3142 5 4 3 2 1 0

Preface

Some of the most challenging problems in science and engineering are being addressed by the integration of computation and science, a research field known as computational science.

Computational science plays a vital role in fundamental advances in biology, physics, chemistry, astronomy, and a host of other disciplines. This is through the coordination of computation, data management, access to instrumentation, knowledge synthesis, and the use of new devices. It has an impact on researchers and practitioners in the sciences and beyond. The sheer size of many challenges in computational science dictates the use of supercomputing, parallel and distributed processing, grid-based processing, advanced visualization and sophisticated algorithms.

At the dawn of the 21st century the series of International Conferences on Computational Science (ICCS) was initiated with a first meeting in May 2001 in San Francisco. The success of that meeting motivated the organization of the second meeting held in Amsterdam April 21–24, 2002, where over 500 participants pushed the research field further.

The International Conference on Computational Science 2003 (ICCS 2003) is the follow-up to these earlier conferences. ICCS 2003 is unique, in that it was a single event held at two different sites almost opposite each other on the globe – Melbourne, Australia and St. Petersburg, Russian Federation. The conference ran on the same dates at both locations and all the presented work was published in a single set of proceedings, which you hold in your hands right now.

ICCS 2003 brought together experts from a range of disciplines: mathematicians and computer scientists providing basic computing expertise, and researchers and scientists from various application areas who are pioneering advanced applications of computational methods in sciences such as physics, chemistry, life sciences, engineering, arts and humanities; along with software developers and vendors. The intent was to discuss problems and solutions in these areas, identify new issues, and shape future directions for research, as well as help industrial users apply advanced computational techniques.

Many of the advances in computational science are related to Grid Computing. The Grid has provided a way to link computation, data, networking, instruments and other resources together to solve today's complex and critical problems. As such, it is becoming a natural environment for the computational sciences. In these proceedings you will find original research in this new era of computational science and the challenges involved in building the information infrastructure needed to enable science and engineering discoveries of the future.

These four volumes, LNCS 2657, 2658, 2659 and 2660, contain the proceedings of the ICCS 2003 meeting. The volumes consist of over 460 peer-reviewed, contributed and invited papers presented at the conference in Melbourne, Australia and St. Petersburg, Russian Federation. The acceptance rate for oral pre-

sentations was 40% of the submitted papers. The papers presented reflect the aim of the scientific organization to bring together major players in the emerging field of computational science.

The conference included 27 workshops (10 in St. Petersburg and 17 in Australia), 6 presentations by Keynote speakers, and over 460 contributed papers selected for oral presentations and posters. Each paper/poster was refereed by at least two referees.

We are deeply indebted to all the authors who submitted high-quality papers to the conference, without this depth of support and commitment there would have been no conference at all. We acknowledge the members of the program committee and all those involved in the refereeing process, and the workshop organizers and all those in the community who helped us to convene a successful conference. Special thanks go to Dick van Albada, Martin Lack, Zhiming Zhao and Yan Xu for preparation of the proceedings; they did a marvelous job! Amitava Datta, Denis Shamonin, Mila Chevalier, Alexander Boukhanovsky and Elena Stankova are acknowledged for their assistance in the organization and all those 1001 things that need to be done to make a large (distributed!) conference like this a success!

Of course ICCS 2003 would not have been possible without the support of our sponsors, and we therefore gratefully acknowledge their help in the realization of this conference.

Amsterdam, June 2003

Peter M.A. Sloot,

on behalf of the co-editors:
David Abramson
Alexander Bogdanov
Jack J. Dongarra
Albert Zomaya
Yuriy Gorbachev

Organization

The conference was organized by the Section Computational Science, The University of Amsterdam, The Netherlands; the Innovative Computing Laboratory at The University of Tennessee, USA; the School of Computer Science and Software Engineering, Monash University, Victoria, Australia; the School of Information Technologies, The University of Sydney, New South Wales, Australia; and the Institute for High Performance Computing and Information Systems, St. Petersburg, Russian Federation.

Conference Chairs

Alexander Bogdanov, Chair of the St. Petersburg ICCS 2003 conference site
David Abramson, Chair of the Melbourne ICCS 2003 conference site
Jack J. Dongarra, Scientific and Overall Co-chair (The University of Tennessee, Knoxville, USA)
Peter M.A. Sloot, Scientific and Overall Chair (The University of Amsterdam, The Netherlands)

Workshops Organization and Program Chairs

Yuriy Gorbachev (IHPCIS, St. Petersburg, Russian Federation)
Albert Zomaya (The University of Sydney, Australia)

Local Organizing Committees

Martin Lack & Associates Pty. Ltd. (Australia)
Elena Stankova (IHPCIS, Russian Federation)
Alexander Boukhanovsky (IHPCIS, Russian Federation)
Mila Chevalier (NIP, Russian Federation)

Program Committee

Albert Y. Zomaya (The University of Sydney, Australia)
Alexander Bogdanov (IHPCIS, Russia)
Alexander Zhmakin (PhTI RAS, Russian Federation)
Alfons Hoekstra (The University of Amsterdam, The Netherlands)
Alistair Rendell (Australian National University, Australia)

Andrzej M. Goscinski (Deakin University, Australia)
Antonio Lagana (University of Perugia, Italy)
Azzedine Boukerche (University of North Texas, USA)
Bastien Chopard (University of Geneva, Switzerland)
Beniamino Di Martino (Seconda Universita' di Napoli, Italy)
Bernard Pailthorpe (The University of Queensland, Australia)
Dale Shires (US Army Research Laboratory, USA)
David A. Bader (University of New Mexico, USA)
Dick van Albada (The University of Amsterdam, The Netherlands)
Dieter Kranzlmueller (Johannes Kepler University Linz, Austria)
Edward Moreno (Euripides Foundation of Marilia, Brazil)
Elena Zudilova (The University of Amsterdam, The Netherlands)
Francis Lau (The University of Hong Kong, Hong Kong)
Geoffrey Fox (Indiana University, USA)
Graham Megson (The University of Reading, UK)
Greg Watson (LANL, USA)
Hai Jin (Huazhong University of Science and Technology, China)
Hassan Diab (American University of Beirut, Lebanon)
Hong Shen (Japan Advanced Institute of Science and Technology, Japan)
James Glimm (Stony Brook University, USA)
Jemal H. Abawajy (Carleton University, Canada)
Jerzy Wasniewski (UNI-C Danish IT Center for Education and Research, Denmark)
Jesús Vigo-Aguiar (University of Salamanca, Spain)
Jose Laginha Palma (University of Porto, Portugal)
Kevin Burrage (The University of Queensland, Australia)
Koichi Wada (University of Tsukuba, Japan)
Marian Bubak (AGH, Cracow, Poland)
Matthias Müller (University of Stuttgart, Germany)
Michael Johnson (The University of Sydney, Australia)
Michael Mascagni (Florida State University, USA)
Nikolay Borisov (SPbSU, Russian Federation)
Paul Coddington (University of Adelaide, Australia)
Paul Roe (Queensland University of Technology, Australia)
Peter Kacsuk (MTA SZTAKI Research Institute, Hungary)
Peter M.A. Sloot (The University of Amsterdam, The Netherlands)
Putchong Uthayopas (Kasetsart University, Thailand)
Rajkumar Buyya (Melbourne University, Australia)
Richard Ramaroson (ONERA, France)
Robert Evarestov (SPbSU, Russian Federation)
Rod Blais (University of Calgary, Canada)
Ron Perrott (Queen's University of Belfast, UK)
Rosie Renaut (Arizona State University, USA)
Srinivas Aluru (Iowa State University, USA)
Stephan Olariu (Old Dominion University, USA)

Tarek El-Ghazawil (George Washington University, USA)
Vaidy Sunderam (Emory University, USA)
Valery Zolotarev (SPbSU, Russian Federation)
Vasil Alexandrov (The University of Reading, UK)
Vladimir P. Nechiporenko (Ministry of Industry, Science and Technologies, Russian Federation)
Xiaodong Zhang (National Science Foundation, USA)
Yong Xue (Chinese Academy of Sciences, China)
Yuriy Gorbachev (IHPCIS, Russian Federation)
Zdzislaw Meglicki (Indiana University, USA)

Workshop Organizers

Computer Algebra Systems and Their Applications
A. Iglesias (University of Cantabria, Spain)
A. Galvez (University of Cantabria, Spain)

Computer Graphics
A. Iglesias (University of Cantabria, Spain)

Computational Science of Lattice Boltzmann Modeling
B. Chopard (University of Geneva, Switzerland)
A.G. Hoekstra (The University of Amsterdam , The Netherlands)

Computational Finance and Economics
X. Deng (City University of Hongkong, Hongkong)
S. Wang (Chinese Academy of Sciences, China)

Numerical Methods for Structured Systems
N. Del Buono (University of Bari, Italy)
L. Lopez (University of Bari, Italy)
T. Politi (Politecnico di Bari, Italy)

High-Performance Environmental Computations
E. Stankova (Institute for High Performance Computing and Information Systems, Russian Federation)
A. Boukhanovsky (Institute for High Performance Computing and Information Systems, Russian Federation)

Grid Computing for Computational Science
M. Müller (University of Stuttgart, Germany)
C. Lee (Aerospace Corporation, USA)

Computational Chemistry and Molecular Dynamics
A. Lagana (Perugia University, Italy)

Recursive and Adaptive Signal/Image Processing (RASIP)
I.V. Semoushin (Ulyanovsk State University, Russian Federation)

Numerical Methods for Singular Differential and Differential-Algebraic Equations
V.K. Gorbunov (Ulyanovsk State University, Russian Federation)

Workshop on Parallel Linear Algebra (WoPLA03)

M. Hegland, (Australian National University, Australia)
P. Strazdins (Australian National University, Australia)

Java in Computational Science

A. Wendelborn (University of Adelaide, Australia)
P. Coddington (University of Adelaide, Australia)

Computational Earthquake Physics and Solid Earth System Simulation

P. Mora (Australian Computational Earth Systems Simulator)
H. Muhlhaus (Australian Computational Earth Systems Simulator)
S. Abe (Australian Computational Earth Systems Simulator)
D. Weatherley (QUAKES, Australia)

Performance Evaluation, Modeling and Analysis of Scientific Applications on Large-Scale Systems

A. Hoisie, (LANL, USA)
D.J. Kerbyson, (LANL, USA)
A. Snavely (SDSC, University of California, USA)
J. Vetter, (LLNL, USA)

Scientific Visualization and Human-Machine Interaction in a Problem Solving Environment

E. Zudilova (The University of Amsterdam, The Netherlands)
T. Adriaansen (Telecommunications & Industrial Physics, CSIRO)

Innovative Solutions for Grid Computing

J.J. Dongarra (The University of Tennessee, USA)
F. Desprez (LIP ENS, France)
T. Priol (INRIA/IRISA)

Terascale Performance Analysis

D.A. Reed (NCSA, USA)
R. Nandkumar (NCSA, USA)
R. Pennington (NCSA, USA)
J. Towns (NCSA, USA)
C.L. Mendes (University of Illinois, USA)

Computational Chemistry in the 21st Century: Applications and Methods

T.H. Dunning, Jr. (JICS, ORNL, USA)
R.J. Harrison (ORNL, USA)
L. Radom (Australian National University, Australia)
A. Rendell (Australian National University, Australia)

Tools for Program Development and Analysis in Computational Science

D. Kranzlmueller (Johannes Kepler University, Austria)
R. Wismüller (University of Vienna, Austria)
A. Bode (Technische Universität München, Germany)
J. Volkert (Johannes Kepler University, Austria)

Parallel Input/Output Management Techniques (PIOMT2003)
J.H. Abawajy (Carleton University, Canada)
Dynamic Data Driven Application Systems
F. Darema (NSF/CISE, USA)
Complex Problem-Solving Environments for Grid Computing (WCPSE02)
D. Walker (Cardiff University, UK)
Modeling and Simulation in Supercomputing and Telecommunications
Y. Mun (Soongsil University, Korea)
Modeling of Multimedia Sychronization in Mobile Information Systems
D.C. Lee (Howon University, Korea)
K.J. Kim (Kyonggi University, Korea)
OpenMP for Large Scale Applications
B. Chapman (University of Houston, USA)
M. Bull (EPCC, UK)
Modelling Morphogenesis and Pattern Formation in Biology
J.A. Kaandorp (The University of Amsterdam, The Netherlands)
Adaptive Algorithms for Parallel and Distributed Computing Environments
S. Moore (University of Tennessee, USA)
V. Eijkhout (University of Tennessee, USA)

Sponsoring Organizations

The University of Amsterdam, The Netherlands
Hewlett-Packard
Springer-Verlag, Germany
Netherlands Institute in St. Petersburg, (NIP)
Ministry of Industry, Science and Technologies of the Russian Federation
Committee of Science and High Education of the Government of St. Petersburg
St. Petersburg State Technical University
Institute for High Performance Computing and Information Systems, St. Petersburg
IBM Australia
Microsoft
Cray Inc.
Dolphin Interconnect
Microway
Etnus
ceanet
NAG
Pallas GmbH

Table of Contents, Part IV

Australian Track

Workshop on Terascale Performance Analysis

A Model for Predicting the Optimum Checkpoint Interval for Restart Dumps 3
John Daly

Terascale I/O Solutions 13
Nathan Stone, John Kochmar, Paul Nowoczynski, J. Ray Scott, Derek Simmel, Jason Sommerfield, Chad Vizino

Scaling Molecular Dynamics to 3000 Processors with Projections: A Performance Analysis Case Study 23
Laxmikant V. Kalé, Sameer Kumar, Gengbin Zheng, Chee Wai Lee

Real-Time Terascale Implementation of Tele-immersion 33
Nikhil Kelshikar, Xenophon Zabulis, Jane Mulligan, Kostas Daniilidis, Vivek Sawant, Sudipta Sinha, Travis Sparks, Scott Larsen, Herman Towles, Ketan Mayer-Patel, Henry Fuchs, John Urbanic, Kathy Benninger, Raghurama Reddy, Gwendolyn Huntoon

Performance Analysis of PHASTA on NCSA Intel IA-64 Linux Cluster 43
Wai Yip Kwok

Performance Instrumentation and Measurement for Terascale Systems . . . 53
Jack Dongarra, Allen D. Malony, Shirley Moore, Philip Mucci, Sameer Shende

Logging Kernel Events on Clusters 63
Jürgen Reuter, Walter F. Tichy

Workshop on Computational Chemistry in the 21st Century: Applications and Methods

Recent Advances in QM and QM/MM Methods 75
Mark S. Gordon, Michael W. Schmidt

UTChem – A Program for *ab initio* Quantum Chemistry 84
Takeshi Yanai, Haruyuki Nakano, Takahito Nakajima, Takao Tsuneda, So Hirata, Yukio Kawashima, Yoshihide Nakao, Muneaki Kamiya, Hideo Sekino, Kimihiko Hirao

Local Correlation Models 96
Martin Head-Gordon, Troy Van Voorhis, Gregory J.O. Beran, Barry Dunietz

Multiresolution Quantum Chemistry in Multiwavelet Bases 103
Robert J. Harrison, George I. Fann, Takeshi Yanai, Gregory Beylkin

Management of Web and Associated Grid Technologies for Quantum Chemistry Computation 111
Kim K. Baldridge, Jerry P. Greenberg

The Extensible Computational Chemistry Environment: A Problem Solving Environment for High Performance Theoretical Chemistry 122
Gary Black, Karen Schuchardt, Debbie Gracio, Bruce Palmer

Computational Quantum Chemistry Experiments via the Web 132
Brian F. Yates, Brian J. Duke

Tonto: A Fortran Based Object-Oriented System for Quantum Chemistry and Crystallography 142
Dylan Jayatilaka, Daniel J. Grimwood

Explicitly Correlated Second Order Perturbation Theory with Frozen Gaussian-Type Geminals 152
Seiichiro Ten-no

Molecular Potential Energy Surfaces by Interpolation 159
Michael A. Collins

NWChem: New Functionality 168
Theresa L. Windus, Eric J. Bylaska, Michel Dupuis, So Hirata, Lisa Pollack, Dayle M. Smith, T.P. Straatsma, Edoardo Aprà

Workshop on Tools for Program Development and Analysis in Computational Science

Interfacing Computer Aided Parallelization and Performance Analysis . . . 181
Gabriele Jost, Haoqiang Jin, Jesus Labarta, Judit Gimenez

Dynamic Performance Tuning of Distributed Programming Libraries 191
Anna Morajko, Oleg Morajko, Josep Jorba, Tomàs Margalef, Emilio Luque

Parallel Program Debugging with MAD – A Practical Approach 201
Dieter Kranzlmüller, Axel Rimnac

Workshop on Parallel Input/Output Management Techniques (PIOMT2003)

Enhancing Prediction Accuracy in PCM-Based File Prefetch by Constrained Pattern Replacement Algorithm 213
Inchul Choi, Chanik Park

Parallel I/O Scheduling in Multiprogrammed Cluster Computing Systems 223
Jemal H. Abawajy

Design and Evaluation of Distributed Smart Disk Architecture for I/O-Intensive Workloads 230
Steve Chiu, Wei-keng Liao, Alok Choudhary

An Adaptive Load Balancing Algorithm for Large Data Parallel Processing with Communication Delay 242
Kenji Imasaki, Jemal H. Abawajy

A Flexible Multiagent Parallel File System for Clusters 248
María S. Pérez, Jesús Carretero, Félix García, José M. Peña, Víctor Robles

A New Proportional-Share Disk Scheduling Algorithm: Trading-Off I/O Throughput and QoS Guarantee 257
Young Jin Nam, Chanik Park

Workshop on Dynamic Data Driven Application Systems

Computational Aspects of Chemical Data Assimilation into Atmospheric Models 269
Gregory R. Carmichael, Dacian N. Daescu, Adrian Sandu, Tianfeng Chai

Virtual Telemetry for Dynamic Data-Driven Application Simulations 279
Craig C. Douglas, Yalchin Efendiev, Richard Ewing, Raytcho Lazarov, Martin J. Cole, Greg Jones, Chris R. Johnson

Tornado Detection with Support Vector Machines 289
Theodore B. Trafalis, Huseyin Ince, Michael B. Richman

Computational Science Simulations Based on Web Services 299
Paul Chew, Nikos Chrisochoides, S. Gopalsamy, Gerd Heber, Tony Ingraffea, Edward Luke, Joaquim Neto, Keshav Pingali, Alan Shih, Bharat Soni, Paul Stodghill, David Thompson, Steve Vavasis, Paul Wawrzynek

DDEMA: A Data Driven Environment for Multiphysics Applications 309
John Michopoulos, Panagiota Tsompanopoulou, Elias Houstis, John Rice, Charbel Farhat, Michel Lesoinne, Frederic Lechenault

Discrete Event Solution of Gas Dynamics within the DEVS Framework . . 319
J. Nutaro, B.P. Zeigler, R. Jammalamadaka, S. Akerkar

Data Driven Design Optimization Methodology: A Dynamic Data Driven Application System 329
Doyle Knight

Derivation of Natural Stimulus Feature Set Using a Data-Driven Model 337
Alexander G. Dimitrov, Tomas Gedeon, Brendan Mumey, Ross Snider, Zane Aldworth, Albert E. Parker, John P. Miller

Performance Evaluation of Generalized Polynomial Chaos 346
Dongbin Xiu, Didier Lucor, C.-H. Su, George Em Karniadakis

Driving Scientific Applications by Data in Distributed Environments 355
Joel Saltz, Umit Catalyurek, Tahsin Kurc, Mike Gray, Shannon Hastings, Steve Langella, Sivaramakrishnan Narayanan, Ryan Martino, Steven Bryant, Malgorzata Peszynka, Mary Wheeler, Alan Sussman, Michael Beynon, Christian Hansen, Don Stredney, Dennis Sessanna

Simulating Sellers' Behavior in a Reverse Auction B2B Exchange 365
Subhajyoti Bandyopadhyay, Alok R. Chaturvedi, John M. Barron, Jackie Rees, Shailendra Mehta

Rapid Real-Time Interdisciplinary Ocean Forecasting Using Adaptive Sampling and Adaptive Modeling and Legacy Codes: Component Encapsulation Using XML 375
Constantinos Evangelinos, Robert Chang, Pierre F.J. Lermusiaux, Nicholas M. Patrikalakis

A Computational Infrastructure for Reliable Computer Simulations 385
J. Tinsley Oden, James C. Browne, Ivo Babuška, Kenneth M. Liechti, Leszek F. Demkowicz

Workshop on Complex Problem-Solving Environments for Grid Computing (WCPSE02)

Automating Metadata Web Service Deployment for Problem Solving Environments 393
Ozgur Balsoy, Ying Jin, Galip Aydin, Marlon Pierce, Geoffrey Fox

Building Problem Solving Environments with Application Web Service Toolkits 403
Choonhan Youn, Marlon Pierce, Geoffrey Fox

GSF: A Problems Solving Environment Supporting Multi-models Parallel Grid Programming 413
Qian-ni Deng, Xin-da Lu

Implementation of a Grid-Enabled Problem Solving Environment in Matlab 420
Hakki Eres, Graeme Pound, Zhouan Jiao, Jasmin Wason, Fenglian Xu, Andy Keane, Simon Cox

GSiB: PSE Infrastructure for Dynamic Service-Oriented Grid Applications 430
Yan Huang

Workshop on Modeling and Simulation in Supercomputing and Telecommunications

Partial Dual Unicasting Based Handoff for Real-Time Traffic in MIPv6 Networks 443
DaeKyu Choi, Hyunseung Choo

Simulations on Batching in Video-on-Demand Transmissions 453
Juan Segarra, Vicent Cholvi

The Design of XML-Based Internet Security Integrated System Architecture 463
Kwang H. Kim, Tae-Kyung Kim, Dong S. Kim, Tai M. Chung

Performance Analysis of a Mobility Support Scheme between Subnetworks in Differentiated Services 473
Jongoh Kim, Jonghyun Choi, Youngsong Mun

Performance Analysis for Real-Time Grid Systems on COTS Operating Systems 482
Eui-Nam Huh, Youngsong Mun

Scalable Keyframe Extraction Using One-Class Support Vector Machine 491
YoungSik Choi, Sangyoun Lee

A Design and Performance Evaluation of 2-Level Call Admission Control Scheme in Mobile Multimedia Networks 500
Myung Il Kim, Sung Jo Kim

Performance Evaluation of a Relocation Method for Mobile Computing Maintaining Data Consistency 510
Byung-Kwan Lee, Am-Suk Oh

Reliability Problem on All Pairs Quickest Paths 518
Young-Cheol Bang, Hyunseung Choo, Youngsong Mun

A Concurrency Control Algorithm for Firm Real-Time Database Systems 524
Seok Jae Lee, Jae Ryong Shin, Seok Il Song, Jae Soo Yoo, Ki Hyung Cho

Workshop on Modeling of Multimedia Synchronization in Mobile Information System

Secure Firewall Traversal in Mobile IP Network 535
Jung-Min Park, Min-Jung Jin, Kijoon Chae

Buffering Management Scheme for Multimedia Synchronization in Mobile Information System 545
Gi-Sung Lee, Jeung-gyu Jee, Sok-Pal Cho

Resource Reservation and Allocation Based on Direction Prediction for Handoff in Mobile Multimedia Networks 555
Jongchan Lee, Hongjin Kim, Kuinam J. Kim

Petri-Net Model of Multimedia Synchronization in Mobile Environment 566
Keun-Wang Lee, Jong-Hee Lee, Hae-Seok Oh

New Mobility Scheme for Reducing Location Traffic in Mobile Networks 576
Il-Sun Hwang, Myungsun Lee, Ki-sung Yoo, Jin-wook Chung

Mobile Animation Algorithm for Cyber Museum 586
Sung-Soo Hong, Sang-Kil Kim

Certificate Management Client System for E-transactions on Internet 596
Jeom goo Kim, Yoochan Ra, Jaehwan Lim

Threat Description for the PP by Using the Concept of the Assets Protected by TOE 605
Tai-hoon Kim, Byung-gyu No, Dong Chun Lee

Real-Time Data Management for Network Information System 614
Dong Chun Lee

Efficient Migration Algorithm for Mobile Agents in Wired/Wireless Internets 626
Sukwoo Han, Kuinam J. Kim, Young-Sung Kim

Workshop on OpenMP for Large Scale Applications

OpenMP in the Field: Anecdotes from Practice 637
Russell K. Standish, Clinton Chee, Nils Smeds

OpenMP and NUMA Architectures I: Investigating Memory Placement on the SCI Origin 3000 648
Nathan Robertson, Alistair Rendell

Poor Scalability of Parallel Shared Memory Model: Myth or Reality? 657
Mark Kremenetsky, Arthur Raefsky, Steve Reinhardt

Asynchronous Execution of OpenMP Code 667
Tien-hsiung Weng, Barbara Chapman

Workshop on Modeling Morphogenesis and Pattern Formation in Biology

Virtual Reef: A Visualisation Framework for Marine Simulation Models . . 679
Stephen Jeffrey

Modelling Developmental Regulatory Networks 688
Tommy Krul, Jaap A. Kaandorp, Joke G. Blom

A Biomechanical Imitative Model for Pulsatile Morphogenesis 698
L.V. Beloussov, V.I. Grabovsky

Using Computational Plant Science Tools to Investigate Morphological Aspects of Compensatory Growth 708
David Thornby, Michael Renton, Jim Hanan

3D Visualization of Complex Surfaces Using Dynamic Delaunay Tessellation 718
M.L. Gavrilova, J. Pivovarov

Workshop on Adaptive Algorithms for Parallel and Distributed Computing Environments

CODELAB: A Developers' Tool for Efficient Code Generation and Optimization 729
Dragan Mirković, S. Lennart Johnsson

A Knowledge Discovery Methodology for Behavior Analysis of Large-Scale Applications on Parallel Architectures 739
Elias N. Houstis, Vassilios S. Verykios, Ann C. Catlin, John R. Rice

Performance Modeling for Dynamic Algorithm Selection 749
Michael O. McCracken, Allan Snavely, Allen Malony

Self-Adapting Numerical Software and Automatic Tuning of Heuristics 759
Jack Dongarra, Victor Eijkhout

Poster Papers

Computer Simulation of Multi-dimensional Bone Deformity Correction and Lengthening by Unilateral External Fixator 771
Yoon Hyuk Kim

Problems and Prospects for Quantum Computational Speed-up 779
E.V. Krishnamurthy

A Configuration Algorithm for Provisioning of Cell-Based MPLS Network 789
Daniel Won-Kyu Hong, Choong Seon Hong, Dong-Sik Yun

Exact and Approximate Algorithms for Topological Design of Wide Area Networks with Non-simultaneous Single Commodity Flows 799
Andrzej Kasprzak

Modeling and Simulation of Distributed Security Models 809
Hee Suk Seo, Tae Ho Cho, Sung Do Chi

An Efficient Collective Communication Method for Grid Scale Networks 819
Kyung-Lang Park, Hwang-Jik Lee, Youn-Joo Lee, Oh-Young Kwon, Sung-Yong Park, Hyung-Woo Park, Shin-Dug Kim

Reliability Theory Model and Expected Life Shortest Path in Stochastic and Time-Dependent Networks 829
Guo-zhen Tan, Xiang-fu Xia, Wen Gao

Using a GA Adaptor in Multi-applications 839
Cheng Wei Hang, Yen Cheung

Automated Debugging in Java Programs Using HDM 849
Hoon-Joon Kouh, Weon-Hee Yoo

Pattern Based Approaches to Pre-processing Structured Text: A Newsfeed Example 859
Paul Bogg

Evolutionary Rule Generation Classification and Its Application to Multi-class Data 868
Susan E. Bedingfield, Kate A. Smith

Self-Organizing Hybrid Neurofuzzy Networks 877
Sung-Kwun Oh, Su-Chong Joo, Chang-Won Jeong, Hyun-Ki Kim

FOM: A Framework for Metaheuristic Optimization 886
J.A. Parejo, J. Racero, F. Guerrero, T. Kwok, K.A. Smith

ClassdescMP: Easy MPI Programming in C++ 896
Russell K. Standish, Duraid Madina

A Novel Evolutionary Approach to Linear Time-Series Forecasting Model 903
Prakash Vijayan, S. Suresh

Design and Implementation of XML DBMS Based on Generic Data Model 911
JongSeon Lim, Sung yoon Bang, Kyung-Soo Joo

Developing a Unified Design Methodology Based on Extended Entity-Relationship Model for XML 920
Mun-Young Choi, JongSeon Lim, Kyung-Soo Joo

Investigating Neural Network Modeling Decisions for the Australian All-Ordinaries Index 930
Andrew Flitman, Mark Barnes, Deniss Teng Tai Kiat

Rapid Prototyping Methodology and Environments for Fuzzy Applications 940
Chantana Chantrapornchai

Multiple Correlation Analysis of Fuzzy Sets 950
Nancy P. Lin

An Efficient Dispersal and Encryption Scheme for Secure Distributed Information Storage 958
Sung Jin Choi, Hee Yong Youn, Bo Kyoung Lee

A DTD for an XML-Based Mathematical Modeling Language 968
Marcos Calle, S. Lozano, Kate Smith, Terence Kwok, Juan J. Domínguez

A Compress-Based Association Mining Algorithm for Large Dataset 978
Mafruz Zaman Ashrafi, David Taniar, Kate Smith

Engineering Persistent Queue System for a Unified Stock Transaction Platform 988
Jingcun Wang, Carol Blum

An Efficient Algorithm for Computing Inverses in GF(2^m) Using Dual Bases 994
Hyeong Seon Yoo, Seok Ung Yoon, Eui Sun Kim

Object Oriented Parallel Programming Model on a Network of Workstations 1000
B. Suresh, R. Nadarajan

Paroxysmal Atrial Fibrillation Prediction Application Using Genetic Algorithms 1011
Sonia Mota, Eduardo Ros, Francisco de Toro, Julio Ortega

A Metadata Tool for Retrieval from Heterogeneous Distributed XML Documents 1020
Young-Kwang Nam, Joseph Goguen, Guilian Wang

Effective Similarity Search Methods for Large Video Data Streams 1030
Seok-Lyong Lee, Seok-Ju Chun, Ju-Hong Lee

A New Method for Locating the Global Optimum: Application of the Cutting Angle Method to Molecular Structure Prediction 1040
Kieran F. Lim, Gleb Beliakov, Lynn Batten

Performance Characteristics of a Cost-Effective Medium-Sized Beowulf Cluster Supercomputer 1050
Andre L.C. Barczak, Chris H. Messom, Martin J. Johnson

A Novel Approach to Role-Based Access Control 1060
Song-hwa Chae, Wonil Kim, Dong-kyoo Kim

Designated Verification of Digital Watermark for Network Based Image Distribution 1069
Hyung-Woo Lee, Im-Yeong Lee

A Parallel Loop Self-Scheduling on Extremely Heterogeneous PC Clusters 1079
Chao-Tung Yang, Shun-Chyi Chang

Robust Reference Point Detection Using Gradient of Fingerprint Direction and Feature Extraction Method 1089
Junbum Park, Hanseok Ko

GPD-Based State Modification by Weighted Linear Loss Function 1100
Taehee Kwon, Hanseok Ko

Spectral Subtraction Using Spectral Harmonics for Robust Speech Recognition in Car Environments 1109
Jounghoon Beh, Hanseok Ko

A Workflow Management and Grid Computing Approach to Molecular Simulation-Based Bio/Nano Experiments 1117
Karpjoo Jeong, Dongwook Kim, Moon Hae Kim, Suntae Hwang, Seunho Jung, Youngho Lim, Sangsan Lee

Simulation Infrastructure for Modeling Large Scale Neural Systems 1127
Charles C. Peck, James Kozloski, A. Ravishankar Rao, Guillermo A. Cecchi

Monotone Iterative Methods and Schwarz Methods for Nonlinear Parabolic PDE with Time Delay 1137
Daoud S. Daoud

Author Index 1149

Table of Contents, Part I

Russian Track

Keynote Paper

Industrial Challenges for Numerical Simulation of Crystal Growth 3
Dmitry K. Ofengeim, Alexander I. Zhmakin

Track on Scientific Computing in Mathematics, Mechanis, Physics, and Chemistry

Three-Way Decomposition and Nuclear Magnetic Resonance 15
Martin Billeter, Vladislav Orekhov

Semi-lagrangian Semi-implicit Fully Splitted Hydrostatic Atmospheric Model 25
Andrei Bourchtein

Object-Oriented Framework for Modelling of Pollutant Transport in River Network 35
Gennadiy Donchyts, Mark Zheleznyak

Heat Transfer, Newton's Law of Cooling and the Law of Entropy Increase Simulated by the Real-Time Computer Experiment in Java 45
Adam Galant, Ryszard Kutner, Andrzej Majerowski

Parallel Solution of the Poisson-Boltzmann Equation for Proteins 54
Shura Hayryan, Chin-Kun Hu, Edik Hayryan, Imrikh Pokorny

Monte Carlo Method for Calculating the Electrostatic Energy of a Molecule 63
Michael Mascagni, Nikolai A. Simonov

Track on New Large-Scale Simulation Application Areas

A Model of Immune Suppression and Repertoire Evolution 75
John Burns, Heather J. Ruskin

Julia Sets and Yang-Lee Zeros of the Potts Model on Bethe Lattices 85
Ruben Ghulghazaryan, Nerses Ananikyan, Tore M. Jonassen

Dynamic Mean Semi-variance Portfolio Selection 95
Ali Lari-Lavassani, Xun Li

Computer Simulations of Multiparticle-Contacts Dynamics 105
Jacek S. Leszczynski

Studying System Properties with Rough Sets 115
Alicja Mieszkowicz-Rolka, Leszek Rolka

Stochastic Modeling of Temporal Variability of HIV-1 Population 125
Ilia Kiryukhin, Kirill Saskov, Alexander Boukhanovsky, Wilco Keulen, Charles Boucher, Peter M.A. Sloot

Modeling Traffic of Information Packets on Graphs with Complex Topology 136
Bosiljka Tadić

Track on Parallel and Distributed Computing

An Efficient Load Balancing Algorithm Employing a Symmetric Balanced Incomplete Block Design 147
Okbin Lee, Sangho Lee, Seongyeol Kim, Ilyong Chung

A Parallel Virtual Machine for Bulk Synchronous Parallel ML 155
Frédéric Gava, Frédéric Loulergue

ParoC++: A Requirement-Driven Parallel Object-Oriented Programming Language 165
Tuan-Anh Nguyen, Pierre Kuonen

Parallelization Scheme for an Approximate Solution to Time Constraint Problems 175
Tuan-Anh Nguyen, Pierre Kuonen

About the Relationship between Election Problem and Failure Detector in Asynchronous Distributed Systems 185
Sung-Hoon Park

Two Phases Based Cluster Formation Scheme for Mobile Ad Hoc Networks 194
KiCheoul Wang, Gihwan Cho

Track on Grid Computing and Hybrid Computational Methods

Architecture of the Grid for Interactive Applications 207
Marian Bubak, Maciej Malawski, Katarzyna Zając

Monitoring and Performance Analysis of Grid Applications 214
Bartosz Baliś, Marian Bubak, Włodzimierz Funika, Tomasz Szepieniec, Roland Wismüller

A Parallel Programming Environment on Grid 225
Weiqin Tong, Jingbo Ding, Lizhi Cai

Software Development in the Grid: The DAMIEN Tool-Set 235
Edgar Gabriel, Rainer Keller, Peggy Lindner, Matthias S. Müller, Michael M. Resch

An XML-Based Framework for Loosely Coupled Applications on Grid Environments 245
Andreas Hoheisel, Uwe Der

Grid Service Provider: How to Improve Flexibility of Grid User Interfaces? 255
Maciej Bogdanski, Michal Kosiedowski, Cezary Mazurek, Malgorzata Wolniewicz

The NorduGrid Architecture and Middleware for Scientific Applications 264
O. Smirnova, P. Eerola, T. Ekelöf, M. Ellert, J.R. Hansen, A. Konstantinov, B. Kónya, J.L. Nielsen, F. Ould-Saada, A. Wäänänen

Efficient Synchronization of Replicated Data in Distributed Systems 274
Thorsten Schütt, Florian Schintke, Alexander Reinefeld

Experimental Grid Access for Dynamic Discovery and Data Transfer in Distributed Interactive Simulation Systems 284
Alfredo Tirado-Ramos, Katarzyna Zając, Zhiming Zhao, Peter M.A. Sloot, Dick van Albada, Marian Bubak

Track on New Algorithmic Approaches to Existing Application Areas

The Uniform Posture Map Algorithm for the Real-Time Interactive Motion Transitions of an Articulated Body 295
Jin Ok Kim, Bum Ro Lee, Chin Hyun Chung, Jun Hwang, Woongjae Lee

The Inductive Inverse Kinematics Algorithm to Manipulate the Posture of an Articulated Body 305
Jin Ok Kim, Bum Ro Lee, Chin Hyun Chung, Jun Hwang, Woongjae Lee

On the Extraction of the Valid Speech-Sound by the Merging Algorithm with the Discrete Wavelet Transform 314
Jin Ok Kim, Han Wook Paek, Chin Hyun Chung, Jun Hwang, Woongjae Lee

Higher Order Temporal Rules 323
Paul Cotofrei, Kilian Stoffel

A Cellular Automata Model for Simulating Pyroclastic Flows and First Application to 1991 Pinatubo Eruption 333
Gino M. Crisci, Salvatore Di Gregorio, Rocco Rongo, William Spataro

Properties of the Genetic Code under Directional, Asymmetric Mutational Pressure 343
Małgorzata Dudkiewicz, Paweł Mackiewicz, Aleksandra Nowicka, Maria Kowalczuk, Dorota Mackiewicz, Natalia Polak, Kamila Smolarczyk, Mirosław R. Dudek, Stanisław Cebrat

Nonlinear Internal Model Control Using Neural Networks and Fuzzy Logic: Application to an Electromechanical Process 351
Rodolfo E. Haber, J.R. Alique, A. Alique, R.H. Haber

Computer Simulation of Stochastic Models with SDE-Solver Software Package 361
Aleksander Janicki, Adam Izydorczyk, Przemyslaw Gradalski

Hierarchical Neural Network Based Compression of ECG Signals 371
Bekir Karlik

A Nonlinear Model for Predicting ECG R-R Interval Variation Based on the Evolutionary Computation Approach 378
Bo Yeon Kim, Yun Seok Chang, Kwang Suk Park

An Efficient Election Protocol in a Mobile Computing Environment 387
Sung-Hoon Park

Very Large Bayesian Networks in Text Classification 397
Mieczysław A. Kłopotek, Marcin Woch

Stochastic Simulation of Time Series by Using the Spatial-Temporal Weierstrass Function 407
Ryszard Kutner, Filip Świtała

Nonlinear Time Series Prediction by Weighted Vector Quantization 417
A. Lendasse, D. Francois, V. Wertz, M. Verleysen

Simulation and Modelling of Computer Networks 427
Antoni Izworski, Slawomir Skowronski, Jozef B. Lewoc

A Parallel 3D Piecewise Constant Reconstruction Algorithm for Asymmetric Virus Structures 437
Robert E. Lynch, Yongchang Ji, Dan C. Marinescu, Hong Lin

Computational Challenges in Multiple Wavetable Interpolation Synthesis 447
Jonathan Mohr, Xiaobo Li

Segment Self-Guide Reconstruction Algorithm Based on Object-Oriented Quantization 457
Xuanqin Mou, Hengyong Yu, Yuanlong Cai

Analytical Approach for Channel Assignments in Cellular Networks 466
Vladimir V. Shakhov, Hyunseung Choo

An Analysis of the Reliability Overhead Generated by the JRM-Protocol 474
Gunther Stuer, Jan Broeckhove, Frans Arickx

A New Reduced Rank Square Root Kalman Filter for Data Assimilation in Mathematical Models 482
Dimitri Treebushny, Henrik Madsen

A Synchronization Problem on 1-Bit Communication Cellular Automata 492
Hiroshi Umeo, Koshi Michisaka, Naoki Kamikawa

A New Approach to Survivability of Connection Oriented Networks 501
Krzysztof Walkowiak

Creativity and Delusions: The Dopaminergic Modulation of Cortical Maps 511
Luís Alfredo Vidal de Carvalho, Daniele Quintella Mendes, Roseli S. Wedemann

Some Notes on Numerical Convergence of the Stochastic Perturbation Method 521
Marcin Kamiński

Track on Advanced Numerical Algorithms

Transforming Pseudo-Triangulations 533
Sergei Bespamyatnikh

Extrapolated Multistep Methods and Local-Global Step Size Control 540
Gennady Y. Kulikov, Sergey K. Shindin

Robust Multigrid Technique 550
Serguei Martynenko

Comparison of 3D Flow Fields Arising in Mixed and Standard Unstructured Finite Elements 560
Annamaria Mazzia, Giorgio Pini, Mario Putti, Flavio Sartoretto

Improved Objective Functions for Tetrahedral Mesh Optimisation 568
Rafael Montenegro, José María Escobar, Eduardo Rodríguez, Gustavo Montero, José María González-Yuste

Track on Problem Solving Environments (Including: Visualisation Technologies, Web Technologies, and Software Component Technologies

A Flexible Infrastructure for the Support of Distributed Learning 581
Manuel Caeiro, Luis Anido, Martín Llamas, Luis M. Álvárez, Fernando A. Mikic

Maximum Disparity Threshold Estimation for Stereo Imaging Systems via Variogram Analysis 591
Bogusław Cyganek, Jan Borgosz

A Fuzzy Artificial World: Zamin II 601
Ramin Halavati, Saeed Bagheri Shouraki

Fast Parallel Algorithm for Volume Rendering and Its Experiment on Computational Grid 610
Hyung-Jun Kim, Sung-Up Jo, Yong-won Kwon, So-Hyun Ryu, Yong-je Woo, Chang-Sung Jeong, Hyoungwoo Park

A Hand Tracking for a Human Computer Interaction System by the Modified Block Matching Algorithm 619
Jin Ok Kim

Flexible Component Architecture for Information WEB Portals 629
Łukasz Dutka, Jacek Kitowski

A Problem Solving Environment for Modelling Stony Coral Morphogenesis 639
Roeland Merks, Alfons Hoekstra, Jaap Kaandorp, Peter M.A. Sloot

Component Software Systems for Surface Water Simulation 649
L. Postma, P.M.A. Boderie, J.A.G. van Gils, J.K.L. van Beek

A Web Services Broker for E-learning 659
Judith Rodríguez, Luis Anido, Manuel J. Fernández, Fernando Mikic, Luis M. Álvárez

2D and 3D Numerical Models of the Growth of Foraminiferal Shells 669
Paweł Łabaj, Paweł Topa, Jarosław Tyszka, Witold Alda

ISS-Studio: A Prototype for a User-Friendly Tool for Designing Interactive Experiments in Problem Solving Environments 679
Z. Zhao, G.D. van Albada, A. Tirado-Ramos, K. Zajac, P.M.A. Sloot

Track on Computer Algebra Systems and Their Applications

Romberg Integration: A Symbolic Approach with Mathematica 691
Ali Yazıcı, Tanıl Ergenç, Irfan Altas

Numerical Continuation of Fold Bifurcations of Limit Cycles in MATCONT 701
Annick Dhooge, Willy Govaerts, Yuri A. Kuznetsov

Computer Algebra System Maple: A New Software Library 711
Victor Aladjev

Computer Aided Symbolic Linear Algebra 718
Juan M. de Olazábal

Computing Maximal Tori Using LiE and Mathematica 728
Alfred G. Noël

Error Correcting Codes with Mathematica 737
Igor Gashkov

Computer Algebra for Analyzing the Vibrations of Nonlinear Structures 747
Algimantas Čepulkauskas, Regina Kulvietienė, Genadijus Kulvietis

A webMathematica Application for Mathematics Learning 754
Giovannina Albano, Bernardo D'Auria, Saverio Salerno

Checking the Four-Valued Boolean Algebra by the Use PLANNER 764
Vitaly Eltekov

An Application with webMathematica 774
Ünal Ufuktepe

CAS Mathematica in Random Studies 781
Igor E. Poloskov

Strategies for the Interactive Resolution of Calculus Problems 791
Fernando Díez, Roberto Moriyón

A Web-Based Intelligent System for Geometric Discovery 801
Francisco Botana

Computer Algebra for Fingerprint Matching 811
Stefano Bistarelli, Giandomenico Boffi, Fabio Rossi

Basic Calculus on Time Scale with Mathematica 821
Ahmet Yantır, Ünal Ufuktepe

SyNRAC: A Maple-Package for Solving Real Algebraic Constraints 828
Hirokazu Anai, Hitoshi Yanami

Symbolic Computation of High-Order Exact Picard Iterates for Systems of Linear Differential Equations with Time-Periodic Coefficients 838
Armando G.M. Neves

Binary Distillation Column Design Using Mathematica 848
Akemi Gálvez, Andrés Iglesias

Symbolic Solution of Kepler's Generalized Equation 858
Juan Félix San-Juan, Alberto Abad

Workshop on Computer Graphics

Approximation Algorithms to Minimum Vertex Cover Problems on Polygons and Terrains 869
Ana Paula Tomás, António Leslie Bajuelos, Fábio Marques

Automatic Determination of Algebraic Surfaces as Loci of Points 879
Francisco Botana

Probability Distribution of Index Distances in Normal Index Array for Normal Vector Compression 887
Deok-Soo Kim, Youngsong Cho, Donguk Kim, Hyun Kim

A New Differential Approach for Parametric-Implicit Surface Intersection 897
J. Puig-Pey, A. Gálvez, A. Iglesias

Accelerate Volume Splatting by Using Run Length Encoding 907
Jiawan Zhang, Jizhou Sun, Zhigang Sun

A Fragment Culling Technique for Rendering Arbitrary Portals 915
Nick Lowe, Amitava Datta

Efficient Representation of Triangle Meshes for Simultaneous Modification and Rendering 925
Horst Birthelmer, Ingo Soetebier, Jörg Sahm

A New Architecture for Simulating the Behavior of Virtual Agents 935
F. Luengo, A. Iglesias

Workshop on Computational Science of Lattice Boltzmann Modeling

Lattice Boltzmann Method to Study Hydrodynamic Properties of 2D Fractal Aggregates 947
Hung P. Nguyen, Bastien Chopard, Serge Stoll

Lattice Boltzmann Simulation of Thermal Microflows with Heterogeneus Catalysis 957
Sauro Succi

Flow Interference in Bluff Body Wakes 967
Rodrigo Surmas, Luís Orlando Emerich dos Santos, Paulo Cesar Philippi

Boundary Conditions for Thermal Lattice Boltzmann Simulations 977
Annunziata D'Orazio, Sauro Succi

A Reduced-Storage Implementation of the Lattice Boltzmann Equation .. 987
R. Argentini, A.F. Bakker, C.P. Lowe

Lattice BGK Simulations of Unsteady Flow in a 2D Elastic Tube 997
Alfons G. Hoekstra, Jos van 't Hoff, Abdel Monim M. Artoli, Peter M.A. Sloot

A Three-Parameter Non-linear Lattice-Boltzmann Model for Ideal Miscible Fluids 1007
Paulo Cesar Facin, Paulo Cesar Philippi, Luís Orlando Emerich dos Santos

Lattice Boltzmann Simulation of Bubble Flows 1015
Takaji Inamuro, Takeshi Ogata, Fumimaru Ogino

Droplet Spreading on Heterogeneous Surfaces Using a Three-Dimensional Lattice Boltzmann Model 1024
A. Dupuis, A.J. Briant, C.M. Pooley, J.M. Yeomans

Lattice Boltzmann, a Robust and Accurate Solver for Interactive Computational Hemodynamics 1034
A.M. Artoli, D. Kandhai, H.C.J. Hoefsloot, A.G. Hoekstra, P.M.A. Sloot

Lattice Boltzmann Analysis of the Flow Reduction Mechanism in Stented Cerebral Aneurysms for the Endovascular Treatment 1044
Miki Hirabayashi, Makoto Ohta, Daniel A. Rüfenacht, Bastien Chopard

Lattice Boltzmann Simulation and Visualisation of Adsorption Processes in Complex Geometries 1054
J. Bernsdorf, U. Jaekel, T. Zeiser, T. Takei, H. Matsumoto, K. Nishizawa

Virtual Fires Experiences with Developing a LBGK Based Real Time Tunnel Fire Simulator for Virtual Environments 1062
Wilhelm Brandstätter, Christian Redl

Lattice Boltzmann Scheme for Diffusion on Triangular Grids 1072
R.G.M. van der Sman

Author Index 1083

Table of Contents, Part II

Russian Track

Workshop on Computational Finance and Economics

Parallel Computing Method of Valuing for Multi-asset European Option 3
Weimin Zheng, Jiwu Shu, Xiaotie Deng, Yonggen Gu

A Fuzzy Approach to Portfolio Rebalancing with Transaction Costs 10
Yong Fang, K.K. Lai, Shou-Yang Wang

Mining Investment Venture Rules from Insurance Data Based on Decision Tree 20
Jinlan Tian, Suqin Zhang, Lin Zhu, Ben Li

Market-Based Interest Rates: Deterministic Volatility Case 28
Guibin Lu, Qiying Hu

Double Auction in Two-Level Markets 34
Ning Chen, Xiaotie Deng, Hong Zhu

Community Network with Integrated Services 46
ZhiMei Wu, Jun Wang, HuanQiang Zhang

A Set of Data Mining Models to Classify Credit Cardholder Behavior 54
Gang Kou, Yi Peng, Yong Shi, Weixuan Xu

Continuous Time Markov Decision Processes with Expected Discounted Total Rewards 64
Qiying Hu, Jianyong Liu, Wuyi Yue

Core Equivalence in Economy for Modal Logic 74
Takashi Matsuhisa

Model on Analysis of Industrial Relation Based on the Binary Relation Theory 84
Kai-ya Wu, Xiao-jian Chen, Jia-zhong Qian, Ru-zhong Li

Has Chinese Stock Market Become Efficient? Evidence from a New Approach 90
Max Chen, Yongmiao Hong

Workshop on Numerical Methods for Structured Systems

Multi-symplectic Spectral Methods for the Sine-Gordon Equation 101
A.L. Islas, C.M. Schober

A Survey on Methods for Computing Matrix Exponentials in Numerical Schemes for ODEs 111
Nicoletta Del Buono, Luciano Lopez

A Discrete Approach for the Inverse Singular Value Problem in Some Quadratic Group 121
T. Politi

Two Step Runge-Kutta-Nyström Methods for Oscillatory Problems Based on Mixed Polynomials 131
Beatrice Paternoster

A Symplectic Lanczos-Type Algorithm to Compute the Eigenvalues of Positive Definite Hamiltonian Matrices 139
Pierluigi Amodio

Applying Stabilization Techniques to Orthogonal Gradient Flows 149
C. Mastroserio, T. Politi

Workshop on High-Performance Environmental Computations

Coupling General Circulation Models on a Meta-computer 161
Wolfgang Joppich, Johannes Quaas

Numerical Simulation of Cloud Dynamics and Microphysics 171
Elena N. Stankova, Mikhail A. Zatevakhin

Optimal Numerical Realization of the Energy Balance Equation for Wind Wave Models 179
Igor V. Lavrenov

Dynamic Modelling of Environment-Industry Systems 188
Igor Kantardgi

Simulation of Water Exchange in Enclosed Water Bodies 195
Erdal Özhan, Lale Balas

A Baroclinic Three Dimensional Numerical Model Applied to Coastal Lagoons 205
Lale Balas, Erdal Özhan

Stochastic Simulation of Inhomogeneous Metocean Fields. Part I: Annual Variability 213
Alexander V. Boukhanovsky, Harald E. Krogstad, Leonid J. Lopatoukhin, Valentin A. Rozhkov

Stochastic Simulation of Inhomogeneous Metocean Fields. Part II: Synoptic Variability and Rare Events 223
Alexander V. Boukhanovsky, Harald E. Krogstad, Leonid J. Lopatoukhin, Valentin A. Rozhkov, Gerassimos A. Athanassoulis, Christos N. Stephanakos

Stochastic Simulation of Inhomogeneous Metocean Fields. Part III: High-Performance Parallel Algorithms 234
Alexander V. Boukhanovsky, Sergey V. Ivanov

Workshop on Grid Computing for Computational Science

Performance Comparison of Process Allocation Schemes Depending upon Resource Availability on Grid Computing Environment 247
Hiroshi Yamamoto, Kenji Kawahara, Tetsuya Takine, Yuji Oi

Efficient Load Balancing by Adaptive Bypasses for the Migration on the Internet 257
Yukio Hayashi

A Distributed Data Storage Architecture for Event Processing by Using the Globus Grid Toolkit 267
Han Fei, Nuno Almeida, Paulo Trezentos, Jaime E. Villate, Antonio Amorim

Generalization of the Fast Consistency Algorithm to a Grid with Multiple High Demand Zones 275
Jesús Acosta-Elias, Leandro Navarro-Moldes

Performance Analysis of a Parallel Application in the GRID 285
Holger Brunst, Edgar Gabriel, Marc Lange, Matthias S. Müller, Wolfgang E. Nagel, Michael M. Resch

Workshop on Computational Chemistry and Molecular Dynamics

Linear Algebra Computation Benchmarks on a Model Grid Platform 297
Loriano Storchi, Carlo Manuali, Osvaldo Gervasi, Giuseppe Vitillaro, Antonio Laganà, Francesco Tarantelli

Uniform Access to the Distributed Resources for the Computational Chemistry Using UNICORE 307
Jarosław Pytliński, Łukasz Skorwider, Krzysztof Benedyczak, Michał Wroński, Piotr Bała, Valentina Huber

Common Data Format for Program Sharing and Integration 316
Elda Rossi, Andrew Emerson, Stefano Evangelisti

A Multiscale Virtual Reality Approach to Chemical Experiments 324
Antonio Riganelli, Osvaldo Gervasi, Antonio Laganà, Margarita Albertì

Theoretical Analysis on Mechanisms Implied in Hybrid Integrated Circuit Building 331
Giacomo Giorgi, Filippo De Angelis, Nazzareno Re, Antonio Sgamellotti

Parallel Models for a Discrete Variable Wavepacket Propagation 341
D. Bellucci, S. Tasso, Antonio Laganà

Calculation of Displacement Matrix Elements for Morse Oscillators 350
Zimei Rong, Dominique Cavagnat, Laure Lespade

Initial Value Semiclassical Approaches to Reactive and Non Reactive Transition Probabilities 357
N. Faginas Lago, Antonio Laganà

Theoretical Investigations on the Reactions of $C_6H_5^+$ and $C_{10}H_7^+$ with D_2 366
Marco Di Stefano, Marzio Rosi, Antonio Sgamellotti

Density Functional Investigations on the C-C Bond Formation and Cleavage in Molecular Batteries 376
Paola Belanzoni, Marzio Rosi, Antonio Sgamellotti

Violation of Covalent Bonding in Fullerenes 386
E.F. Sheka

Workshop on Recursive and Adaptive Signal/Image Processing (RASIP)

Jointly Performed Computational Tasks in the Multi-mode System Identification 407
Innokenti Semoushin

Fault Point Detection with the Bank of Competitive Kalman Filters 417
Innokenti Semoushin, Julia Tsyganova, Maria V. Kulikova

On Effective Computation of the Logarithm of the Likelihood Ratio Function for Gaussian Signals 427
Maria V. Kulikova

Developing a Simulation Tool Box in MATLAB and Using It for Non-linear Adaptive Filtering Investigation 436
Oleg Gorokhov, Innokenti Semoushin

Comparative Analysis of Stochastic Identification Methods and Fault Diagnosis for Multi-mode Discrete Systems 446
Olga Fatyanova, Alexey Kondratiev

Computational Expenditure Reduction in Pseudo-Gradient Image Parameter Estimation 456
Alexandr Tashlinskii

Estimates Conformity Principle in the Problems of Identification 463
Vladimir Fursov

The ARM-Approach Based Local Modelling of the Gravitational Field .. 471
Sultan Valeev, Konstantin Samokhvalov

Workshop on Numerical Methods for Singular Differential and Differential-Algebraic Equations

The Parameterization Method in Singular Differential-Algebraic Equations 483
Vladimir K. Gorbunov, Igor V. Lutoshkin

Development of the Normal Spline Method for Linear Integro-Differential Equations 492
Vladimir K. Gorbunov, Vyacheslav V. Petrischev, Vyacheslav Y. Sviridov

To Numerical Solution of Singular Perturbed Equations Transformed to the Best Argument 500
E.B. Kuznetsov, S.D. Krasnikov

The Best Parameterization of Initial Value Problem for Mixed Difference-Differential Equation 507
A. Kopylov, E.B. Kuznetsov

Numerical Solution of Differential-Algebraic Equations by Block Methods 516
Michael V. Bulatov

Generalized Jordan Sets in the Theory of Singular Partial Differential-Operator Equations 523
Michael V. Falaleev, Olga A. Romanova, Nicholas A. Sidorov

Invariant Manifolds and Grobman-Hartman Theorem for Equations with Degenerate Operator at the Derivative 533
Bülent Karasözen, Irina Konopleva, Boris Loginov

Poster Papers

Modeling of the Potential Energy Surface of Regrouping Reaction in Collinear Three-Atom Collision System Using Nonlinear Optimization . 545
A.S. Gevorkyan, A.V. Ghulyan, A.R. Barseghyan

Workflow for Simulators Based on Finite Element Method 555
Felix C.G. Santos, Mardoqueu Vieira, Maria Lencastre

Parallel Implementation of the DSMC Method Coupled with a Continuum Solution: Simulation of a Lubrication Problem in Magnetic Disc Storage 565
Sergey Denisikhin, Vladimir Memnonov, Svetlana Zhuravleva

Markowitz-Type Heuristics for Computing Jacobian Matrices Efficiently 575
Andreas Albrecht, Peter Gottschling, Uwe Naumann

Propagation of the Hydraulic Head in an Elastic Pipeline 585
Blanka Filipová, Pavel Nevřiva, Štěpán Ožana

Numerical Revelation of the Molecular Structure for Reaction Effective Stimulator or Inhibitor by the Method of Hamiltonean Systematization of Chemical Reaction System Kinetic Models 593
Levon A. Tavadyan, Gagik A. Martoyan, Seyran H. Minasyan

Numerical Revelation of Kinetic Significance for Steps and Species in Complex Chemical Reaction Mechanisms by Hamiltonian Systematization Method 600
Gagik A. Martoyan, Levon A. Tavadyan

Optimization of Computations in Global Geopotential Field Applications 610
J.A.R. Blais, D.A. Provins

Data Assimilation for 2-D Advection-Dispersion Equations 619
Sergey Kivva

Mathematical Modelling the Ethnic System 629
Victor Korobitsin, Julia Frolova

Simulations for Thermal Analysis of MOSFET IPM Using IMS Substrate 636
Malgorzata Langer, Zbigniew Lisik, Ewa Raj, Nam Kyun Kim, Jan Szmidt

Dynamic Simulations for Thermal Analysis of MOSFET IPM on IMS Substrate 644
Malgorzata Langer, Zbigniew Lisik, Ewa Raj, Nam Kyun Kim, Jan Szmidt

Correlation between Mutation Pressure, Selection Pressure, and Occurrence of Amino Acids 650
Aleksandra Nowicka, Paweł Mackiewicz, Małgorzata Dudkiewicz, Dorota Mackiewicz, Maria Kowalczuk, Stanisław Cebrat, Mirosław R. Dudek

Introducing CEES: Complex Economic Environments Simulator 658
Ismael Rodríguez, Manuel Núñez

Structure of Bilayer Membranes of Gemini Surfactants with Rigid and Flexible Spacers from MD Simulations 668
Dmitry Yakovlev, Edo S. Boek

Algorithms for All-Pairs Reliable Quickest Paths 678
Young-Cheol Bang, Nageswara S.V. Rao, S. Radhakrishnan

The Unified Design Methodology for Application Based on XML Schema 685
Yoon Bang Sung, Mun-Young Choi, Kyung-Soo Joo

Automatic Recognition of Alzheimer's Disease Using Genetic Algorithms and Neural Network 695
Sunyoung Cho, Bo Yeon Kim, Eunhea Park, Yun Seok Chang, Jongwoo Kim, Kyungchun Chung, Weiwan Whang, Hyuntaek Kim

Traffic Characterization of the Web Server Attacks of Worm Viruses 703
Kihun Chong, Ha Yoon Song, Sam H. Noh

An Object-Oriented Software Platform for Examination of Algorithms for Image Processing and Compression 713
Bogusław Cyganek, Jan Borgosz

Combined Detector of Locally-Oriented Structures and Corners in Images Based on a Scale-Space Tensor Representation of Local Neighborhoods of Pixels 721
Bogusław Cyganek

Telecommunication Jitter Measurement in the Transmultipexer Systems Method Based on the Wavelet Pattern Correlation 731
Jan Borgosz, Bogusław Cyganek

Self-Organizing Compact Modeling Methodology for High-Speed Passive Electrical Interconnection Structures 741
Tom Dhaene

Specification and Automated Recognition of Algorithmic Concepts with ALCOR 748
Beniamino Di Martino, Anna Bonifacio

Modelling of Complex Systems Given as a Mealy Machine with Linear Decision Diagrams 758
P. Dziurzanski

Empirical Evaluation of the Difficulty of Finding a Good Value of k for the Nearest Neighbor 766
Francisco J. Ferrer-Troyano, Jesús S. Aguilar-Ruiz, José C. Riquelme

Replicated Ambient Petri Nets 774
David de Frutos Escrig, Olga Marroquín Alonso

Neural Networks for Event Detection from Time Series: A BP Algorithm Approach 784
Dayong Gao, Y. Kinouchi, K. Ito

Data Loss Reparation Due to Indeterminate Fine-Grained Parallel Computation 794
Ekaterina O. Gorbunova, Yuri V. Kondratenko, Michael G. Sadovsky

Measurement Models for Survivability and Competitiveness of Very Large E-marketplace 802
Jingzhi Guo, Chengzheng Sun

Embedded Fuzzy Control System: Application to an Electromechanical System 812
R.E. Haber, J.R. Alique, A. Alique, J.E. Jiménez

A Dynamically Grouped Multi-multicast Stream Scheduling Strategy for Video-on-Demand Systems 822
Dafu Deng, Hai Jin, Zongfen Han

Multilevel System as Multigraph 832
Waldemar Korczyński, José de Jesús Cruz Guzmán, Zbigniew Oziewicz

Fast Exponentiaion over GF(2^m) Based on Cellular Automata 841
Kyo-Min Ku, Kyeoung-Ju Ha, Kee-Young Yoo

Interacting Automata for Modelling Distributed Systems 851
Irina A. Lomazova

The Reachability Problem in a Concave Region: A New Context 861
Ali Mohades, Mohammad Ebrahim Shiri, Mohammadreza Razzazi

Generalized Coordinates for Cellular Automata Grids 869
Lev Naumov

On Generating Random Network Structures: Trees 879
Alexey S. Rodionov, Hyunseung Choo

Sensitivities for a Single Drop Simulation 888
Christian H. Bischof, H. Martin Bücker, Arno Rasch, Emil Slusanschi

A Simple Model of Drive with Friction for Control System Simulation . . . 897
Adam Woźniak

Research on Fuzzy Inventory Control under Supply Chain Management Environment 907
Guangyu Xiong and Hannu Koivisto

Independent Zone Setup Scheme for Re-configurable Wireless Network . . 917
Jae-Pil Yoo, Kee-cheon Kim, SunYoung Han

Finding Synchronization-Free Parallelism for Non-uniform Loops 925
Volodymyr Beletskyy

A Transformation to Provide Deadlock–Free Programs 935
Pablo Boronat, Vicente Cholvi

Building the Program Parallelization System Based on a Very Wide Spectrum Program Transformation System 945
Alexander Alexeyevich Bukatov

Performance Evaluation of the Striped Checkpointing Algorithm on the Distributed RAID for Cluster Computer 955
Yun Seok Chang, Sun Young Cho, Bo Yeon Kim

An Evaluation of Globus and Legion Software Environments 963
M.A.R. Dantas, J.N.C. Allemand, L.B.C. Passos

An Agent Model for Managing Distributed Software Resources in Grid Environment 971
Jingbo Ding, Weiqin Tong

Parallel DNA Sequence Alignment Using a DSM System in a Cluster of Workstations 981
Renata Cristina Faray Melo, Maria Emília Telles Walter, Alba Cristina Magalhaes Alves de Melo, Rodolfo B. Batista

CSA&S/PV: Parallel Framework for Complex Systems Simulation 991
Ewa Niewiadomska-Szynkiewicz, Maciej Żmuda

A Parallel Framework for Computational Science 1002
Fernando Rubio, Ismael Rodríguez

Application Controlled IPC Synchrony – An Event Driven Multithreaded Approach 1012
Susmit Bagchi, Mads Nygaard

ToCL: A Thread Oriented Communication Library to Interface VIA and GM Protocols 1022
Albano Alves, António Pina, José Exposto, José Rufino

A Multi Dimensional Visualization and Analysis Toolkit for Astrophysics 1032
Daniela Ferro, Vincenzo Antonuccio-Delogu, Ugo Becciani, Angela Germaná, Claudio Gheller, Maura Melotti

Error Correcting Codes with Mathematica 1042
Igor Gashkov

Mobile Work Environment for Grid Users. Grid Applications' Framework 1052
Michal Kosiedowski, Miroslaw Kupczyk, Rafal Lichwala, Norbert Meyer, Bartek Palak, Marcin Plóciennik, Pawel Wolniewicz, Stefano Beco

EC Transactions Use Different Web-Based Platforms 1059
Whe Dar Lin

MOIRAE – An Innovative Component Architecture with Distributed Control Features 1069
Katia Leal, José Herrera, José M. Peña, Ernestina Menasalvas

Applying Computational Science Techniques to Support Adaptive Learning 1079
Juan M. Santos, Luis Anido, Martín Llamas, Luis M. Álvarez, Fernando A. Mikic

The Use of the Cooperative Solver SibCalc in Modeling of Complex Problems 1088
Tamara Kashevarova, Alexander Semenov

Computational and Soft Skills Development through the Project Based Learning 1098
Innokenti Semoushin, Julia Tsyganova, Vladimir Ugarov

XML-Based Interface Model for Socially Adaptive Web-Based Systems User Interfaces 1107
Janusz Sobecki

Author Index 1117

Table of Contents, Part III

Australian Track

Track on Applications

A Bayes Algorithm for the Multitask Pattern Recognition Problem – Direct Approach 3
Edward Puchala

The Development of a Virtual Reality Environment to Model the Experience of Schizophrenia 11
Jennifer Tichon, Jasmine Banks, Peter Yellowlees

An Optimization-Based Approach to Patient Grouping for Acute Healthcare in Australia 20
A.M. Bagirov, L. Churilov

Dynamic Parameterization to Simulate DIN Export Due to Gypsy Moth Defoliation 30
Ping Wang, Lewis C. Linker, Keith N. Eshleman

Multi-model Simulations of Chicken Limb Morphogenesis 39
R. Chaturvedi, J.A. Izaguirre, C. Huang, T. Cickovski, P. Virtue, G. Thomas, G. Forgacs, M. Alber, G. Hentschel, S.A. Newman, J.A. Glazier

ProtoMol: A Molecular Dynamics Research Framework for Algorithmic Development 50
T. Matthey, A. Ko, J.A. Izaguirre

An Efficient Navigation Method for Virtual Endoscopy Using Volume Ray Casting 60
Byeong-Seok Shin, Suk Hyun Lim

Constructing a Near-Minimal-Volume Computational Box for Molecular Dynamics Simulations with Periodic Boundary Conditions 70
Henk Bekker, Jur P. van den Berg, Tsjerk A. Wassenaar

Recast of the Outputs of a Deterministic Model to Get a Better Estimate of Water Quality for Decision Makings 80
Ping Wang, Lewis C. Linker, Carl Cerco, Gary Shenk, Richard Batiuk

A Fuzzy Approach for Overcurrent Relays Simulation 90
Hossein Askarian Abyaneh, Hossein Kazemi Karegar, Majid Al-Dabbagh

A Knowledge-Based Technique for Constraints Satisfaction in Manpower Allocation 100
Khaireel A. Mohamed, Amitava Datta, Ryszard Kozera

A Symbolic Approach to Vagueness Management 109
Mazen El-Sayed, Daniel Pacholczyk

Development of Multiple Job Execution and Visualization System on ITBL System Infrastructure Software and Its Utilization for Parametric Studies in Environmental Modeling 120
Yoshio Suzuki, Nobuko Matsumoto, Nobuhiro Yamagishi, Kenji Higuchi, Takayuki Otani, Haruyasu Nagai, Hiroaki Terada, Akiko Furuno, Masamichi Chino, Takuya Kobayashi

A Genetic Algorithm for Predicting RNA Pseudoknot Structures 130
Dongkyu Lee, Kyungsook Han

Computational Approach to Structural Analysis of Protein-RNA Complexes 140
Namshik Han, Hyunwoo Kim, Kyungsook Han

Improved Web Searching through Neural Network Based Index Generation 151
Xiaozhe Wang, Damminda Alahakoon, Kate A. Smith

Neural Network for Modeling Nonlinear Time Series: A New Approach .. 159
Chokri Slim, Abdelwahed Trabelsi

Export Behaviour Modeling Using EvoNF Approach 169
Ron Edwards, Ajith Abraham, Sonja Petrovic-Lazarevic

Simulation Studies of a Multi-priority Dual Queue (MPDQ) with Preemptive and Non-preemptive Scheduling 179
Anthony Bedford, Panlop Zeephongsekul

Visualization of Protein-Protein Interaction Networks Using Force-Directed Layout 190
Yanga Byun, Kyungsook Han

Track on Clusters and Grids

Counting Polyominoes: A Parallel Implementation for Cluster Computing 203
Iwan Jensen

Hyper-BLAST: A Parallelized BLAST on Cluster System 213
Hong-Soog Kim, Hae-Jin Kim, Dong-Soo Han

Parallel Superposition for Bulk Synchronous Parallel ML 223
Frédéric Loulergue

visPerf: Monitoring Tool for Grid Computing 233
DongWoo Lee, Jack J. Dongarra, R.S. Ramakrishna

Design and Implementation of Intelligent Scheduler for Gaussian Portal on Quantum Chemistry Grid 244
Takeshi Nishikawa, Umpei Nagashima, Satoshi Sekiguchi

Extensions to Web Service Techniques for Integrating Jini into a Service-Oriented Architecture for the Grid 254
Yan Huang, David W. Walker

Multiple-Level Grid Algorithm for Getting 2D Road Map in 3D Virtual Scene 264
Jiangchun Wang, Shensheng Zhang, Jianqiang Luo

Parallelisation of Nonequilibrium Molecular Dynamics Code for Polymer Melts Using OpenMP 275
Zhongwu Zhou, B.D. Todd, Peter J. Daivis

A Fault Tolerance Service for QoS in Grid Computing 286
Hwa Min Lee, Kwang Sik Chung, Sung Ho Jin, Dae-Won Lee, Won Gyu Lee, Soon Young Jung, Heon Chang Yu

Performing Grid Computation with Enhanced Web Service and Service Invocation Technologies 297
Gang Xue, Graeme E. Pound, Simon J. Cox

Software Architecture and Performance Comparison of MPI/Pro and MPICH 307
Rossen Dimitrov, Anthony Skjellum

Virtual and Dynamic Hierarchical Architecture for E-science Grid 316
Lican Huang, Zhaohui Wu, Yunhe Pan

Track on Models and Algorithms

Algorithmic Entropy, Phase Transition, and Smart Systems 333
E.V. Krishnamurthy

Computations of Coupled Electronic States in Quantum Dot/Wetting Layer Cylindrical Structures 343
Roderick V.N. Melnik, Kate N. Zotsenko

Deadlock Free Specification Based on Local Process Properties 350
D.P. Simpson, J.S. Reeve

On the Reconfiguration Algorithm for Fault-Tolerant VLSI Arrays 360
Jigang Wu, Srikanthan Thambipillai

Automated Generation of Kinetic Chemical Mechanisms Using Rewriting 367
Olivier Bournez, Guy-Marie Côme, Valérie Conraud, Hélène Kirchner, Liliana Ibănescu

Contraction versus Relaxation: A Comparison of Two Approaches for the Negative Cost Cycle Detection Problem 377
K. Subramani, L. Kovalchick

Direct Method for Solving a Transmission Problem with a Discontinuous Coefficient and the Dirac Distribution 388
Hideyuki Koshigoe

Track on Web Engineering

CORBA Based Real-Time Object-Group Platform in Distributed Computing Environments 401
Su-Chong Joo, Sung-Kwun Oh, Chang-Sun Shin, Jun Hwang

Structural Similarity between XML Documents and DTDs 412
Patrick K.L. Ng, Vincent T.Y. Ng

Web Personalisation with the Cover Coefficient Algorithm 422
Matthew Anderson, Irfan Altas, Geoff Fellows

Infrastructure of an Agile Supply Chain System: A Virtual Agent Modeling Approach 432
H.C.W. Lau, Christina W.Y. Wong

Track on Networking

Performance Improvement of Deflection Routing in Optical Burst Switching Networks 445
Sukyoung Lee, Kotikalapudi Sriram, HyunSook Kim, JooSeok Song

A Performance Evaluation on Qos-Supporting Multicast Protocol over Conventional Multicast Protocol 453
Won-Hyuck Choi, Tae-Seung Lee, Jung-Sun Kim

Improving the Performance of Multistage Interconnection Networks under Nonuniform Traffic Pattern on Shorter Cycles 463
Hyunseung Choo, Youngsong Mun

Symmetric Tree Replication Protocol for Efficient Distributed Storage System 474
Sung Chune Choi, Hee Yong Youn, Joong Sup Choi

A Group Key Management Supporting Individual Batch Rekeying and Fault Tolerance for Secure Multicast 485
Hojae Hyun, Keechon Kim, Sunyoung Han

Architecture for Internal Communication in Multi-gigabit IP Routers 495
Young-Cheol Bang, W.B. Lee, Hyunseung Choo, N.S.V. Rao

Low-Cost Fault-Tolerance Protocol for Large-Scale Network Monitoring 504
JinHo Ahn, SungGi Min, YoungIl Choi, ByungSun Lee

Telecommunication Network Modeling and Planning Tool on ASP Clusters 514
P. Bacquet, O. Brun, J.M. Garcia, T. Monteil, P. Pascal, S. Richard

Track on Parallel Methods and Systems

Fault-Tolerant Routing in Mesh-Connected 2D Tori 527
Eunseuk Oh, Jong-Seok Kim, Hyeong-Ok Lee

Network-Tree Model and Shortest Path Algorithm 537
Guozhen Tan, Xiaojun Han, Wen Gao

Parallel Genetic Algorithm for a Flow-Shop Problem with Multiprocessor Tasks 548
C. Oguz, Yu-Fai Fung, M. Fikret Ercan, X.T. Qi

Performance Variability of Highly Parallel Architectures 560
William T.C. Kramer, Clint Ryan

JDOS: A Jini Based Distributed Operating System 570
M. Saywell, J.S. Reeve

Parallel Blocked Sparse Matrix-Vector Multiplication with Dynamic Parameter Selection Method 581
Makoto Kudo, Hisayasu Kuroda, Yasumasa Kanada

Parallelization of the Discrete Gradient Method of Non-smooth Optimization and Its Applications 592
G. Beliakov, J.E. Monsalve Tobon, A.M. Bagirov

A Parallel Prime Edge-Length Crystallographic FFT 602
Jaime Seguel, Daniel Burbano

A Service-Oriented Framework for Parallel Medical Image Reconstruction 612
S. Benkner, A. Dimitrov, G. Engelbrecht, R. Schmidt, N. Terziev

Parallel Implicit Solution of Full Navier-Stokes Equations 622
V.Ü. Ünal, Ü. Gülçat

Track on Data Mining

Application of the Confidence Measure in Knowledge Acquisition Process 635
Michal Wozniak

Computational Science and Data Mining 644
Flaviu Adrian Mărginean

Data Sharing Model for Sequence Alignment to Reduce Database Retrieve 652
Min Jun Kim, Jai-Hoon Kim, Jin-Won Jung, Weontae Lee

Workshop on Parallel Linear Algebra (WoPLA03)

Self-Adapting Software for Numerical Linear Algebra Library Routines on Clusters 665
Zizhong Chen, Jack J. Dongarra, Piotr Luszczek, Kenneth Roche

V-Invariant Methods for Generalised Least Squares Problems 673
M.R. Osborne

Parallelisation of Sparse Grids for Large Scale Data Analysis 683
Jochen Garcke, Markus Hegland, Ole Nielsen

An Augmented Lanczos Algorithm for the Efficient Computation of a Dot-Product of a Function of a Large Sparse Symmetric Matrix 693
Roger B. Sidje, Kevin Burrage, B. Philippe

Memory Hierarchy Optimizations and Performance Bounds for Sparse A^TAx 705
Richard Vuduc, Attila Gyulassy, James W. Demmel, Katherine A. Yelick

Issues in the Design of Scalable Out-of-Core Dense Symmetric Indefinite Factorization Algorithms 715
Peter E. Strazdins

Application of the Multi-level Parallelism (MLP) Software to a Finite Element Groundwater Program Using Iterative Solvers with Comparison to MPI 725
Fred Tracy

Workshop on Java in Computational Science

Visual Parameteric Modeler for Rapid Composition of Parameter-Sweep Applications for Processing on Global Grids 739
Shoaib Burq, Steve Melnikoff, Kim Branson, Rajkumar Buyya

Method Call Acceleration in Embedded Java Virtual Machines 750
M. Debbabi, M. Erhioui, L. Ketari, N. Tawbi, H. Yahyaoui, S. Zhioua

Jade: A Parallel Message-Driven Java 760
Jayant DeSouza, Laxmikant V. Kalé

Workshop on Computational Earthquake Physics and Solid Earth System Simulation

Parallel Finite Element Analysis Platform for the Earth Simulator: GeoFEM 773
Hiroshi Okuda, Kengo Nakajima, Mikio Iizuka, Li Chen, Hisashi Nakamura

Mantle Convection Modeling with Viscoelastic/Brittle Lithosphere: Numerical and Computational Methodology 781
Louis Moresi, David May, Justin Freeman, Bill Appelbe

Anisotropic Convection Model for the Earth's Mantle 788
Hans-B. Mühlhaus, M. Čada, Louis Moresi

Finite Element Simulation of Stress Evolution in a Frictional Contact System 798
H.L. Xing, Peter Mora, A. Makinouchi

Transparent Boundary Conditions for Wave Propagation on Unbounded Domains 807
Dorin-Cezar Ionescu, Heiner Igel

A 2D Numerical Model for Simulating the Physics of Fault Systems 817
Peter Mora, Dion Weatherley

Strategies for the Detection and Analysis of Space-Time Patterns of Earthquakes on Complex Fault Systems 827
John B. Rundle, William Klein, Kristy Tiampo, Andrea Donnellan, Geoffrey Fox

Texture Alignment in Simple Shear 837
Frédéric Dufour, Hans-B. Mühlhaus, Louis Moresi

Mechanical Properties of the Earth's Crust with Self-Similar Distribution of Faults 845
Arcady V. Dyskin

Performance Modeling Codes for the QuakeSim Problem Solving Environment 855
Jay Parker, Andrea Donnellan, Gregory Lyzenga, John B. Rundle, Terry Tullis

Interacting Data Services for Distributed Earthquake Modeling 863
Marlon Pierce, Choonhan Youn, Geoffrey Fox

Apparent Strain Localization and Shear Wave Dispersion in Elastic Fault Gouge with Microrotations 873
E. Pasternak, Hans-B. Mühlhaus, Arcady V. Dyskin

Efficient Implementation of Complex Particle Shapes in the Lattice Solid Model 883
Steffen Abe, Peter Mora

A Method of Hidden Markov Model Optimization for Use with Geophysical Data Sets 892
Robert A. Granat

Workshop on Performance Evaluation, Modeling, and Analysis of Scientific Applications on Large-Scale Systems

A Performance Model of Non-deterministic Particle Transport on Large-Scale Systems 905
Mark M. Mathis, Darren J. Kerbyson, Adolfy Hoisie

A Compiler Approach to Performance Prediction Using Empirical-Based Modeling 916
Pedro C. Diniz

A Performance Prediction Framework for Scientific Applications 926
Laura Carrington, Allan Snavely, Xiaofeng Gao, Nicole Wolter

Identification of Performance Characteristics from Multi-view Trace Analysis 936
Daniel Spooner, Darren Kerbyson

Compiler Directed Parallelization of Loops in Scale for Shared-Memory Multiprocessors 946
Gregory S. Johnson, Simha Sethumadhavan

A New Data Compression Technique for Event Based Program Traces . . . 956
Andreas Knüpfer

Exploiting Stability to Reduce Time-Space Cost for Memory Tracing 966
Xiaofeng Gao, Allan Snavely

Workshop on Scientific Visualization and Human-Machine Interaction in a Problem Solving Environment

Oh Behave! Agents-Based Behavioral Representations in Problem Solving Environments 979
M. North, C. Macal, P. Campbell

JBeanStudio: A Component-Oriented Visual Software Authoring System for a Problem Solving Environment – Supporting Exploratory Visualization – 985
Masahiro Takatsuka

Multi-agent Approach for Visualisation of Fuzzy Systems 995
Binh Pham, Ross Brown

Towards Biomedical Problem Solving in a Game Environment 1005
Yang Cai, Ingo Snel, B. Suman Bharathi, Clementine Klein, Judith Klein-Seetharaman

Learning Surgical Interventions by Navigating in Virtual Reality Case Spaces 1015
Piet Kommers, Steffan Rödel, Jan-Maarten Luursema, Bob Geelkerken, Eelco Kunst

Virtual Reality and Desktop as a Combined Interaction-Visualisation Medium for a Problem-Solving Environment . . . 1025
E.V. Zudilova, Peter M.A. Sloot

Online Trajectory Classification 1035
Corina Sas, Gregory O'Hare, Ronan Reilly

Trajectory Mapping for Landmine Detection Training 1045
Yang Cai

A Low-Cost Model Acquisition System for Computer Graphics Applications 1054
Minh Tran, Amitava Datta, Nick Lowe

How Many Pixels Do We Need to See Things? 1064
Yang Cai

Bio-Feedback Based Simulator for Mission Critical Training 1074
Igor Balk

Workshop on Innovative Solutions for Grid Computing

Proposing and Evaluating Allocation Algorithms in a Grid Environment . 1083
Salvatore Cavalieri, Salvatore Monforte, Fabio Scibilia

Hierarchical Addressing and Routing Mechanisms for Distributed Applications over Heterogeneous Networks 1093
Damien Magoni

Active Network Approach to Grid Management 1103
Alex Galis, Jean-Patrick Gelas, Laurent Lefèvre, Kun Yang

The PAGIS Grid Application Environment 1113
Darren Webb, Andrew L. Wendelborn

Visual Modeler for Grid Modeling and Simulation (GridSim) Toolkit 1123
Anthony Sulistio, Chee Shin Yeo, Rajkumar Buyya

Layered Resource Representation in Grid Environment: An Example from VEGA Grid 1133
Fangpeng Dong, Yili Gong, Wei Li, Zhiwei Xu

TCM-Grid: Weaving a Medical Grid for Traditional Chinese Medicine . . . 1143
Huajun Chen, Zhaohui Wu, Chang Huang, Jiefeng Xu

Author Index 1153

A Model for Predicting the Optimum Checkpoint Interval for Restart Dumps

John Daly

Raytheon Intelligence and Information Systems, Garland, TX

Abstract. As the run time of an application approaches the the mean time to interrupt (MTTI) for the system on which it is running, it becomes necessary to generate intermediate snapshots of the application's run state, known as checkpoint files or restart dumps. In the event of a system failure that halts program execution, these snapshots allow an application to resume computing from the most recently saved intermediate state instead of starting over at the beginning of the calculation. In this paper three models for predicting the optimum compute intervals between restart dumps are discussed. These models are evaluated by comparing their results to a simulation that emulate an application running on a actual system with interrupts. The results will be used to derive a simple method for calculating the optimum restart interval.

1 Introduction

When running on a system where the frequency of interrupts is low compared to the runtime of a particular application, the total wall clock time accumulated while running that application will simply be the computational solution time. However, if we can reasonably expect one or more interrupts to occur during the runtime of an application then we must adopt a strategy such as checkpoint restart that allows us to write out an image of the current state of the calculation at pre-determined intervals. Qualitatively we can see that checkpointing too frequently is disadvantageous, because writing a dump file can be time consuming, but not writing often enough might be equally perilous, because all the work since the last checkpoint will be lost in the event of an interrupt. In this study we will quantify the optimum restart interval that minimizes the total application run time, and express the result in terms of a simple analytic approximation.

One strategy for optimizing the compute interval between dumps τ is to generate a cost function $T_w(\tau)$, the total wall clock time to complete the execution of an application, and find its minima. Quantitatively speaking

$$T_w(\tau) = \mathit{Solve\ Time} + \mathit{Dump\ Time} + \mathit{Rework\ Time} + \mathit{Restart\ Time}\ . \quad (1)$$

Solve time is defined as time spent doing actual computational cycles working towards a final solution. For a system with no interrupts the wall clock time $T_w(\tau)$ consists entirely of computation time. Dump time is overhead spent writing out the checkpoint files required to restart the application after an interrupt.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 3–12, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Rework time is the amount of wall clock time lost when an application is killed by an interrupt prior to completing a restart dump. It is the amount of time elapsed since the last restart dump was successfully written. Restart time is the time required before an application is able to resume real computational work. It includes all overhead costs associated with restarting a calculation after an interrupt.

2 The Models

2.1 A First Order Model

John W. Young [8] proposed $\tau_{opt} = \sqrt{2\delta M}$ as a useful first order approximation of the optimum checkpoint interval, where δ is the time to write a checkpoint file, M is the mean time between system failures, and τ_{opt} is the optimum compute time between writing checkpoint files. We will start with a derivation similar to Young's estimate before moving on to deriving more accurate models. To help us consider how different terms contribute to the total wall clock time, and how the wall clock time relates to the solve time, consider figure 1 which provides a conceptual view of an application run encountering a single interrupt.

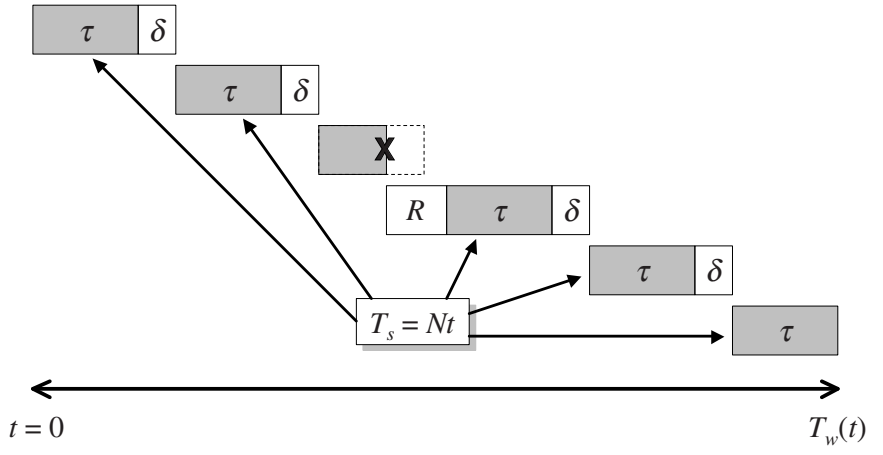


Fig. 1. The application time line broken up into five passed compute segments and one failed compute segment designated by on **X**. Passed segments are of three types: solve-dump, restart-solve-dump, or solve-to-end. An application run is complete when the accumulated computation time τ of all the passed segments is equal to the total solution time T_s for the application.

Using figure 1 it is a straightforward to construct the cost function for total wall clock time. Solve time will be $\tau n(\tau)$ where $n(\tau)$ is the number of passed segments required to complete a calculation. Dump time will be $\delta(n(\tau) - 1)$

where one is subtracted because there is no dump on the last segment. For rework time we make the simplifying assumption that an interrupt is equally likely to occur at any time during a compute segment. This implies that over a large number of failures the time spent in rework is going to be approximately half the segment length times the number of interrupts. In other words, rework time will be described by $\frac{1}{2}(\tau + \delta)N(\tau)$ where $N(\tau)$ is the expected number of interrupts over the course of the calculation. Finally, the total restart time is simply $RN(\tau)$, the amount of time required to restart times the total number of failures. Combining these terms we construct our basic cost function as

$$T_w(\tau) = \tau n(\tau) + (n(\tau) - 1)\,\delta + \frac{1}{2}\,(\tau + \delta)\,N(\tau) + RN(\tau)\ . \tag{2}$$

Next we determine how to express the number of passed segments $n(\tau)$ and the expected number of restarts $N(\tau)$ as a function of the compute time per segment. As we see from figure 1, the number of passed segments required to complete a jobs is just the solve time T_s for the job, which is fixed, divided by τ. The expected number of restarts, assuming the number of interrupts is in some sense statistically significant, will be the product of the number of passed segments required to complete the calculation and the probability of each segment failing. Assuming interrupts arrive according to a Poisson process [1], and approximating the exponential term by a first order series expansion, we get

$$N(\tau) = \frac{T_s}{\tau}\left(e^{\frac{\tau+\delta}{M}} - 1\right) \cong \frac{T_s}{\tau}\left(\frac{\tau+\delta}{M}\right) \quad \text{for} \quad \frac{\tau+\delta}{M} \ll 1\ . \tag{3}$$

We made the simplifying assumption that the contribution of the probability of failure in a restart-solve-dump or a solve-to-end segment to the total number of failures is inconsequential compared to the probability of a failure in a solve-dump segment. We will need to revisit this assumption in our complete model.

Substituting the terms for $n(\tau)$ and $N(\tau)$ into our cost function gives

$$T_w(\tau) = T_s + \left(\frac{T_s}{\tau} - 1\right)\delta + \left[\frac{1}{2}\,(\tau + \delta) + R\right]\frac{T_s}{\tau}\left(\frac{\tau+\delta}{M}\right)\ . \tag{4}$$

Equation 4 will our cost function for the first order model. We are interested in finding minima for values of $\tau > 0$. To do this we consider solutions of the first derivative with respect to τ equal to zero.

$$-\frac{\delta T_s}{\tau^2} + \frac{T_s}{M}\left(\frac{1}{2} - \frac{\delta^2}{2\tau^2} - \frac{\delta R}{\tau^2}\right) = -\frac{1}{\tau^2}\left(2\delta M - 2\delta R + \delta^2\right) + 1 = 0\ . \tag{5}$$

Thus the minimization problem reduces to a simple quadratic form. Assuming that the delta squared term in negligible, the assumption we made when we expanded the exponential failure term in equation 3, we recover Young's original solution with an added term for the restart overhead.

$$\tau = \sqrt{2\delta(M + R)} \quad \text{for} \quad \delta \ll M\ . \tag{6}$$

[1] See [4] and others.

2.2 A Modified Model

The biggest limitation of the linear model is its poor predictive ability for small values of M. The decision to use the series expansion of the exponential term in equation 3 was justified by assuming that the elapsed time between restarts is substantially less than the mean time between failures for the system. However, as computing moves toward terascale systems, a multiplication of the numbers of system components is resulting in a proportional decrease in the overall reliability of the system. [2] In general, we cannot assume that $(\tau + \delta)/M$ will be negligible. With this in mind, let us rewrite equation 4 retaining the exponential term.

$$T_w(\tau) = T_s + \left(\frac{T_s}{\tau} - 1\right)\delta + \left[\frac{1}{2}(\tau + \delta) + R\right]\frac{T_s}{\tau}\left(e^{\frac{\tau+\delta}{M}} - 1\right) \ . \tag{7}$$

Using equation 7 as our new cost function, we will find the minima with respect τ once again setting the derivative to zero.

$$e^{\frac{\tau+\delta}{M}}\left[\tau^2 + (\delta + 2R)\tau - (\delta + 2R)M\right] + 2RM - \delta M = 0 \ . \tag{8}$$

Instead of expanding the exponential term, recast equation 8 as follows

$$\frac{\tau + \delta}{M} = \ln\left[\frac{(\delta - 2R)\,M}{\tau^2 + (\delta + 2R)\tau - (\delta + 2R)M}\right] = \ln\left[g(\tau)\right] \ . \tag{9}$$

We know that $(\tau + \delta)/M$ is never negative, which means that $g(\tau) > 1$ for all τ. So we will try a series expansion for the natural logarithm of $g(\tau)$ as follows:

$$\begin{aligned}\frac{\tau + \delta}{M} &= \frac{g(\tau) - 1}{g(\tau)} + \frac{1}{2}\left(\frac{g(\tau) - 1}{g(\tau)}\right)^2 + \frac{1}{3}\left(\frac{g(\tau) - 1}{g(\tau)}\right)^3 \cdots \\ &= \left(1 - \frac{1}{g(\tau)}\right) + \frac{1}{2}\left(1 - \frac{1}{g(\tau)}\right)^2 + \frac{1}{3}\left(1 - \frac{1}{g(\tau)}\right)^3 \cdots \end{aligned} \ . \tag{10}$$

Since $g(\tau) > 1$, as we noted in equation 9, it follows that $0 < 1/g(\tau) < 1$ and therefore $(1 - 1/g(\tau)) < 1$. Thus, the series expansion is expected to converge, albeit slowly for large values of $g(\tau)$. Ignoring higher order terms once again, we are left with a simple relationship, which can be reduced to a quadratic form.

$$\begin{gathered}\frac{\tau + \delta}{M} = \left(1 - \frac{\tau^2 + (\delta + 2R)\tau - (\delta + 2R)M}{(\delta + 2R)M}\right) \\ \Downarrow \\ \tau^2 + 2\delta\tau - \left(\delta^2 - 2\delta(M + R)\right) = 0 \ .\end{gathered} \tag{11}$$

Therefore, the value of τ which minimizes equation 7 is approximately

$$\tau = \sqrt{2\delta(M + R)} - \delta \ . \tag{12}$$

[2] For example, preliminary data for the first 10 TeraOP/s segment of the 30 TeraOP/s ASCI Q machine at Los Alamos National Laboratory indicates $M \cong 6.7$ hours.

2.3 A Complete Model

With both first order and modified models now derived we will reconsider two of our assumptions which turn out not to be very accurate for small M. The first assumption that will cause us difficulty was made in equation 2 where the fraction of a segment requiring rework is approximated as one-half. In fact, that was a reasonable approximation for large M, but as M approaches $\tau + \delta$ the fraction of rework drops off rather precipitously as depicted in figure 2. For very small M, the expected failure time is going to actually be shorter than the solve-dump segment itself, meaning that the beginning of the segment will see far more failures than the end of the segment.

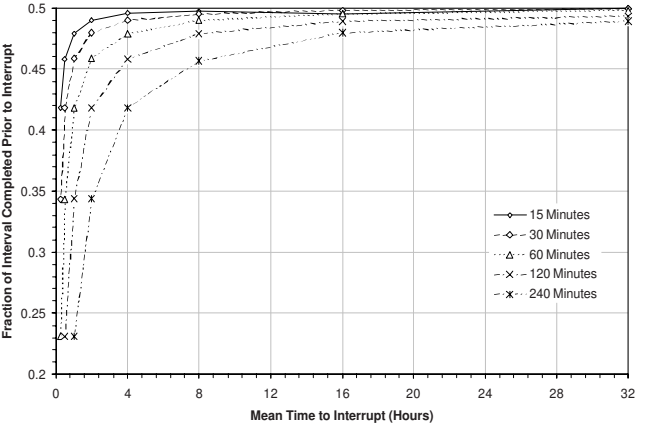


Fig. 2. The fractional amount of rework required following a system interrupt for various compute intervals is reported as the mean of multiple simulations for a given compute interval. Each function represents a different interval between restart dumps, ranging from 15 minutes to 4 hours.

To better grasp of how these expected failures are behaving, consider that the probability of a failure occurring halfway through any arbitrary compute segment is actually the sum of the probabilities of the failure occurring halfway through the first segment plus the probability of it occurring halfway through the second interval and so forth. In other words, the distribution function describing the probability of failure at a time t in any arbitrary compute segment will be

$$f(t) = \frac{1}{M}e^{-\frac{t}{M}} + \frac{1}{M}e^{-\frac{t+\Delta t}{M}} + \frac{1}{M}e^{-\frac{t+2\Delta t}{M}} + \cdots = \frac{e^{-\frac{t}{M}}}{M\left(1 - e^{-\frac{\Delta t}{M}}\right)} \ . \qquad (13)$$

Therefore, the expected point of failure for a random variable T in the range $0 \le T \le \Delta t$ in terms of the probability density function $f(t)$ will be given by

$$\begin{aligned} E(T) &= \int_0^{\Delta t} t f(t)\, dt \\ &= \frac{\int_0^{\Delta t} te^{-\frac{t}{M}}\, dt}{M\left(1-e^{-\frac{\Delta t}{M}}\right)} = \frac{\left(-Mte^{-\frac{t}{M}} - M^2 e^{-\frac{t}{M}}\right)\Big|_0^{\Delta t}}{M\left(1-e^{-\frac{\Delta t}{M}}\right)} = \frac{Me^{\frac{\Delta t}{M}} - M - \Delta t}{e^{\frac{\Delta t}{M}} - 1} \\ &= M + \frac{\Delta t}{1 - e^{\frac{\Delta t}{M}}} \ . \end{aligned} \tag{14}$$

So, instead of $\frac{1}{2}$, the expected fraction of rework $\Phi(\Delta t)$ over a time interval Δt will actually be $E(T)$ derived in equation 14 divided by the length of the interval

$$\phi(\Delta t) = \frac{M}{\Delta t} + \frac{1}{1 - e^{\frac{\Delta t}{M}}} \ . \tag{15}$$

The second difficulty with our simplified models is that the segment size for a failure is always assumed to be $\tau + \delta$, which means a failure never occurs in a restart-solve-dump segment. Failures occurring in a restart-solve-dump segment should be represented by a segment length of $R + \tau + \delta$ in the model. If we complete the restart-solve-dump segment successfully then the contribution of the interrupt to the solution wall clock time is the restart time plus the expected rework time for a solve-dump segment. Otherwise the contribution to wall clock time is the expected rework time for the restart-solve-dump segment.

Previously, in equation 2, we estimated the contribution of restart and rework to our cost function as

$$\frac{1}{2}\left(\tau + \delta\right) N(\tau) + RN(\tau) \ . \tag{16}$$

Based on the correction for the fraction of rework in equation 15, the distinction between failed solve-dump segments and restart-solve-dump segments can be incorporated into equation 16 as

$$[\phi_1(\tau)(\tau + \delta)N(\tau) + RN(\tau)]P(\tau) + \phi_2(\tau)(R + \tau + \delta)(1 - P(\tau))N(\tau) \ . \tag{17}$$

where $\phi_1(\tau)$ is the fractional rework associated with a solve-dump segment, $\phi_2(\tau)$ is the fractional rework associated with a restart-solve-dump segment, and $P(\tau)$ is the probability of successfully completing the restart-solve-dump segment immediately following an interrupt.

Furthermore, in order to allow for the possibility of multiple restarts in a single compute segment we must redefine $N(\tau)$. Instead of estimating the total number of failures by the number of compute segments divided by the probability of a failure in each segment, we will now use the total wall clock time divided by the mean time between failures. So after we replacing $N(\tau)$ by $T_w(\tau)/M$ our complete model in implicit form becomes

$$\begin{aligned} T_w(\tau) = T_s + \left(\frac{T_s}{\tau} - 1\right)\delta + \\ \{[\phi_1(\tau)(\tau + \delta) + R]P(\tau) + \phi_2(\tau)(R + \tau + \delta)(1 - P(\tau))\}\, \frac{T_w(\tau)}{M} \end{aligned} \tag{18}$$

or, in explicit form, it is

$$T_w(\tau) = \frac{T_s - \delta + \frac{\delta T_s}{\tau}}{1 - \frac{1}{M}\left\{[\phi_1(\tau)(\tau+\delta)+R]P(\tau) + \phi_2(\tau)(R+\tau+\delta)(1-P(\tau))\right\}}$$

$$\text{where} \left\{ \begin{array}{c} \phi_1(\tau) = \dfrac{M}{\tau+\delta} + \dfrac{1}{1-e^{\frac{\tau+\delta}{M}}} \\ \phi_2(\tau) = \dfrac{M}{R+\tau+\delta} + \dfrac{1}{1-e^{\frac{R+\tau+\delta}{M}}} \\ P(\tau) = e^{-\frac{R+\tau+\delta}{M}} \end{array} \right\} . \tag{19}$$

The classic method of minimizing such a function numerically is to solve for zeros of the derivative using an iterative approach such as Newton-Rhapson. In this case obtaining the derivative analytically is far from trivial, so we will use a simple bisection method to determine the minimum. Starting with function evaluations at $t_{lo} = \epsilon$ and $t_{hi} = M + R + \delta$ we compute $t_{avg} = \frac{1}{2}(t_{lo} + t_{hi})$. Then we compute the derivative at each of the three points using a first order central difference scheme and compare the signs of the derivatives. The minimum value of $T_w(\tau)$ is located in whichever of the sub ranges $[\tau_{lo}, \tau_{avg}]$ or $[\tau_{avg}, \tau_{hi}]$ have derivatives of opposite signs at their endpoints. By successively bisecting subsegments of the range $[\epsilon, M]$ until $\tau_{hi} - \tau_{lo} < \epsilon$ we arrive at an approximation of τ that minimizes the solution wall clock time.

3 The Simulation

To validate the model results, a simulation was developed that generates pseudo-random interrupts from an exponential deviate. It then simulates the execution of a real code using these randomly generated interrupts to determine in which compute segments restarts will occur. By keeping track of both the accumulated wall clock time and the accumulated solve time we are able to determine the total wall clock time for solution by cycling through compute segments until the accumulated solve time from each of the segments is equal to the total solve time. The simulation is run multiple times and results are reported in terms of the median wall clock time and a 90% range (represented by the error bars in figures 3-5).

4 Results

In our first test case illustrated by figure 3 we find excellent agreement between the first order model, complete model, and simulation results. Notice that median wall times reported by the simulation increase more slowly for $\tau > \tau_{opt}$ than for $\tau < \tau_{opt}$ even though the variances are larger. This means that an overly conservative estimate of the optimum restart interval may actually result in longer run times than an equivalent over prediction.

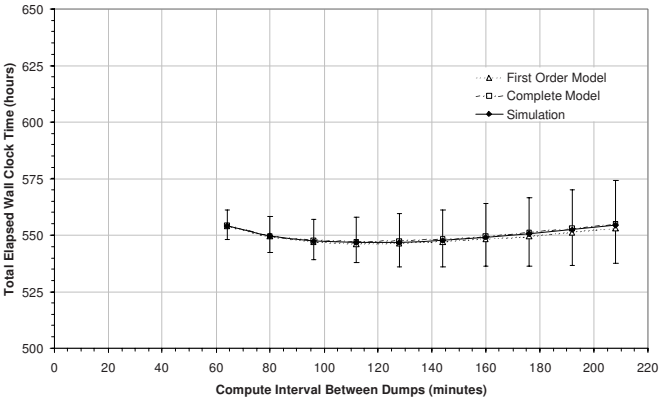


Fig. 3. A comparison of model and simulation results for $M = 24$ hrs. $T_s = 500$ hrs, $R = 10$ mins, and $\delta = 5$ mins. The complete model predicts $\tau_{opt} = 114$ mins.

In figure 4 the MTTI decreases and we see a significant increase in runtime and shortening of the optimum compute interval. We find that the first order model is beginning to diverge from the simulation results because the linear approximation to the exponential is breaking down as $\tau + \delta$ approaches $O(M)$.

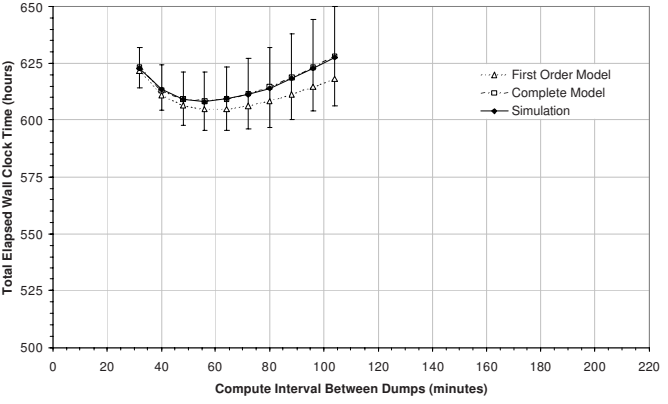


Fig. 4. A comparison of model and simulation results for $M = 6$ hrs. $T_s = 500$ hrs, $R = 10$ mins, and $\delta = 5$ mins. The complete model predicts $\tau_{opt} = 56$ mins.

Figure 5 represents a pathological case where for some choices of τ the interval between restart dumps actually exceeds the MTTI for the system. The first order model fails to predict either the total runtime or the minimum runtime. However, agreement between the simulation and the complete model is still excellent.

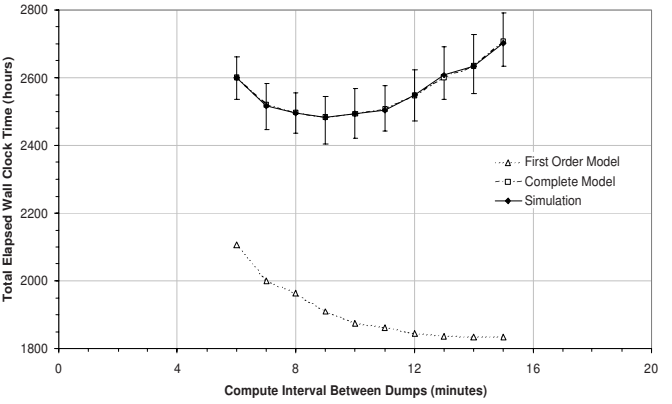


Fig. 5. A comparison of model and simulation results for $M = 15$ mins. $T_s = 500$ hrs, $R = 10$ mins, and $\delta = 5$ mins. The complete model predicts $\tau_{opt} = 9$ mins.

Finally, figure 6 shows a comparison between the calculated compute interval τ for all three models. The agreement between the predictions of the modified model and the complete model are within 5% for all MTTI greater than 1 hour.

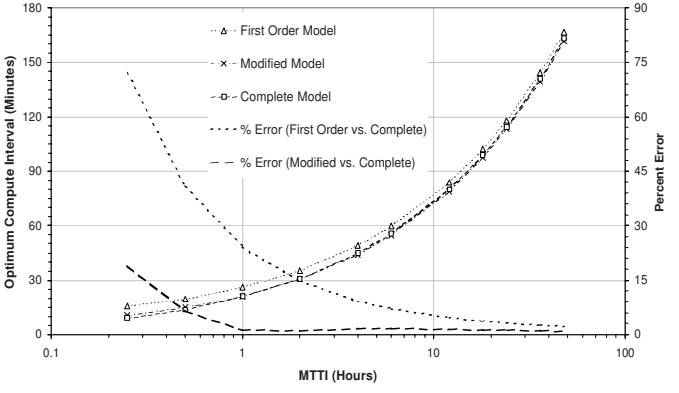


Fig. 6. A comparison of all three models for predicting the optimum compute interval between writing checkpoint files. $T_s = 500$ hrs, $R = 10$ mins, and $\delta = 5$ mins.

Figures 3-5 show the results of the first order and the complete model compared to the simulation results with error bars to indicate the range in which 90% of the simulation data fell. The main purpose of these is to validate the complete model against the simulation results. We also find that the first order model completely fails to predict simulation results for small M. Not only does

it under-predicted the total wall clock time, it also fails to locate the minimum because the function has been sufficiently flattened by using the linear term of a divergent series to represent the exponential. Finally, figure 6 compares the first order model and the modified model to the complete model. This comparison is intended to demonstrate that the modified model is in fact an improvement over the first order model. Empirically, we see that for $(\tau + \delta)/M < \frac{1}{2}$ we get good agreement between the τ_{opt} predicted by the modified model and that predicted by the complete model.

5 Conclusions

We considered three models for predicting the runtime and optimum restart interval. We compared their runtime predictions to the results of a simulation. Since the complete model agrees with simulation in predicting total application runtime we used it to establish a baseline for the optimal compute interval. Then we compared the first order and modified models to the complete model and concluded that the optimum restart interval estimate given by the modified model is in fact an improvement over the first order model. In other words, $\tau_{opt} = \sqrt{2\delta(M + R)} - \delta$ is an excellent estimator of the optimum compute interval between restart dumps for values of $(\tau + \delta)/M < \frac{1}{2}$.

References

1. Bruno, J., Coffman, E. G.: Optimal Fault-Tolerant Computing on Multiprocessor Systems. Acta Informatica **34** (1997) 881–904
2. Coffman, E. G., Gilbert, E. N.: Optimal Strategies for Scheduling Checkpoints and Preventive Maintenance. IEEE Transactions on Reliability **39** (1990) 9–18
3. Dimitrov, B., Khalil, Z., Kolev, N., Petrov, P.: On the Optimal Total Processing Time Using Checkpoints. IEEE Transactions on Software Engineering **17** (1991) 436–442
4. Kwak, S. W., Choi, B. J., Kim, B. K.: An Optimal Checkpointing-Strategy for Real-Time Control Systems Under Transient Faults. IEEE Transactions on Reliability **50** (2001) 293–301
5. Ling, Y., Mi, J., Lin, X.: A Variational Calculus Approach to Optimal Checkpoint Placement. IEEE Transactions on Computers **50** (2001) 699–708
6. Magazine, M. J.: A Closed-Form Solution to a Class of Quadratic Knapsack Problems. INFOR **30** (1992) 6–10
7. Vaidya, N. H.: Impact of Checkpoint Latency on Overhead Ratio of a Checkpointing Scheme. IEEE Transactions on Computers **46** (1997) 942–947
8. Young, J. W.: A First Order Approximation to the Optimum Checkpoint Interval. Communications of the ACM **17** (1974) 530–531

Terascale I/O Solutions

Nathan Stone, John Kochmar, Paul Nowoczynski, J. Ray Scott, Derek Simmel, Jason Sommerfield, and Chad Vizino

Pittsburgh Supercomputing Center, 4400 Fifth Avenue, Pittsburgh, PA 15213, USA
{stone, kochmar, pauln, scott, dsimmel, jasons, vizino}@psc.edu

Abstract. PSC has architected and delivered the TCS-1 machine, a Terascale Computing System for use in unclassified research. PSC has enhanced the effective usability and utilization of this resource by providing custom I/O solutions in four key areas: high-performance communication, high-performance file migration, checkpoint/recovery and an updated hierarchical storage management system. These I/O solutions have a synergistic effect that is leveraged in their design, implementation and integration. Each successive enhancement builds on its predecessors, thereby exacting the highest performance (*e.g.* multi GB/sec file transfers) from the available hardware. This paper presents a technical overview of these solutions from design to integration to application.

1 Introduction

We have four well-developed and unique custom I/O solutions to suit the needs of TCS-1 users: high-performance communication, high-performance file migration, checkpoint/recovery and a new hierarchical storage manager. These solutions, although they address independent issues, have been integrated in ways that can and will allow TCS-1 users an unprecedented quality of service in these areas. The impact of these solutions enhances the performance both of the TCS-1 machine and of the applications that run there.

1.1 Machine Configuration

The TCS-1 machine is a cluster of more than 750 Compaq ES-45 quad-processor servers, which are built with 1.0 GHz EV68 generation Alpha processors. Thus there is a pool of more than 3000 processors available to the compute partition of the machine, which does not include "hot" spare nodes positioned for immediate and automated scheduling in the case of a compute node failure. The nodes are interconnected via two independent communication fabrics (or "rails") called "QsNet",[1] by Quadrics. The TCS-1 is a Compaq AlphaServer SC[2] system. As such

[1] See: http://www.quadrics.com/.
[2] See: http://www.hp.com/techservers/systems/sys_sc.html.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 13-22, 2003.
© Springer-Verlag Berlin Heidelberg 2003

its software configuration is composed from three layers. The operating system on each node is Tru64. Nodes are grouped into sets of 32 by a cluster-management layer called TruCluster, which creates clusters from Tru64 nodes. In addition to providing high-availability of services that run within the TruClusters, this layer provides some advanced file system features. The third layer is called the Resource Management System (RMS). RMS provides for some level of hardware event handling, a basic job queueing interface, a parallel job launching mechanism and the ability to divide the available nodes into "partitions" for convenience. Many of our high-level site customizations are layered over RMS, from custom scheduling and job control to event monitoring and management, although some site customizations necessarily intercede at a lower level.

2 TCSCOMM

The first feature we present is a high-performance communication library called TCSCOMM. This is a user-level library that allows applications to use the QsNet to communicate outside of the RMS environment. By using TCSCOMM, system level daemons and user-level applications can transmit data at near Quadrics network rated speeds, with extremely low latencies.

QsNet is a high bandwidth, low latency network comprised of 2 major components: the Elan network cards and the Elite switch components. A full "fat-tree" switch topology allows point-to-point non-contending communication between any two nodes at full network bandwidth. On TCS-1 full bandwidth is approximately 250MB/s per QsNet rail with a 5 microsecond latency.

2.1 The Elan System Libraries

Two system libraries are available for communicating over QsNet using the Elan cards: the `libelan`[3] library, and the `libelan3` library. The `libelan` library is a high-level library that facilitates communications between processes locally and on other machines connected to the switch. This library requires RMS to set the memory layout, communications contexts and capabilities. This library is designed to be less architecture dependent, so that applications and libraries written against this library can be more easily ported to future versions of the QsNet interconnect.

The `libelan3` library, a lower level interface, is specific to the elan3 card and interconnect and much of the `libelan` library is based on it. With this interface, applications can control the specific memory layout for communications, as well as create specific context and capabilities for the communications. In addition applications can access the lowest levels of the communication layer, providing the highest levels of performance at the expense of code complexity and portability to future versions of the interconnect.

[3] Quadrics Supercomputers World Ltd. Elan Programming Manual, January 1999.

2.2 A Custom Elan3-Based API

It was our intent to use the QsNet for system applications and daemons (running outside of RMS), so we designed the TCSCOMM API to use the `libelan3` library rather than the `libelan` library. Using the `libelan3` library had the following benefits:

- It allowed us to provide communications between Tru64 and Linux (IA-32 and IA-64) hosts on the same QsNet network by ensuring the same memory layouts on the two architectures. By default, the `libelan` library chooses a layout that matches the process memory layout, which differed between the various architectures and OSes.
- It allowed us to create custom capabilities and contexts for the running applications. This allowed us to run outside of RMS, so that we could create system daemons and applications that didn't need to depend on RMS to run.
- Using `libelan3` also allowed us to develop a multi-rail communications scheme that we use to improve performance by striping communications over multiple independent QsNet rails, improving over-all performance.

At the moment, the library only supports point-to-point communications. There is no concept in this library of a multi-cast operation, even though the QsNet supports this type of operation in the form of a hardware broadcast.

2.3 Memory Management

Mapping the memory directly from the Elan card allows for the greatest performance, over 220 MB/s on a single rail, and better than 400 MB/s over 2 rails. This contrasts with 170MB/s and 270MB/s when not using the API's memory malloc routines, respectively. The maximum memory available to use off of the card is 32 MB. This size is configurable at library compile time, and in addition to the 32MB available for `tcscomm_Malloc`, 16MB is used for send/receive buffers for the `tcscomm_Send` and `tcscomm_Recv` calls when using memory not allocated via `tcscomm_Malloc`.

2.4 Message Transfer Protocol

The TCSCOMM library uses a two-stage method to transfer data from one client to another:

- Queued DMA operations are used to send data transfer requests from the sender to the receiver.
- The receiver uses the elan3_getdma function to transfer large blocks of data from the sender to the receiver, based on a request pulled from the message queue.

When one process wants to send data to another process, it stores the data in an area of memory known to the library, and then tells the remote process to fetch the data, allowing the receiver to synchronize the data transfers.

3 TCSIO

The second feature we present is a high-performance file migration utility called TCSIO. This is an object-oriented client-server toolkit. As such, we introduce the presence of an "I/O daemon" that runs on each node where disk resources are to be presented—both file servers and compute nodes, for the purpose of accessing their local disks. The I/O daemons are conversant in multiple transport protocols, including a reference implementation in TCP/IP and an expansion to the TCSCOMM library. The protocol for connecting to an I/O daemon begins with a TCP/IP socket connection and includes, among other things, a negotiation to the highest performance protocol for bulk data transport. Metadata are always transmitted over the TCP/IP connection. By separating functionality between the client and the I/O daemon we achieve several useful benefits, as follows.

3.1 High-Performance

All of the performance-oriented software is isolated within the daemons, minimizing exposure to the user's environment. This client-server subsystem has been shown to achieve transfers at roughly 2 GB/sec aggregate, utilizing parallelism for the transfer of multiple files. Furthermore, distributed clients transferring to /dev/null on the file servers have been benchmarked at roughly 2.4 GB/sec, revealing that faster file systems on the file servers would improve the overall transfer performance. This is an area of ongoing work.

3.2 Bypass NFS Cross-Mounts

The presence of an I/O daemon at the point of origin of disk resources allows us to explicitly transfer files from source to destination via our custom TCSIO protocol. This has the immediate advantage that we can replace NFS cross-mounts with our higher-performance protocol, which is faster than NFS due to the use of TCSCOMM. Furthermore, in a system the size of TCS-1, the high node-count can lead to literally hundreds of cross-mounted file systems. This presents a scaling problem that can significantly degrade the performance of conventional files systems such as NFS.

3.3 Third-Party Copy

By passing a copy *request* from a client to the I/O daemon the client can leverage the I/O capabilities of a reliable third-party process. Thus a compute job can, for example, pass a non-blocking file migration request to an I/O daemon and return to CPU-intensive compute tasks while the I/O daemon (a separate process) handles the I/O intensive tasks involved with the migration.

3.4 Pluggable Transport Protocols

Communications in the I/O daemon are encapsulated in a "Connection" object. In this way, future transport protocols can be added with minimal code modification, in a manner completely transparent to user applications running on the system.

3.5 I/O Redirection

Some applications of TCSCOMM are being introduced directly into users' applications. One of these instances will use the TCSCOMM library directly, not to migrate an existing file, but to redirect file-oriented I/O to a remote file server. This will be further described in the Checkpoint/Recovery section below.

3.6 Extensible Service Integration

The I/O daemon has already been instrumented with additional features, beyond file migration, for other custom TCS-1 services. Thus, the I/O daemon provides a foundation for building other integrated services, like checkpoint/recovery and our new HSM implementation, which we discuss below.

4 Checkpoint/Recovery (CPR)

We have created an application-level checkpoint/recovery (CPR) library to enable TCS-1 users to checkpoint and recover jobs in a way that makes optimal use of machine resources and provides additional features otherwise inaccessible to them. Of dominant concern in the design of this library was the minimization of "non-compute" time—time spent either in I/O or library overhead.

Some features of the CPR library are discussed in a previous publication,[4] including the description of independent I/O schemata (also called "plans") and how a user's job selects the desired I/O schema for a particular job execution by the means of an environment variable. Of note is the fact that each I/O plan is a complete implementation for generating and retrieving checkpoint data. As such, any of the available plans could be selected for any job execution, although certain job types may experience performance differences between the various I/O schemata. Furthermore, since all plans are accessed via the same fixed API, the CPR library is extensible to support new schema implementations. To access a newly added schema, users need only re-link their application to the updated CPR library and update the corresponding CPR plan environment variable.

---

[4] N. Stone, J.R. Scott, J. Kochmar, J. Sommerfield, R. Subramanya, R. Reddy, K. Vargo, "Mass Storage on the Terascale Computing System", Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Technologies, p. 67, 2001.

The checkpoint implementation also supports a passive pre-emption mechanism. It provides users with a means for checking whether their job should write a checkpoint in anticipation of an impending job termination.

The recovery implementation is also dependent on the selected checkpoint schema. All plans rely upon some type of redundancy in storing checkpoint files, as a means of protecting against lost of files, storage media or file-serving nodes. Recovery plans range from retrieving full duplicates of lost files to regenerating lost files from parity files. In all cases the recovery is handled by a single call to the API.

4.1 Failover Syntax

When identifying files internally, the checkpoint system utilizes a failover syntax that expresses potentially redundant routes to a given file path. Failover paths are generally strings that include host and path notations with reserved keywords. Here are a few illustrative examples:
"`node[4-10]:/scratch/`" indicates the absolute path "`/scratch/`" that exists only on the hosts "node4, node5, … node10" and is only accessible locally on those nodes;
"`{local}:/local/checkpoint/`" indicates the absolute path "`/local/checkpoint/`" that exists on all nodes but is exclusively accessible on the node where it is written;
"`{cluster}:/usr/storage/`" indicates the absolute path "`/usr/storage/`" that is mounted from the cluster file system and is thus accessible equally from all nodes within a given TruCluster set;
"`node[0-9](n+1):/usr/speedy/`" indicates the absolute path "`/usr/speedy/`" that exists on hosts "node0, node1, … node9" and are pair-wise redundantly accessible (*e.g.* via multi-initiator SCSI), for example node0 and node1 can access each other's /usr/speedy, and similarly for node2 and node3, node4 and node5, and so on.
Using this failover notation during the creation and registration of checkpoint files allows the propagation of this redundancy information to the recovery step, thereby providing alternative access routes to checkpoint data in the case of hardware or other failures.

Performance of the checkpoint I/O methods is internally monitored and reported to an external database for offline analysis and evaluation. In the event that this reporting becomes burdensome either to the user's job execution or the system's recording resources this behavior can be deactivated by a switch in a system configuration file `/etc/tcsiod.conf`.

4.2 Scheduler Integration

Checkpoint/recovery is of some usefulness without automation, but its full value is only achieved when it is integrated into the scheduling environment for automated restart and recovery. Our CPR system has been integrated into the Resource

Management System (RMS) to automatically flag machine-failure conditions as distinct from job failure conditions. It has also been integrated into "Simon," our custom scheduler based upon OpenPBS.[5] PBS, like most schedulers, provides hooks for scripts that can be run immediately before and after job execution; for PBS these are called the "prologue" and "epilogue" scripts. By providing a supplement to the epilogue script we ensure proper cleanup of CPR resources, re-running of failed jobs that will need to resume from checkpoint and notification of administrators and the user of noteworthy job-related conditions. The epilogue script even automates refunding of "lost" compute time, which we discuss further below. The precise steps executed within that script depend on both the job exit status, captured by RMS, and the state information, if any, held in the checkpoint system. In this way jobs that abnormally terminate because of machine failure conditions will be automatically recovered without intervention from the administrator or the user.

4.3 Accounting Integration

Many computing sites monitor their usage, charging users or grants directly for resources consumed by compute jobs. The accuracy of the records is imperative. Since CPR handling occurs within a user's job, CPR handling can cause extensions to the wall-time consumed by a user's job and thus, an inflated job charge. The integration of the CPR system and the scheduling system is made complete by a further integration of the accounting system. To eliminate charging the user for CPR handling, the CPR system keeps track of the time spent in CPR activities and reports this in a table that is available to our accounting system. Furthermore, by measuring the time between the last complete checkpoint and the end of a failed job one can determine the amount of time "lost" by a compute job that terminated due to machine failure and post that in a manner accessible to the accounting system as well. In our CPR system the PBS epilogue script records this automatically, as noted above. In this way, resources lost or consumed by CPR activities can be credited back to the user's allocation.

By policy, neither nodes nor processors within TCS-1 are time-shared. Thus, the formula for job charging on TCS-1 is as follows:

$$C = \sum_{i=1}^{N_r} (UT_i - RT_i - \min(LT_i, MaxLT)) * N$$

where:

C = total charge for user's job (node-hrs)
N_r = total number of RMS resources created by OpenPBS job
UT_i = wall-clock time (hrs) for each RMS resource
RT_i = time spent in checkpoint file recovery for each RMS resource
LT_i = "lost time" calculated for each RMS resource
$MaxLT$ = maximum refundable "lost time", set by policy
N = number of nodes requested

[5] See: http://www.openpbs.com/.

4.4 Examples

Encouraging users to make use of application-level checkpointing is, to a large degree, a matter of user education and is motivated by the direct advantages to the user. Aside from the obvious benefit of job recovery, users gain recovery automation, an assurance of checkpoint file availability, and access to the highest-performance storage resources on the TCS-1 by using our CPR library. As a first step toward achieving this user education we provide many examples of CPR-instrumented codes ranging from simple two-integer test cases to Laplace solvers, in both C and Fortran. In this way we hope to make it easier for users to understand how to properly utilize the library and its features.

5 Hierarchical Storage Manager

We have designed a new archiving system that will tightly integrate into TCS-1. The Scalable Lightweight Archival Storage Hierarchy (SLASH) is a heterogenous system comprised of two major components. The first component is the Linux Cache Node (LCN) cluster. LCNs are arranged as a loosely distributed caching cluster that is enmeshed in the QsNet network. The second component is XFS, a metadata file system that distributes and maintains the metadata for the distributed disk caches. The third component is DMF, a tape archiver that manages the tape archives and their front-end disk cache. XFS and DMF are tightly integrated, as discussed below.

5.1 Linux Cache Nodes (LCNs)

The Linux cache nodes (LCNs) have been designed to handle considerable amounts of I/O and house many disks. LCNs have the internal capacity to hold up to 32 commodity IDE devices. Given today's maximum disk density a single LCN's usable storage capacity is over 8 terabytes.

The software layer, SLASH-SWL, executes a number of tasks such as getting permission from XFS for file system update operations and obtaining LCN residency information for specific file. SLASH-SWL also manages the local cache consistency, which offloads tasks from XFS.

SLASH-SWL exports an API that most file transport applications, *e.g.* the TCS I/O daemons, can use. When uploading data to the HSM, an I/O daemon running on an LCN obtains the target upload file descriptor from a SLASH-SWL API function (“`hsmCacheUploadInit`”) instead of directly from an `open()` system function. For HSM downloads another SLASH-SWL API call (“`hsmCacheDownloadInit`”) is used, which can either return a local read-only file descriptor or, if the local LCN does not have the most recent cached file, redirect I/O to another I/O daemon on the LCN holding the most recent version.

5.2 XFS/DMF

The XFS[6] subsystem is an advanced Unix file system that incorporates capability for user-supplied information into its metadata. It runs on Silicon Graphics, Inc. (SGI) hardware as well as most Linux platforms. The Data Migration Facility (DMF)[7] is an additional software package, also from SGI, that integrated HSM functionality into the XFS file system. DMF is currently most mature on the SGI hardware platform, though Linux options are emerging. The current architecture has a moderately sized and robust SGI server connected running the XFS/DMF file system. The design includes over ten terabytes of disk cache presented by several external RAID enclosures and petabytes of tape storage connected to the SGI machine. Failover paths to these storage devices are possible through a Fibre Channel switch.

SLASH uses the “user metadata location" of the XFS metadata to store its own internal consistency information. Noted information stored here is the identifier of the LCN, if any, which holds the most recent version of the file. SLASH requests from the LCNs are transmitted via an RPC layer to the XFS file system.

5.3 User Access Methods

The preferred access method for SLASH is the TCSIO subsystem. Used within the Terascale Computing system, TCSIO to SLASH will perform data transfers via the Quadrics network. When transferring groups of files, such as in the case of reading or writing checkpoint files, TCSIO can take advantage of SLASH's large distributed cache by batching many simultaneous requests to the LCNs. This method will yield bandwidths of order gigabytes per second.

A secure, interactive client, similar to 'ftp', will also be provided. This will be accessible via the wide or local area network. The interactive client will be built with the SLASH-SWL API so that it will be able to take advantage of the large distributed cache.

6 Total Integration

We have presented a discussion of application services designed to augment both the performance of the TCS-1 machine and the applications that run there. Our innovations range from communications to file systems to job checkpoint/recovery to hierarchical storage, yet all of these areas have been integrated to extract the highest

[6] See http://oss.sgi.com/projects/xfs/.
[7] See http://www.sgi.com/products/storage/software.html#dmf.

possible performance from this machine. The I/O enhancements described above are not mere research constructs. These are deeply integrated into one another and into the production services now used by many TCS-1 users. In addition, some of these services have been designed and written in a machine-independent or even platform-independent manner to facilitate portability of these facilities to additional platforms at PSC and elsewhere in the scientific computing community.

Scaling Molecular Dynamics to 3000 Processors with Projections: A Performance Analysis Case Study

Laxmikant V. Kalé, Sameer Kumar, Gengbin Zheng, and Chee Wai Lee
{kale,skumar2,gzheng,cheelee}@cs.uiuc.edu

Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract. Some of the most challenging applications to parallelize scalably are the ones that present a relatively small amount of computation per iteration. Multiple interacting performance challenges must be identified and solved to attain high parallel efficiency in such cases. We present a case study involving NAMD, a parallel molecular dynamics application, and efforts to scale it to run on 3000 processors with Tera-FLOPS level performance. NAMD is implemented in Charm++, and the performance analysis was carried out using "projections", the performance visualization/analysis tool associated with Charm++. We will showcase a series of optimizations facilitated by projections. The resultant performance of NAMD led to a Gordon Bell award at SC2002.

1 Introduction

How does one understand and optimize performance of complex applications running on very large parallel machines? The complexity of application means that multiple competing and interacting factors are responsible for its performance behavior. These factors often mask each other. Further, different factors may dominate as the number of processors used changes. Therefore, performance problems that manifest on the largest machine configuration cannot be debugged on smaller configurations — one must make most of the few runs one can get on the large configuration.

The approach we have pursued for this problem is based on application-level visual and analytical performance feedback. This is facilitated further by use of Charm++ [KK96] and more recently, Adaptive MPI [BKdSH01] which enables MPI programs to use the features of Charm++. The language runtime (Section 2), is capable of automatically recording substantial performance information at a level that makes sense to the application programmer.

We will illustrate this approach via a performance optimization case study. The application we have chosen is NAMD, a production-quality parallel molecular dynamics program developed using Charm++, and in routine use by biophysicists. The benchmark we describe runs each timestep in 25 seconds on a single processor of the PSC Lemieux machine, which consists of 3,000 processors! Given the amount of coordination and communication that must be done in each time-step, one can see the performance challenges involved in completing each timestep in 12 msecs, a result we have achieved.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 23–32, 2003.
© Springer-Verlag Berlin Heidelberg 2003

We will first introduce Charm++ and Projections, the performance analysis tool associated with Charm++. We will then describe the parallel structure of NAMD. Utility of projections, and the performance optimization methodology it engenders is described via a series of examples. Each example shows how a problem is identified (and optimizations suggested) using projections, and the effect of resultant optimizations.

2 Charm++ and Projections

Virtualization is the driving force behind Charm++. The basic idea in virtualization is to let the programmer divide the program into a large number of parts independent of the number of processors. The parts may be objects, for example. The programmer does not think of processors explicitly (nor refer to processors in their programs). Instead they program only in terms of the interaction between these virtual entities. Under the hood, the Run Time System (RTS) is aware of processors and maps these virtual processors (VPs) to real processors, and remap them whenever necessary. Charm++ supports asynchronous method invocations, reductions and broadcasts which work correctly and efficiently in the presence of dynamic migrations of objects among processors. It uses the remapping capability to do automatic load balancing.

2.1 Performance Analysis Tools and Techniques in Projections

Projections is a performance analysis subsystem associated with the Charm++. It consists of an automatic instrumentation and tracing system, and an interactive graphic analysis system.

Efficient Automatic Tracing. Since Charm is a message driven system, tracing can be fully automated. Specifically, the RTS knows when it is about to schedule the execution of a particular method of a particular object (in response to a message being picked up from the scheduler's queue).

Users can start and stop instrumentation during specific phases of the program. Different tracing modules can register themselves to the RTS via callbacks. The overhead on the application when NOT tracing is that of an if statement per event (which is typically associated with a message: so not much overhead in comparison). Even this overhead can be eliminated in production runs by linking the program with an optimized version of the RTS.

Unlike MPI, in Charm++ we can retrieve the idle time from the RTS. In MPI when one is waiting at a barrier or a recv, all the time spent in that gets called as communication overhead. However this often includes idle time, because another processor hasn't arrived at the barrier (or hasn't sent the message). Charm RTS can cleanly separate communication overhead from such idle time. This prevents users from erroneous conclusions that the performance is low due to "the slow barrier operations", whereas it may be due to load imbalances.

Two important tracing modules are *log* and *summary*. In the log mode each event is recorded in full detail (including timestamp) in an internal buffer.

The *summary* module reduces the size of output files and memory overhead. It produces (in the default mode) a few lines of output data per processor. For each entry-method it records standard profiling information such as total (max and average) time spent in it and the number of calls to it. It uses an adaptive strategy to limit the size of time-dependent data. This data is recorded in bins corresponding to intervals of size 1ms by default. If the number of intervals exceeds a threshold then the bin-size is increased to shrink the data into fewer bins. This way the size of recorded data is kept bounded.

Analysis and Visualization. The visualization system supports multiple views: A *graph* view shows processor utilization as a function of time (using pre-set or user-specified bins of intervals) for a specified set of processors. One can also turn on a display of specific message types, to see when each phase of the computation started, and how dispersed over time its execution was. One of the simplest uses of this view is to identify the interval of times for a more detailed study.

The *profile* view shows a stacked column bar for each selected processor, for a selected time interval. The time spent by each processor in various activities is shown within each bar. This view clearly separates idle time and communication overhead. This is one of the most useful "summarizing" views in projections. One can identify overloaded processors, unexpected time-spent in specific methods, or high communication overhead from this.

The animation view shows time-varying behavior across processors, which can be arranged in multiple topologies (2D grid is most popular). Although this view is initially interesting, and can provide a "Gestalt" impression for the performance behavior, we find it to be not as useful as other static views, which one can stare at for insights..

The above views can be supported with both summary or log data. The timeline view, which is supported with the log data only, is essentially similar to other timeline tools such as Upshot, and Paragraph [HE91]. It is a highly sophisticated view which presents additional detailed information about events by simple mouse clicks.

3 Overview of NAMD

NAMD is a molecular dynamics program designed for high performance simulation of large biomolecular systems [PZKK02]. Each simulated timestep involves computing forces on each atom, and "integrating" them to update their positions. The forces are due to bonds, and electrostatic forces between atoms within a cut-off radius.

NAMD 2 is parallelized using Charm++ via a novel combination of force and spatial decomposition to generate enough parallelism for parallel machines with a large number of processors. Atoms are partitioned into cubes whose dimensions are slightly larger than the cutoff radius. For each pair of neighboring cubes, we assign a non-bonded force computation object, which can be independently mapped to any processor. The number of such objects is therefore 14 times (26/2 + 1 self-interaction) the number of cubes.

The cubes described above are represented in NAMD 2 by objects called *home patches*. Each home patch is responsible for distributing coordinate data, retrieving forces, and integrating the equations of motion for all of the atoms in the cube of space

owned by the patch. The forces used by the patches are computed by a variety of *compute objects*. There are several varieties of compute objects, responsible for computing the different types of forces (bond, electrostatic, constraint, etc.). On a given processor, there may be multiple "compute objects" that all need the coordinates from the same home patch. To eliminate duplication of communication, a "proxy" of the home patch is created on every processor where its coordinates are needed. The parallel structure of NAMD is shown in Fig. 1.

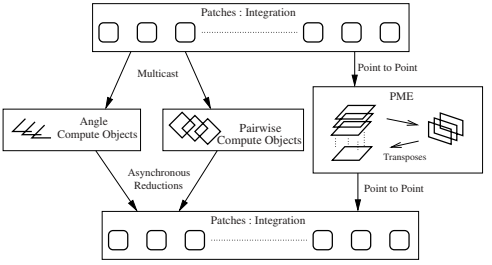


Fig. 1. Parallel structure of NAMD

NAMD employs Charm++'s measurement-based load balancing. When a simulation begins, patches are distributed according to a recursive coordinate bisection scheme, so that each processor receives a number of neighboring patches. All compute objects are then distributed to a processor owning at least one home patch. The framework measures the execution time of each compute object (the object loads), and records other (non-migratable) patch work as "background load." After the simulation runs for several time-steps (typically several seconds to several minutes), the program suspends the simulation to trigger the initial load balancing. The strategy retrieves the object times and background load from the framework, computes an improved load distribution, and redistributes the migratable compute objects.

The initial load balancer is aggressive, starting from the set of required proxies and assigning compute objects in order from larger to smaller, avoiding the need to create new proxies unless necessary. Once a good balance is achieved, atom migration changes very slowly. Another load balance is only needed after several thousand steps.

4 Performance Analysis and Optimizations

We will present the performance optimizations we carried out with the help of Projections in a series of examples. The first two examples involve runs on the ASCI Red machine, while the rest are on PSC Lemieux.

Grainsize Analysis: The benchmark application we used on ASCI Red machine was a 92,000 atom simulation, which took 57 seconds on one processor. Although it scaled reasonable well for few hundred processors, initial performance improvements stalled

beyond 1,000 processors. One of the analysis using projections logs we performed identified a cause. Most of the computation time was spent in force-computation objects. However, as shown in Figure 2, the execution time of computational objects was not uniform: it ranged from 1 to 41 msecs. The variation itself is not a problem (after all, Charm++'s load balancers are expected to handle that). However, having single objects with execution time of 40+ msecs, in a computation that should ideally run in 28 msecs on 2000 processors was clearly infeasible! This observation, and especially the bimodal distribution of execution times, led us to examine the set of computational objects. We found the culprits to be those objects that correspond to electrostatic force computations between cubes that have a common face. If cubes touch only at corners, only a small fraction of atom-pairs will be within the cut-off distance and need to be evaluated. In contrast, those touching at faces have most within-cutoff pairs. Splitting these objects into multiple pieces led to a much improved grainsize distribution as shown in Fig. 2b.

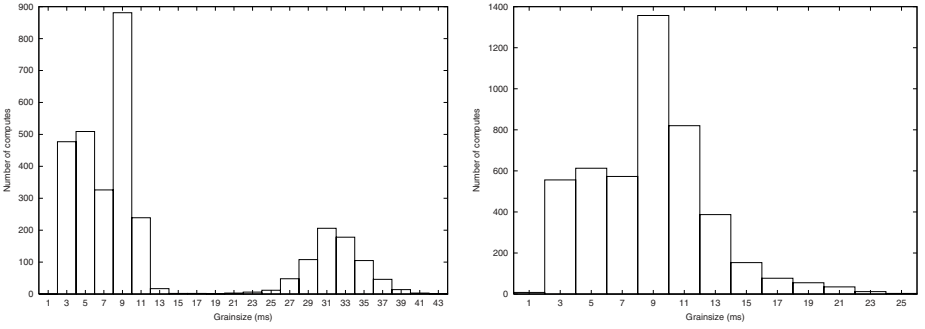


Fig. 2. Grainsize Distribution on ASCI Red

Message Packing Overhead and Multicast: Projections timeline is a very useful tool - it can easily tell what a parallel program is doing by showing the recorded events along the time axis. From a timeline view, one can easily identify what is bad or unexpected and try to optimize it. One of the analysis using projections logs in timeline tool we did for NAMD exemplified this.

Fig. 3 shows a timeline view of one 1024 processors run. The circled event shows the integration phase in NAMD. During the integration phase, each patch combines all incoming forces and calculates the new atom positions, etc. At end of integration, the patch sends the updated atom information to all the computes whose computation depends on it via a multicast. In timeline, the white little ticks at the bottom of the integration event show the message sending events. From Fig. 3, we observed the unusual long time spent in the multicast. The implementation of multicast in Charm++ was not efficient, it treated multicast as individual sends and each send paid the overhead of message copying and allocation. After reimplementing the multicast in Charm++, to avoid this overhead the integration time is greatly shortened as shown in the Fig. 3.

Fig. 3. Effect of Multicast Optimization on Integration Overhead

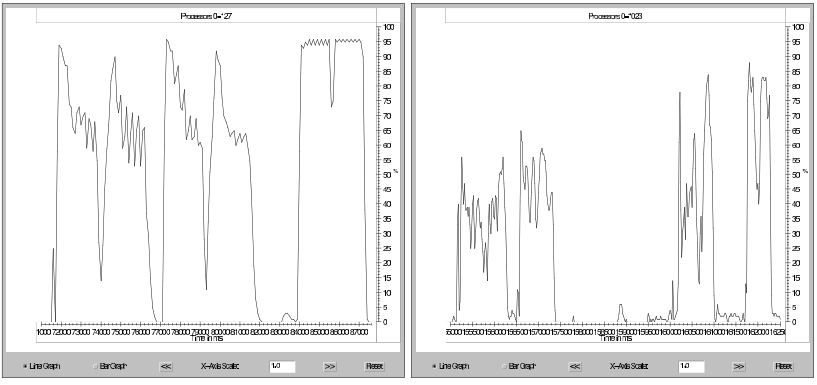


Fig. 4. Processor Utilization against Time on (a) 128 (b) 1024 processors

Load Balancing: Dynamic load balancing was an important performance challenge for this application. The distribution of atoms over space is relatively non-uniform, and (as seen in the grainsize analysis above) the computational work is distributed quite non-uniformly among the objects. We used a measurement-based load balancing framework, which supports runtime load and communication tracing. The RTS admits different strategies (even during a single run) as plug-ins, which use the trace data. We used a specific greedy strategy[KSB+99]. For a 128-processor run, Projections visualization of the utilization graph (Fig. 4(a)) confirmed that the load balancer worked very well: Prior to load balancing (at 82 seconds) relatively bad load imbalance led to utilization averaging to 65-70% in each cycle. However after load balancing, the next 16 steps ran at over 95% utilization.

However, when the same strategy was used on 1024 processors, the results were not as satisfying (Fig. 4 (a)). In particular, (via a profile view not shown here) it became clear that the load on many processors was substantially different than what the load balancer had predicted. Since the greedy strategy used ignored existing placements of objects entirely (in order to create an unconstrained close-to-optimal mapping), it was surmised that the assumptions about background load (due to communication, for ex-

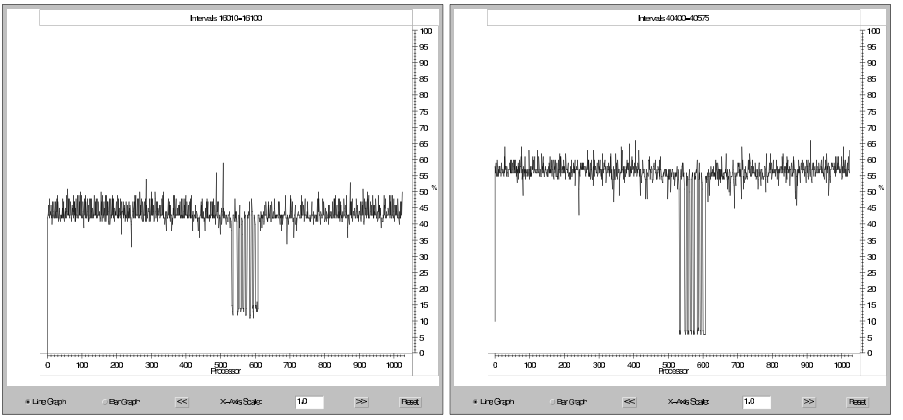
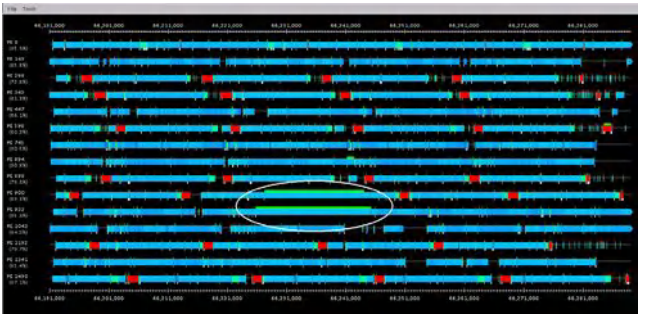


Fig. 5. Processor Utilization for each processor after (a) greedy load balancing and (b) refining

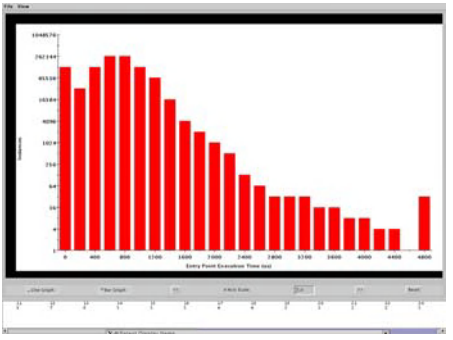
ample) as well as cache performance were substantially different in the new context after the massive object migration induced by load balancer. Since the new mapping was expected to be close to optimal, we didn't want to discard it. Instead, we added another load balancing phase immediately after the greedy reallocation, which used a simpler "refinement" strategy: objects were moved only from the processors that were well above (say 5%) the average load. This ensured that the overall performance context (and communication behavior) was not perturbed significantly after refinement, and so the load-balancer predictions were in line with what happened. In Fig. 4(b), initial greedy balancer works from 157 through 160 seconds, leading to some increase in average utilization. Further, after the refinement strategy finished (within about .7 seconds) at around 161.6 seconds, we can see that utilization is significantly improved. Another view in Projections (Fig. 5), showing utilization as a function of processors for the time intervals before and after refinement, shows this effect clearly.

Note that due to some quirks in the background load, several processors in the range between 500 and 600 were left underloaded by the greedy algorithm. The refinement algorithm did not change the load on those, since it focuses (correctly) only on overloaded processors: having a few underloaded processors doesn't impact the performance much, but having even one overloaded processor slows the overall execution time. Here, we see that 4 overloaded processors (e.g, processor 508) were significantly overloaded before the refinement step, whereas the load is much much closer to the average after refinement. As a result, overall utilization across all processor rises from 45 to 60%.

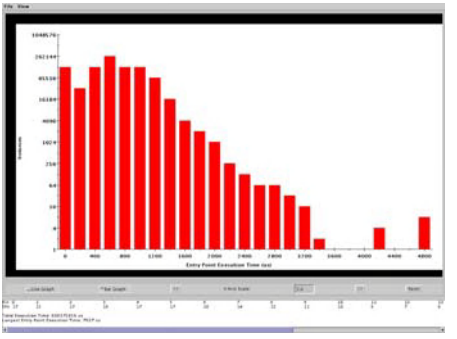
Using 4 processors on each node and "Stretches": The Lemieux machine has 3,000 processors. However, when we tested NAMD on more than 2,000 processors, the program performed relatively poorly. Several separate runs, observed with the timeline view, showed that this was due to several object methods (seemingly) randomly "stretching" to take a much longer time than they took during other time-steps. (For example, see Fig. 6(a)). The two computations highlighted by a green line take over 10 msecs, in comparison with a couple of msecs taken by the same object-method in the other



(a) “Stretched” computations



(b) Before:Large Number of Stretches



(c) After:Fewer Stretches

Fig. 6. “Stretched” executions, and their resolution

timesteps. Such stretching was seen to be more pronounced (stretches of over 50msecs were observed) when we used all 4 processors on each node, in a bid to utilize all 3,000 processors.

Eliminating stretches involved fine tuning the elan communication library provided by Quadrics through significant trial-and-error experimentation. It also involved the use of blocking receives to reduce operating system interference which was causing the stretches. Although we cannot show all the steps involved, the evaluation process was assisted significantly by Projections. Specifically useful was the grainsize analysis histogram that we discussed in an earlier section. Fig. 6(b) shows that about 30 objects were stretched in an earlier 2,250 processor run, beyond 5 msecs, with the largest one well over 20 msecs. After all the fixes were applied, the resultant histogram on 3,000 processors shows (Fig. 6(c)) only 5 stretched executions, with the largest being only 7.5 msecs, in spite of the fact that we are now using all 4 processors on each node.

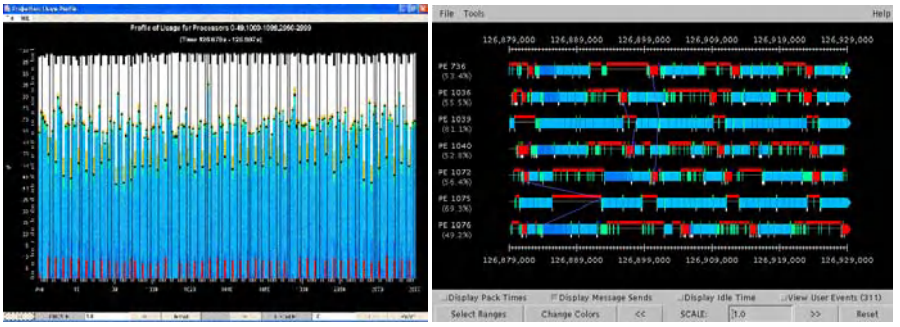


Fig. 7. Profile (a) and Timeline (b) view of a 3000 processor run.

Triumph! Tera-Flop using all 3000 Processors: After using elan based communication with several optimizations, and a few more relatively minor optimizations, we were able to run the application on all 3,000 processors. The time per step was 12 msecs, leading to 1.04 TF performance! Within these 12 msecs, many processors send 30-50 messages containing atoms coordinates (8-10KB each), receive results from the 30-50 force computations objects, carry out asynchronous reductions for total kinetic and potential energies, and migrate atoms to neighboring (26) boxes every 8 steps. For molecular dynamics for biophysics, this is an unprecedented level of performance, exceeding the previous state of art by almost an order of magnitude.

The profile view of selected 200 processors (0-49, 1000-1100, 2950-2999) is shown in Figure 7(a). The white area at the top represents idle time, which is quite substantial (25% or so). Timeline views (Figure 7(b)) show that load balancing is still a possible issue (See processor 1039, identified from the profile view), while communication subsystem still is showing minor, but significant hiccups (a message sent from one of the processor 1076 is not available on processor 1074 for over 10 msec after it is sent). These observations indicate that further performance improvement may be possible!

5 Conclusion and Future Work

We introduced *Projections*, a performance analysis tool used in conjunction with the Charm++ parallel programming system. Projection tracing system automatically creates execution traces, in brief "summary" mode, or detailed "log" mode. We showed how the analysis system, and various views it presents, were used in scaling a production quality application to 3,000 processors and 1 TF.

This experience has helped us identify several new capabilities for Projections. As the problem at hand demanded new analysis, we had to add new capabilities to Projections. Now, we plan to extend Projections, so that users can add such capabilities by expressing simple queries or predicates they want evaluated. The relatively large number and size of trace files in the *log* mode has led us to extend the *summary* mode so that more insightful information can be captured with fewer bytes of summary data. Linking performance visualization system to the source, as done in Pablo, will also be

another useful extension. We have already added preliminary features that use performance counters which will be used in an integrated automatic analysis system. We are currently also in the process of extending these features to AMPI.

Acknowledgements.

NAMD was developed at the Theoretical Biophysics Group (Beckman Institute, University of Illinois) and funded by the National Institutes of Health(NIH PHS 5 P41 RR05969-04). Projections and Charm++ are supported by several projects funded by the Department of Energy (subcontract B523819) and the National Science Foundation (NSF DMR 0121695). The parallel runs were carried out primarily at the Pittsburgh Supercomputing Center(PSC) and the National Center for Supercomputing Applications (NCSA). We are thankful to these organizations and their staff for their continued assistance and for the early access and computer time that we were provided for this work. In particular we would like to thank Jim Phillips(TBG), David O'Neal, Sergiu Sanielevici, John Kochmar and Chad Vizino from PSC and Richard Foster(Hewlett Packard) for helping us make the runs at PSC Lemieux and providing us with technical support.

References

[AMCA+95] V.S. Adve, J. Mellor-Crummey, M. Anderson, K. Kennedy, J.-C. Wang, and D.A. Reed. An Integrated Compilation and Performance Analysis Environment for Data Parallel Programs. In *Proceedings of Supercomputing'95*, December 1995.

[BKdSH01] Milind Bhandarkar, L. V. Kale, Eric de Sturler, and Jay Hoeflinger. Object-Based Adaptive Load Balancing for MPI Programs. In *Proceedings of the International Conference on Computational Science, San Francisco, CA, LNCS 2074*, pages 108–117, May 2001.

[HE91] M.T. Heath and J.A. Etheridge. Visualizing the Performance of Parallel Programs. *IEEE Software*, September 1991.

[KK96] L. V. Kale and Sanjeev Krishnan. Charm++: Parallel Programming with Message-Driven Objects. In Gregory V. Wilson and Paul Lu, editors, *Parallel Programming using C++*, pages 175–213. MIT Press, 1996.

[KSB+99] Laxmikant Kalé, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Aritomo Shinozaki, Krishnan Varadarajan, and Klaus Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151:283–312, 1999.

[LK01] O. Lawlor and L. V. Kalé. Supporting dynamic parallel object arrays. In *Proceedings of ACM 2001 Java Grande/ISCOPE Conference*, pages 21–29, Stanford, CA, Jun 2001.

[PZKK02] James C. Phillips, Gengbin Zheng, Sameer Kumar, and Laxmikant V. Kalé. Namd: Biomolecular simulation on thousands of processors. In *Proceedings of SC 2002*, Baltimore, MD, September 2002.

[San96] Sanjeev Krishnan and L. V. Kale. A parallel array abstraction for data-driven objects. In *Proceedings of Parallel Object-Oriented Methods and Applications Conference*, Santa Fe, NM, February 1996.

[SK96] Amitabh Sinha and L. V. Kale. Towards Automatic Peformance Analysis. In *Proceedings of International Conference on Parallel Processing*, volume III, pages 53–60, August 1996.

Real-time Terascale Implementation of Tele-immersion

Nikhil Kelshikar[1⋆], Xenophon Zabulis[1], Jane Mulligan[4], Kostas Daniilidis[1], Vivek Sawant[2], Sudipta Sinha[2], Travis Sparks[2], Scott Larsen[2], Herman Towles[2], Ketan Mayer-Patel[2], Henry Fuchs[2], John Urbanic[3], Kathy Benninger[3], Raghurama Reddy[3], and Gwendolyn Huntoon[3]

[1] University of Pennsylvania
[2] University of North Carolina at Chapel Hill
[3] Pittsburgh Supercomputing Center
[4] University of Colorado at Boulder

Abstract. Tele-immersion is a new medium that enables a user to share a virtual space with remote participants, by creating the illusion that users at geographically dispersed locations reside at the same physical space. A person is immersed in a remote world, whose 3D representation is acquired remotely, then transmitted and displayed in the viewer's environment. Tele-immersion is effective only when the three components, computation, transmission, and rendering - all operate in real time . In this paper, we describe the real-time implementation of scene reconstruction on the Terascale Computing System at the Pittsburgh Supercomputing Center.

1 Introduction

Tele-immersion enables users at geographically distributed locations to collaborate in a shared space, which integrates the environments at these locations. In an archetypical tele-immersion environment as proposed at the origin of this project [8, 4], a user wears polarized glasses and a tracker capturing the head's pose. On a stereoscopic display, a remote scene is rendered so that it can be viewed from all potential viewpoints in the space of the viewer. To achieve this, an architecture that enables real-time view-independent 3D scene acquisition, transmission, and rendering in a real-time fashion is proposed (see Fig. 1). Most of the computational challenges are posed in the 3D scene acquisition. This stage deals with the association of pixels with the 3D coordinates of the world points they depict, in a view independent coordinate system. This association can be based on finding pixel correspondences between a pair of images. The derived correspondences constitute the basis of the computation of a disparity map, from which the depth of depicted world points and, in turn, their coordinates can be estimated.
The solution of the correspondence problem is associated with many challenging open topics, such as establishing correspondences for pixels that reside in textureless image regions, detecting occlusions, and coping with specular illumination effects.This involves a trade-off between being conservative and producing many holes in the depth or being lenient, covering everything with the cost of having outliers.

⋆ Contact person: Nikhil Kelshikar, nikhil@grasp.cis.upenn.edu, University of Pennsylvania, GRASP Laboratory, 3401 Walnut St., Philadelphia, PA 19104-6228. All three sites acknowledge financial support by the NSF grant IIS-0121293.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 33–42, 2003.
© Springer-Verlag Berlin Heidelberg 2003

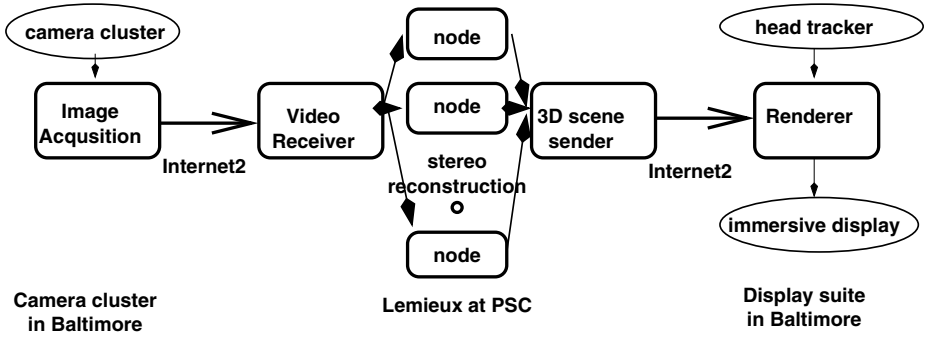


Fig. 1. System architecture. Images are acquired with a cluster of cameras, processed by a parallel computational engine to provide the 3D-description, transmitted, and displayed immersively.

As of summer 2001, we had achieved an 8Hz acquisition rate with limited depth quality, limited resolution and a very small operation space. Eventually, the original collaborative project [3] to produce a perceptually realistic tele-immersive environment, reached a level of maturity such that the remaining performance bottlenecks were well understood. We then established a goal to produce a real-time version with a dramatic increase in the volume of the scanned area, resolution, and depth accuracy. As opposed to other systems which capture one person inside a small area, we employed an array of stereo systems distributed so as they capture all actions from the span of all possible viewpoints of the remote viewer (wide-area scanning). Each stereo unit provides a $2 + \frac{1}{2}$D view as well as the correctly registered texture of this view.

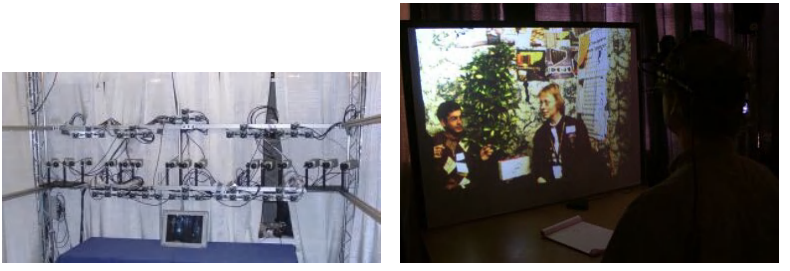


Fig. 2. 30 (of the 55 pictured) cameras were used in the November 2002 Supercomputing Conference demonstration (left). The acquired scene computed at PSC was displayed immersively (right).

To achieve this dramatic improvement in space and resolution, while maintain real-time performance, a significant increment of computational power is required, boosting the computational requirements to the supercomputing level. Such a resource was available at a remote location and, thus, we established one of the first applications where sensing, computation, and display are at different sites and coupled in real time. To overcome the transmission constraints, an initial implementation contains a video server transmitting video streams using TCP/IP and a reliable UDP transmission of the depth maps from the computation to the display site.

1.1 Computational and Networking Requirements

A single camera unit requires a triad of cameras to be grabbing pictures at 30 frames/second at VGA resolution. We use two monochromatic cameras (8 bits per pixel) and one color camera (8 bits per pixel; the color conversion is done in software). The images are captured at 640×480 pixel resolution. This produces data of 7.03 Mbits/frame. At the data capture rate of 30fps, we would produce data at 205.7 Mbits/sec. For each acquisition site we proposed to use a cluster of 10 or more camera units to adequately reconstruct the entire acquisition room, thus increasing data rates to 2.02 Gbits/sec per site. This is the amount of raw data that must make its way to the computing engine. The computing engine must execute $640 \times 480 \times 100$ depths $\times\ 31 \times 31$ kernel size $\approx 29.5G$ multiplications and additions per camera unit. The non-parallel version needed approximately $320 \times 240 \times 64$ depths $\times\ 5 \times 5$ kernel size $\approx 122M$ multiplications and additions per camera unit (actually twice as much because it used trinocular stereo). The produced 3D data stream is 11.72 Mbits/frame and consists of 16 bit inverse depth and a 24 bit color texture. This also scales as the number of camera units and users.

1.2 Related Work

Our real-time wide-area stereo scanning is not directly comparable to any other system in the literature. No other existing system combines viewpoint independent wide-area acquisition with spatially augmented displays and interaction. Though the explosion of Internet has produced many systems claiming tele-presence, none of them is working towards the scale necessary for tele-immersion. The closest to the proposed work is CMU's Virtualized Reality [7] and its early dedicated real-time parallel architecture for stereo. Other multi-camera systems include the view-dependent visual hull approach [5], the Keck laboratory at the University of Maryland [1] and the Argus system at Duke University [2].

2 Acquisition Algorithm

We now elaborate the main steps of the reconstruction algorithm [6] emphasizing the factors that affect the quality of reconstruction and the processing time. The initial implementation is based on two images but it is easily extensible to a polynocular configuration. We rely on the well-known stereo processing steps of matching and triangulation, given that the cameras are calibrated.

Rectification When a 3D-point is projected onto the left and the right image plane of a fixating stereo-rig the difference in the image positions is both in horizontal and vertical directions. Given a point in the first image we can reduce the 2D search to a 1D search if we know the so called *epipolar geometry* of the camera which is given from calibration. Because the subsequent step of correlation is area based, and for reduction of time complexity, we first perform a warping of the image that makes every epipolar line horizontal. This image transformation is called *rectification* and results in corresponding points having coordinates (u, v) and $(u - d, v)$, in left and right rectified images, respectively, where d is the horizontal disparity.

Matching: The degree of correspondence is measured by a modified normalized cross-correlation measure $c(I_L, I_R) = \frac{2\ cov(I_L, I_R)}{var(I_L) + var(I_R)}$., where I_L and I_R are the left and right rectified images over the selected correlation windows. For each pixel (u, v) in the left image, the matching produces a correlation profile $c(u, v, d)$ where d ranges over a disparity range. The definition domain is the so called *disparity range* and depends on the depth of *working volume*, i.e. the range of possible depths we want to reconstruct. The time complexity of matching is linearly proportional to the size of the correlation window as well as to the disparity range.
We consider *all* peaks of the correlation profile as possible disparity hypotheses. This is different from other matching approaches which early decide on the maximum of the matching criterion. We call the resulting list of hypotheses for all positions a *disparity volume*. The hypotheses in the disparity volume are pruned out by a *selection procedure* based on the constraints imposed by the following:

- Visibility: If a spatial point is visible then there can not be any other point in the viewing rays through this point and the left or right camera.
- Ordering: Depth ordering constrains the image positions in the rectified images. Both constraints can be formulated in terms of disparities without reconstructing the considered 3D-point.

The output of this procedure is an integer *disparity map*. To refine the 3-D position estimates, a *sub-pixel correction* of the integer disparity map is computed which results in a sub-pixel disparity map. To achieve fast sub-pixel estimation we fit a quadratic polynomial on the five-neighborhood of the integer disparity at the correlation maximum.

Reconstruction Each of the stereo rigs is calibrated before the experiment using a modification of Bouguet's camera calibration toolbox. Given estimates of the two, 3×4, projection matrices for the left and the right camera and the disparity at each point the coordinates of a 3D-point can be computed.

Color Image Warping The stereo cameras used to compute the 3D points are monochromatic cameras. A third color camera is used to color the 3D points. The calibration technique also estimates the projection matrix for the color camera. The projection matrix is used to compute a lookup table of where the 3D point lies in the color image. This lookup table is to map color to the 3D point set.

Depth Stream Next, the 3D depth and the color image must be sent to the remote viewer. Depth is encoded into a 3D stream which consists of a 16bit inverse depth image and a 24 bit RGB color image. This stream is then encoded in a raw byte format and transmitted over the network. The renderer also receives (once, during initialization) the inverse projection matrix for mapping the viewer coordinate system to the world coordinate system.
The error in the reconstruction depends on the error in the disparity and the error in the calibration matrices. Since the action to be reconstructed is close to the origin of the world coordinate system the depth error due to calibration is negligible compared to the error in the disparities. The principal concern is the number of outliers in the depth estimates which result in large peaks usually appearing near occlusion or texture-less areas.

3 Rendering

It is the task of the rendering system to take the multiple independent streams of 3D depth maps and re-create a life-size, view-dependent, stereo display of the acquired scene. Received depth maps are converted into 3D points and rendered as point clouds from a user tracked viewpoint. Multiple depth map streams are time synchronized and simply Z-buffered to create a composite display frame. Display is accomplished using a two-projector passive stereo configuration, and the user's eye positions are estimated with a HiBall wide-area tracker as described in [3].

While it is relatively easy to scale the video capture and reconstruction front-end, it is a most difficult task to architect a rendering system that is not the system bottleneck. At 640×480 resolution, each depth map stream arriving at the rendering system includes 11.73 Mbits per frame. Without data compression, 10 streams operating at 10 fps can generate a 1.3 Gbps data rate. At 80 per cent reconstruction efficiency (or approximately 250K points per 640×480 resolution stream), ten streams tend to produce 2.5M points to render requiring a point rendering performance of greater than 75M points/sec for truly smooth 30Hz, view-dependent rendering.

Our current system is architectured around a three-PC Linux cluster interconnected with a gigabit network. One PC serves as the network aggregation node for the multiple depth map streams arriving from the Terascale Computing System (TCS) platform. The other two PCs render the left and right-eye views. For performance reasons, data arriving into the aggregation node are multicast using the UDP protocol to the two rendering nodes. These PCs are 2.4GHz dual-processor Dell workstations and the rendering is facilitated by Nvidia GeForce 4 ti4600 cards. Using `Vertex_Array_Range` extensions rather that OpenGL displays, we are able to render 3D point clouds with up to 2M points at 30Hz. The 30Hz view-dependent display loop runs asynchronous to depth map stream update rate, which was limited to approximately 1Hz during the SC02 tests to avoid increasing latency and network buffering. Frames in each stream are time stamped so the render system can re-synchronize these independent streams. Frame swap synchronization between left and right-eye PCs is achieved with simple out-of-band protocols.

4 Terascale System/Parallel Porting Plan

The original development system consisted of a triple of cameras connected to a dual processor machine running at 2.4 GHz. There were five such camera triples connected to five different servers. Each server was used to grab three 320×240 pixel images. The acquired data was processed locally. To reduce the processing time, only the foreground was processed for stereo matching and reconstruction. The serial system used two processors per stream. Each of the processors processed half the image. The algorithm used small correlation kernels of 5×5 size. This system runs at 8 frames per second. The main bottleneck here is the processing hardware.

The quality of the reconstruction was not satisfactory and the real-time requirement precluded the use of any sophisticated algorithms. The images used were low resolution. The use of background subtraction in the images eliminated 66% of the data, hence the viewer could only see the remote participant in an artificial surrounding.

Complete reconstruction of the scene from more cameras using higher resolution images and more sophisticated algorithms, requires much more processing time. The real-

time constraint of this system required us to harness much more processing power. It became obvious that this serial based system would have to be migrated to a parallel platform. The platform chosen was the Terascale Computing System at the Pittsburgh Supercomputing Center. It comprises 3000 1GHz Alpha processors and is called Lemieux. The key parallelization insights were:

- The problem decomposes naturally by each camera stream.
- Serial image analysis code can remain fundamentally the same in the parallel implementation.
- Each processor would process a fraction of the image.

It was decided that a parallel framework, properly constructed, would allow the retention of the serial image analysis code and approach without sacrificing excellent scalability to thousands of processors. In addition, past parallel coding experience led to the incorporation of several other design goals:

- Define explicit parallelization interface to existing serial code.
- Anticipate the need for run-time debugging.
- Demand full scalability - allow no serial code to remain, introduce none.
- Permit partial stream asynchronicity during development, but demand fully asynchronous code when done.
- Design with physical I/O constraints in mind.

In addition, previous experience in development of real-time (defined as maximum allowable latency) codes on other Unix-like parallel platforms led us to anticipate that system services, and particularly ”heart beats” would be problematic. Code with maximum tolerance would be necessary.

5 Data Flow

The parallel framework that was developed was based on servicing each stream independently of others, of scaling within each stream and assuring that physical I/O points on the system were not allowed to become bottlenecks. The resulting schematic for a single stream looks as shown in Fig. 3, left. Data is received over the Internet into a designated Lemieux input node equipped with a 100Mb ethernet interface. It is then divided up into congruent horizontal bands for each of the three cameras and distributed to a variable number of computational nodes for image analysis. These analyzed images are then gathered to a single node, which combines them into a processed frame, and broadcast over the Internet to a remote display station. Note that the streams retain full independence in this schematic. This was maintained in software, even while retaining a coherent single executable, with use of MPI communicator mechanisms.

6 Performance

The switch to a terascale system allowed us to accommodate for more computationally intensive algorithms with higher quality images. We incorporated several changes notably changing the search area and the correlation kernel size. However, the core stereo matching algorithm was not changed, only the parameters used to operate it were tuned to increase match reliability.

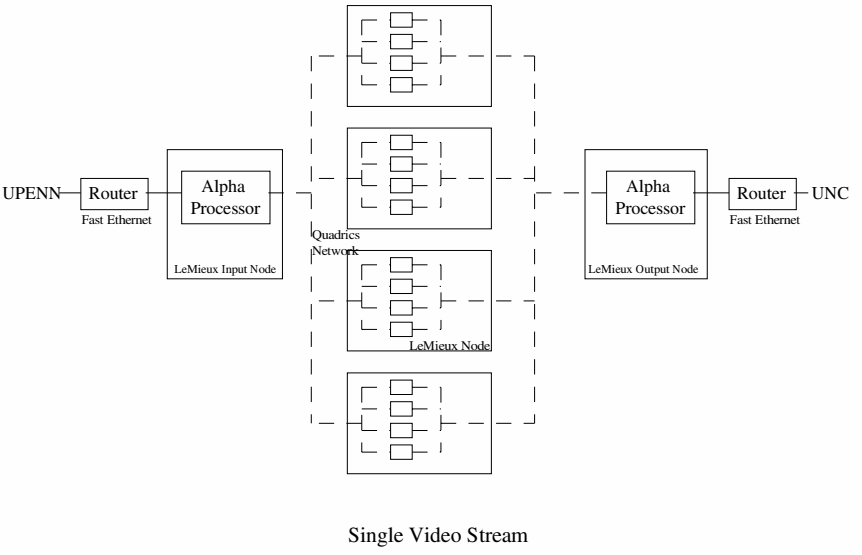


Fig. 3. Single stream data flow.

Computation The parallel system operates on images four times the size of those used in the serial algorithm. The correlation window size is 31×31 rather than 5×5 increasing computation approximately 36 times. However we use binocular instead of trinocular stereo due to which we have to perform matching once rather than performing pairwise matching. Thus, the new system requires at least 72 times more computation. Since we do not perform background subtraction, an additional order of magnitude of complexity is required.

The correlation window size is the main parameter affecting performance. The complexity of the stereo algorithm is $\mathcal{O}(m^2n^2)$ where m^2 is the size of the correlation kernel and n^2 is the size of the image. We ran a series of tests to verify the performance and the scalability of the system. The performance of the real-time system, with networked input of video and network output of 3D streams, is constrained by many external factors which could cause bottlenecks. Hence for performance analysis of the parallel algorithm we switched to file based I/O. The image streams are read from disk and we measure the time for image distribution on the cluster, image analysis and 3D data gathering from the various cluster nodes which contribute to total processing time.

The reconstruction algorithm broadcasts the image to be processed on a particular node in its entirety. Hence as the number of PE's used for the particular stream increases, so does the broadcast time, as shown in Fig. 4, left. Each processor would then perform stereo matching on a small strip of the entire image. This is the lowest level of parallelization. As the number of processors increases, each processor processes fewer pixels. Fig. 4 (right) shows the speedup in the process frame routine which performs

image rectification, stereo matching and the reconstruction of the 3D points. We show the processing time for seven different correlation window sizes. The reconstructed 3D

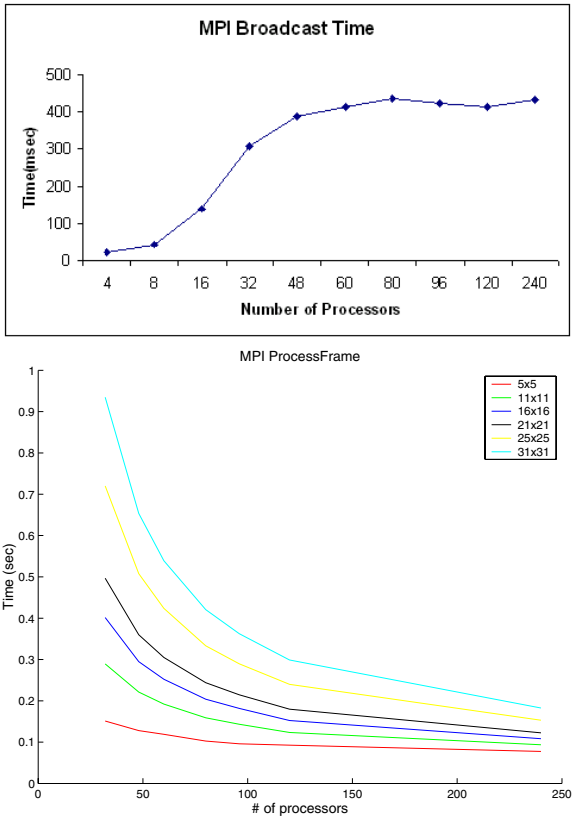


Fig. 4. Left: The time required to broadcast images to each node increases as the number of processors increases. Right: Total processing time in msec vs number of processors. Each plot corresponds to a different kernel size.

points have to be re-assembled as different parts of the images are reconstructed on different nodes. This gather operation speeds up with number of processors because less data must be gathered from each node.

Based on the above studies we have observed that the algorithm scales very efficiently with the number of processors per stream. The program parallelizes so that each stream is synchronized when reading data and runs independently of each other and hence does not affect individual performance. Each stream of images has similar parameters and hence execution time is almost the same.

Networking We conducted several networking and performance tests during the Supercomputing 2002 (SC2002) Conference in Baltimore, MD. The video servers and the rendering system on SC2002 exhibition floor communicated with the PSC Lemieux system over Abilene, the high-speed Internet2 backbone. The video servers and the

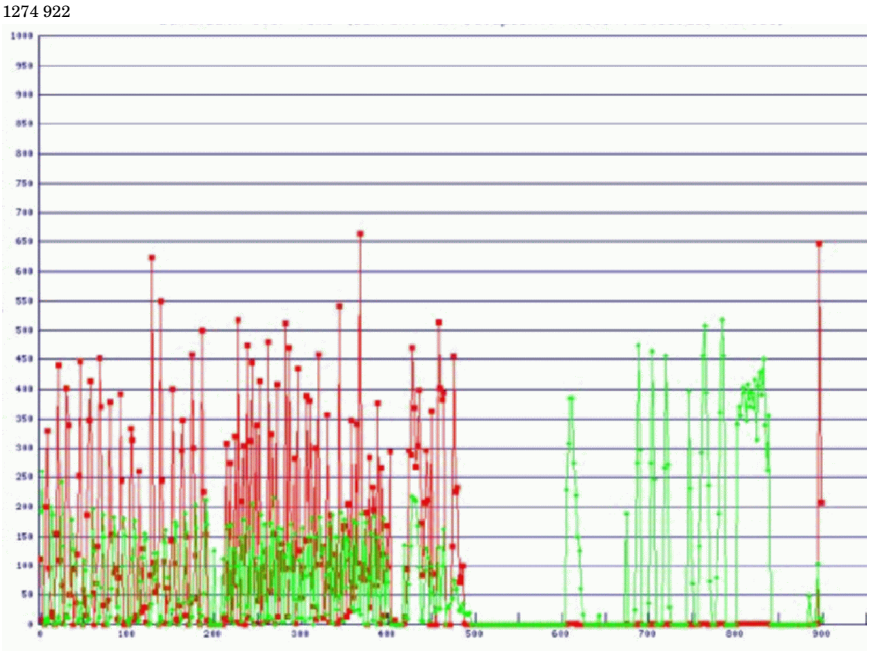


Fig. 5. Bandwidth usage plot from Nov 19 2002 SuperComputing 2002 demonstration, showing Mbps over time. The darker line (red) shows the data traffic from acquisition to computation site, while the lighter line)green) shows the traffic from computation to display site.

rendering system were each connected to Abilene by separate Gigabit links. Tests were performed with nine streams of images (from nine video servers.) The image data originating at video servers was transmitted to Lemieux using TCP. The data was processed on Lemieux and the depth/texture data sets were (optionally) sent back to the rendering system, using Reliable UDP (RUDP), a protocol specifically designed by us for this application. RUDP provides reliable data transmission required by the application without any congestion control, thereby providing better throughput than TCP.

In Fig. 5, shown is the bandwidth usage for the run during the "Bandwidth Challenge" at SC2002. The run consists of two phases. The first phase involved the entire system from image acquisition to rendering. An average frame rate of 1 frame/sec was achieved during this phase. The average data rate from video servers to Lemieux over TCP was 63 Mbps with peak rates over 200 Mbps. From Lemieux to the rendering system, RUDP was able to achieve data rates of over 105Mbps with peaks over 700Mbps (see the caption on the plot.) The frame rate achievable in this end-to-end operation was limited by the capacity of the rendering system to consume the depth/texture data. In the second phase, the rendering system was left out. The image data was processed at Lemieux as before, but it was not transmitted back to the rendering system. In this phase, the frame rate up to 8 frames/sec was achieved and the data rate over 500 Mbits/s was observed for the image transmission over TCP. The burstiness of the traffic is due to synchronous or lock-step operation of the system. During the entire run, 1080 processors (120 per

stream) were employed for the stereo reconstruction on Lemieux. The output of each processors are depth estimates for 4 rows of 640 pixels width.

7 Conclusions - Outlook

We have ported a real-time application from a dedicated serial environment to one that crosses a wide area network and utilizes a centralized parallel computing resource. The parallel code demonstrates excellent scalability and continues to exploit a friendly segmentation between the image analysis and parallel framework which allows concurrent work by all development groups. This nearly perfect scalability, both by PE per stream and by stream, makes us optimistic that we will be able to continue our overall performance gains by the three routes of:

- Better per processor performance through optimization
- Larger machine size runs on a routine basis
- Larger platforms that become available.

We are easily capable of saturating our current, routinely available networking connection. Imminent network enhancements will permit progress on that front. To improve reconstruction quality, we will employ an adaptive window size in order to preserve discontinuities as well as inter-scanline processing to alleviate rectification inaccuracies. In the quest for further performance improvements, we are actively investigating advanced rendering techniques including moving the conversion from depth maps (integer) to 3D points (floats) from the CPU into the graphics hardware. In the future, we would like to apply image compression techniques to reduce the data bandwidth requirements. We are also exploring the issue of latency and susceptibility to network congestion to develop a simple protocol that will minimize both, improving multi-stream throughput. The task of developing a rendering architecture that will scale as easily and linearly as independent camera-capture/reconstruction streams remains a significant research challenge.

References

1. P. Baker and Y. Aloimonos. Complete calibration of a multi-camera network. In *IEEE Workshop on Omnidirectional Vision*, Hilton Head Island, SC, June 12, 2000.
2. D.J. Brady, R.A. Stack, S. Feller, L. Fernandez E. Cull, D. Kammeyer, and R. Brady. Information flow in streaming 3d video. In *Three-Dimensional Video and Display Devices and Systems, SPIE PRESS Vol. CR76*, 2000.
3. H. Towles et al. 3d tele-collaboration over internet2. In *International Workshop on Immersive Telepresence, Juan-les-Pins, France, 06 Dec*, 2003.
4. J. Lanier. Virtually there. *Scientific American*, pages 66–75, April 2001.
5. W. Matusik, C. Buheler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of ACM SIGGRAPH*, 2000. to appear.
6. J. Mulligan, V. Isler, and K. Daniilidis. Trinocular stereo: A new algorithm and its evaluation. *International Journal of Computer Vision*, 47:51–61, 2002.
7. P. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. Int. Conf. on Computer Vision*, pages 3–10, 1998.
8. R. Raskar, G.Welch, M.Cutts, A.Lake, L.Stesin, and H.Fuchs. The office of the future: A unified approach to image- based modeling and spatially immersive displays. In *ACM SIGGRAPH*, pages 179–188, 1998.

Performance Analysis of PHASTA on NCSA Intel IA-64 Linux Cluster

Wai Yip Kwok

National Center for Supercomputing Applications, Champaign, IL 61820, USA
kwok@ncsa.uiuc.edu,
http://www.ncsa.uiuc.edu/∼kwok

Abstract. Performance of a computation-intensive multi-purpose CFD code PHASTA is analyzed on the NCSA Intel IA-64 Linux cluster. The capabilities of current-generation, open-source performance analysis tools available on this terascale system are demonstrated. Code profiling and hardware-performance counting tools are used to measure single-processor performance. Results pinpoint dominant but inefficient subroutines when level-3 optimization is used. Performance of these subroutines improves by compiling with level-2 optimization instead, due to reduction in total instructions. Flop rates of individual subroutines are estimated to guide further tuning. Parallel performance is addressed with performance visualization of inter-processor communication. Results reveal sporadic communication overhead in the function MPI_Waitall. This overhead constitutes about 18% of total simulation time.

1 Introduction

Cluster computers, which are commodity workstations connected by high-speed network, have emerged as a major source of terascale computing systems. New microprocessors such as the Intel IA-64 processors have much power to perform computing-intensive tasks. However, their cutting-edge technology also poses challenges to performance analysis and compiling.

In this study, we report success in using a variety of performance analysis tools to study the single-processor performance and multiple-processor communication of a computational fluid dynamics (CFD) code on the NCSA Intel IA-64 Linux cluster. The analysis provides vital information to code tuning. Section 2 describes the application code, NCSA Linux cluster, and performance analysis tools. Single-processor performance analysis is presented in Section 3. Multi-processor communication is discussed in Section 4. A summary is given in Section 5.

2 Preliminaries

This section provides basic information about the multi-purpose CFD code, the computer architecture, and the tools used to analyse performance of the code.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 43–52, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2.1 Application Code

Parallel Hierarchic Adaptive Stabilized Transient Analysis, or PHASTA software is developed and supported by the Scientific Computation Research Center at the Rensselaer Polytechnic Institute [5]. It can model three-dimensional, compressible or incompressible, laminar or turbulent, steady or unsteady flows, using unstructured grids. A stabilized finite-element formulation for fluid dynamics using mesh-entity hierarchical basis functions is used in the software.

PHASTA consists of more than 150 codes. Most of the codes are written in Fortran 90. These Fortran 90 subroutines carry out all the computation. A small number of codes are written in C and C++. Message passing interface (MPI) is used for data communication when multiple processors are used.

2.2 NCSA Intel IA-64 Linux Cluster

The NCSA Intel IA-64 Linux cluster is a distributed-memory system [3]. It is based on IBM IntelliStation Z Pro 6894 workstation. Each workstation (node) has two 800 MHz Intel Itanium I processors. There are more than 120 nodes. The cluster runs Red Hat Linux version 7.1 and kernel 2.4.16, and uses both Myrinet interconnect and Gigabit Ethernet network.

2.3 Performance Analysis Tools

A variety of tools are used to analyze the performance of PHASTA. GNU gprof and VProf are used for code profiling. GNU gprof is written by Jay Fenlason [2]. It can generate flat profiles, which show how much time is spent in each function, and how many times that function is called. VProf is developed by Curtis Janssen in Sandia National Laboratories [6]. It carries out statistical profiling for hardware-performance events. On the NCSA Itanium Linux cluster, VProf makes use of the Performance Application Programming Interface (PAPI) [4] to access the hardware-performance monitors. PAPI is developed by the Innovative Computing Laboratory at the University of Tennessee. VProf requires Linux kernel version 2.4.19 and PAPI version 2.3.1 to function properly. A node on the NCSA IA-64 cluster is upgraded to this kernel to enable the VProf analysis.

Overall code statistics are gathered with a tool named psrun. The tool psrun [8] is a command-line utility used to gather hardware-performance information on an unmodified executable. It achieves counting of multiple hardware-performance events (multiplexing) through PAPI. It is developed by Rick Kufrin at NCSA.

MPI communication among multiple processors are monitored and visualized with MultiProcessing Environment (MPE) libraries and Upshot [7] developed by Argonne National Laboratory. MPE and Upshot provide users with profiling and visualization tools for their MPI programs.

3 Single-Processor Performance Analysis and Tuning

After the code is ported to the IA-64 Linux cluster and completes some test problems correctly, its single-processor performance is analyzed. To validate the accuracy of profiling tools on the new IA- 64 architecture, both GNU gprof and VProf are used to ensure consistency. PHASTA, when compiled with level-3 optimization, produces profiles shown in Table 1. Profiles measured with GNU gprof and VProf are consistent. The functions e3ls, e3conv and e3wmlt occupy a large portion ($> 60\%$) of the total time consumed. This finding directs optimization efforts to these dominant subroutines.

Table 1. Time-profiles of PHASTA before tuning. All numbers are in percentage

Subroutine name	Percentage of time spent, measured with GNU gprof	Percentage of time spent, measured with VProf
e3ls	33.25	36.10
e3conv	17.43	16.50
e3wmlt	11.99	13.00
fillsparsec	5.61	5.90
e3ivar	4.03	3.00
e3massl	4.02	3.60
sparseap	3.63	3.20
e3mtrx	3.15	2.60
e3visc	1.96	1.30
e3	1.81	1.80
Others	13.12	13.00

When compilation of functions e3ls and e3conv is reviewed, the following warning message is noticed: 'Space exceeded in Data Dependence Test in e3ls (or e3conv). Subdivide routine into smaller ones to avoid optimization loss.' This warning appears regardless of size of grid, block, dataset, etc, as all the above information is fed into the code at the execution rather than the compilation stage. The compilation process of the whole code is then examined and it is found out that the following files induce the above warning: 1. asithf.f, 2. bardmc.f, 3. e3bdg_nd.f, 4. e3bdg.f, 5. e3bvar.f, 6. e3conv.f, 7. e3dc.f, 8. e3ls.f, 9. e3mtrx.f, 10. e3tau.f.

Various compiler flags and loop adjustment are explored to remove the cumbrance in the above codes. It is found at the end that the same subroutines, compiled with level-2 optimization, surprisingly run faster than their level-3 optimized counterparts. The main difference between level-3 and level-2 optimizations is its additional, more aggressive optimization. These optimization procedures, also named high-level language optimizations, include prefetching, scalar replacement, loop transformation, floating-point division optimization and

more data-dependency analysis [1]. The additional optimization affects memory access, instruction parallelism, predication, and software-pipelining. The resulting effects can be beneficial or harmful, depending on the particular code or loop. In the current situation, additional optimization ends up harmful.

As a result, the subroutines that fail the data-dependence test are compiled with level-2 optimization, while the rest stays with level-3 optimization. It should be noted that in subsequent paragraphs level-2 optimization refers to this mixed level-2 and level-3 optimization.

After the change in optimization level, wall-clock time spent in e3ls and e3conv reduces by 48% and 34%, respectively (Figure 1). In terms of percentage of time spent in individual functions, e3ls and e3conv consumes 22% and 15%, respectively, after tuning, down from 33% and 17% before tuning (Table 2).

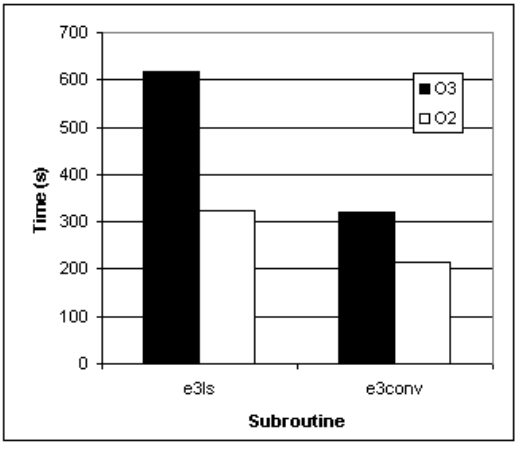


Fig. 1. Wall-clock time spent in subroutines e3ls and e3conv, compiled with level-3 (denoted O3) and level-2 (denoted O2), respectively. Wall-clock time is measured with GNU gprof

Profiles of hardware-performance events provide more information to understand the performance improvement. VProf collects profiling data on various hardware-performance events down to the line level. In this analysis, profile information is collected for the following events:

1. Cycles waiting for memory access (PAPI_MEM_SCY)
2. L3 cache misses (PAPI_L3_TCM)
3. L2 cache misses (PAPI_L2_TCM)
4. Total instructions (PAPI_TOT_INS)
5. Load/store instructions (PAPI_LST_INS)
6. Floating-point instructions (PAPI_FP_INS)
7. Total cycles (PAPI_TOT_CYC)

Table 2. Time-profiles of PHASTA using level-3 and level-2 optimizations, measured with GNU gprof. All numbers are in percentage

Subroutine	Level-3 optimized code	Level-2 optimized code
e3ls	33.25	22.30
e3conv	17.43	15.03
e3wmlt	11.99	15.27
fillsparsec	5.61	8.86
e3ivar	4.03	4.63
e3massl	4.02	4.89
sparseap	3.63	4.74
e3mtrx	3.15	3.36
e3visc	1.96	2.30
e3	1.81	2.30
Others	13.12	16.32

8. Time profile (PROF).

Table 3 shows the profiles for level-3 optimized code. In addition to the percentage profile, more interesting is the actual event counts in various subroutines. Consider a hardware-performance event in a subroutine s_i, the event count N_{s_i} can be obtained by multiplying the event count of the whole code (N_T) with the percentage of event count spent in that subroutine p_{s_i}, as indicated in Equation (1).

$$N_{s_i} = N_T \times p_{s_i}. \tag{1}$$

The total counts of hardware-performance events (N_T's) are measured with psrun, while the distribution of hardware-performance events among subroutines (p_{s_i}) is recorded by VProf (Table 3 for example). Table 4, which shows the event counts in individual subroutines for the level-3 optimized code, is thus derived from Table 3 and the results from psrun. Table 5 lists the same quantities for the level-2 optimized code.

Tables 4 and 5 reveal the distribution of instruction types. Significant portions of instructions are floating-point and load/store. A considerable percentage of instructions cannot be measured with PAPI in the case of level-3 optimization. Floating-point and load/store instructions carried out by various subroutines change slightly by switching optimization level. However, there is a drastic reduction in unclassified instructions carried out by subroutines e3ls and e3conv after tuning. This drop contributes substantially to the performance improvement. Performance tool pfmon pinpoints those unclassified instructions as instructions with no operations (NOPS_RETIRED). L3 cache misses in subroutines e3ls and e3conv also decrease slightly.

Flop rates of the dominant subroutines are shown in Figure 2 for the level-2 optimized code. It reveals that subroutines e3ls and e3conv behave above average. Further tuning should focus on subroutines e3wmlt and fillsparsec, which occupy 18% and 8% of the time spent and perform relatively poorly.

Table 3. Profile of hardware-performance events in level-3 optimized PHASTA, measured with VProf. All numbers are in percentage

Subroutine	MEM_ SCY	L3_ TCM	L2_ TCM	TOT_ INS	LST_ INS	FP_ INS	TOT_ CYC	PROF
e3ls	30.9	22.4	30.2	25.6	33.3	41.8	41.7	36.1
e3conv	15.4	15.0	14.9	13.6	12.7	12.3	18.1	16.5
e3wmlt	19.7	11.1	26.1	21.5	25.6	24.4	12.8	13.0
fillsparsec	3.1	2.6	0.8	2.8	1.9	0.9	2.3	5.9
e3massl	3.3	2.2	5.0	8.1	8.2	1.7	4.1	3.6
sparseap	3.4	14.4	4.4	1.9	2.0	0.9	0.8	3.2
e3ivar	2.4	4.9	2.6	3.5	1.5	2.7	3.5	3.0
e3mtrx	3.3	4.2	4.8	2.7	2.0	1.9	2.6	2.6
e3	2.6	1.7	1.1	1.2	2.0	0.0	0.9	1.8
e3metric	1.3	2.4	1.0	2.5	2.9	2.7	1.6	1.5
others	14.6	19.1	9.1	16.6	7.9	10.7	11.6	12.8

Table 4. Counts of occurrence of hardware performance events in level-3 optimized PHASTA, estimated using psrun and VProf. All numbers are in 10^9, except for FLOP rate

Subroutine	MEM_ SCY	L3_ TCM	L2_ TCM	TOT_ INS	LST_ INS	FP_ INS	TOT_ CYC	FLOP rate (mflops)
e3ls	207	1.26	22.53	403	170	133	703	150
e3conv	103	0.84	11.12	214	65	39	305	102
e3wmlt	132	0.62	19.47	338	131	78	216	286
fillsparsec	21	0.15	0.60	44	10	3	39	59
e3massl	22	0.12	3.73	127	42	5	69	62
sparseap	23	0.81	3.28	30	10	3	13	169
e3ivar	16	0.28	1.94	55	8	9	59	116
e3mtrx	22	0.24	3.58	42	10	6	44	110
e3	17	0.10	0.82	19	10	0	15	0
e3metric	9	0.13	0.75	39	15	9	27	253
others	98	1.07	6.79	261	40	34	196	138
Total	671	5.619	74.601	1574	512	318	1687	150

Table 5. Counts of occurrence of hardware performance events in level-2 optimized PHASTA, estimated using psrun and VProf. All numbers are in 10^9, except for FLOP rate

Subroutine	MEM_ SCY	L3_ TCM	L2_ TCM	TOT_ INS	LST_ INS	FP_ INS	TOT_ CYC	FLOP rate (mflops)
e3ls	187	0.99	21.06	246	153	136	245	440
e3conv	81	0.58	11.63	79	49	48	91	415
e3wmlt	162	0.76	19.39	325	126	78	281	220
fillsparsec	27	0.18	0.76	42	10	3	53	45
e3massl	28	0.11	4.11	122	44	6	92	49
sparseap	30	1.03	3.19	29	11	3	27	88
e3ivar	19	0.31	1.75	53	9	8	72	89
e3mtrx	28	0.19	6.23	43	13	7	53	100
e3	22	0.11	0.76	17	11	0	22	0
e3metric	12	0.15	0.68	37	13	9	38	180
others	124	1.20	6.46	247	44	37	258	113
Total	720	4.51	76.03	1240	484	334	1233	215

Regarding overall performance, PHASTA runs at 220 Mflops (CPU cycles) and 170 Mflops (effective). Compared with the NCSA SGI Origin2000 supercomputer, PHASTA performs 2.4 times faster on the Itanium cluster on a single processor.

4 Multi-processor Communication Analysis

Message Passing Interface (MPI) is implemented in PHASTA for parallel computing. Communication among processors is monitored with Multi-Processing Environment (MPE) libraries. Several multi-processor simulations are run to evaluate the inter-processor communication. Each simulation uses 4 processors and contains 20 iterations. The log files are viewed using the logfile viewer Upshot, and the visualization is shown in Figure 3. Heavy communication is indicated by bundles of black arrows. In each simulation, twenty bundles can be counted, corresponding to the twenty iterations. The time required per iteration is not even. The simulation alternates between long and short time-steps. A long and short time-step takes about 8 and 2 seconds, respectively. The reason for the variation is the re-use of matrix every other step.

It is noteworthy that there are several 'extra-long' time-steps that are marked by rough MPI communication and idleness in processors due to 'MPI_Waitall'. This communication overhead appears at different iterations in simulations carrying out the same calculation, suggesting that the code is not responsible for the peculiar delay. One likely cause is congestion in message communication through the Myrinet switches. Due to synchronization of the code, a large wait-all time on one processor affects all the other processors.

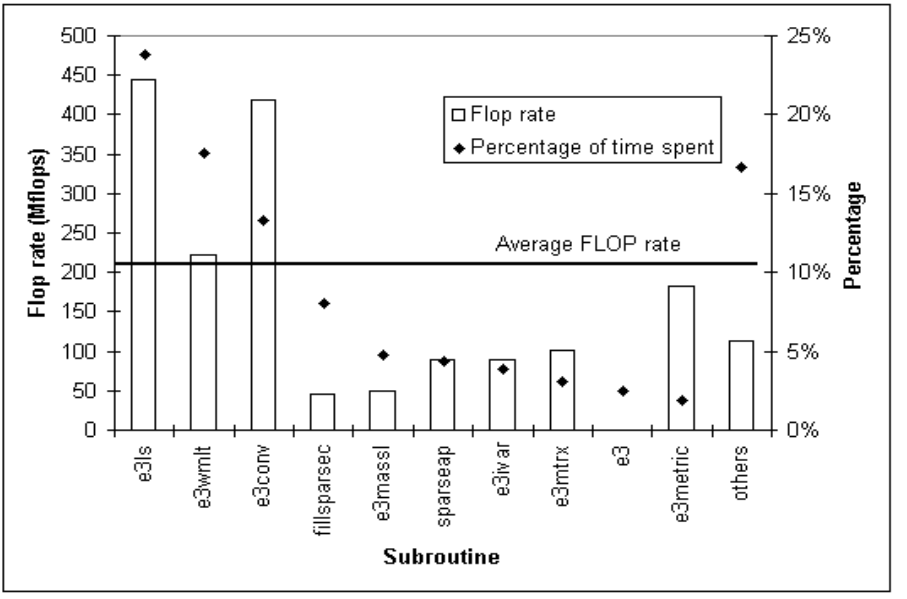


Fig. 2. Flop rate and time profile of dominant subroutines in level-2 optimized PHASTA, estimated using VProf and psrun

An attempt is made to estimate the adverse effect of the congestion in message communication quantitatively. The communication overhead depends on two factors: 1. frequency at which the congestion occurs; 2. amount of time delay the congestion induces. Five tests with 100 iterations each are run to gather relevant statistics. Time-steps are divided into three categories: short time-steps with duration less than 5 seconds, long time-steps with duration between 5 and 10 seconds, and extra-long time-steps that last more than 10 seconds due to communication overhead. The statistics of all five simulations is shown in Table 6. The percentage of time spent in communication overhead s_{oh} is estimated with Equation (2).

$$\begin{aligned} s_{oh} &= N_{el} \times \frac{\tau_{el} - \tau_l}{N_s \times \tau_s + N_l \times \tau_l + N_{el} \times \tau_{el}} \\ &= 83 \times \frac{13.9 - 7.68}{244 \times 176 + 173 \times 7.68 + 83 \times 13.9} \\ &= 18\%. \end{aligned} \tag{2}$$

As a result, a speed-up of about 3.3 instead of an ideal 4 is obtained when 4 processors are used.

A nice feature of Upshot is that it allows one to change the resolution to a much finer level for deeper understanding of data communication. This study is

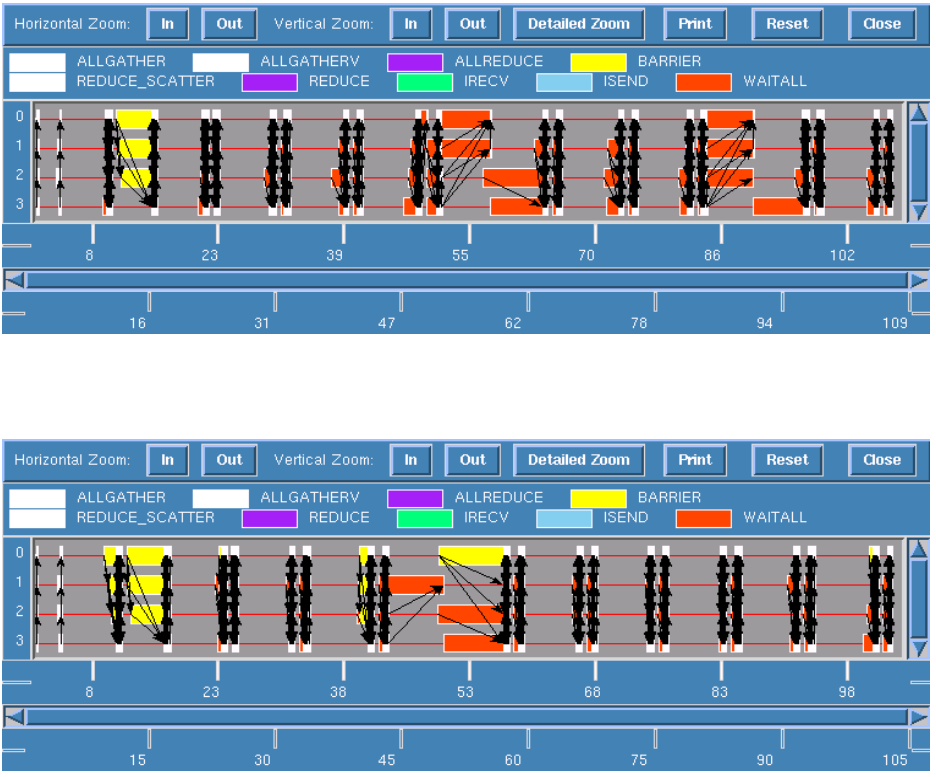


Fig. 3. MPI communication among 4 processors in 2 test simulations, visualized with Upshot. Each simulation contains 20 iterations. Time from 0 to about 100 seconds is represented in the x-axis. The numbers from 0 to 3 on the y-axis are the processor indices

Table 6. Duration of time-steps of five PHASTA benchmark simulations

Simulation	1	2	3	4	5	Total
Number of short time-steps, N_s	49	49	49	49	48	244
Number of long time-steps, N_l	33	30	34	42	34	173
Number of extra-long time-steps, N_{el}	18	21	17	9	18	83
Average length of a short time-step, τ_s (seconds)				1.66		
Average length of a long time-step, τ_l (seconds)				7.68		
Average length of an extra-long time-step, τ_{el} (seconds)				13.9		

conducted on some sample time-steps but not covered in this paper due to limit in space.

A direction of further investigation is an analysis of PHASTA on a larger machine configuration. This analysis requires PHASTA researchers to custom-build the mesh and input conditions, and is currently being pursued.

5 Summary

Single-processor performance and multi-processor communication of a CFD code PHASTA on the NCSA Intel IA-64 Linux cluster are studied. Code profiling and study of compilation process suggest that level-3 optimization hinders performance of some dominant subroutines. Changing optimization to level-2 for the affected subroutines improves performance of PHASTA. Hardware-performance events such as various instructions and cache misses are counted at the subroutine level. Results reveal that the change in optimization level leads to a drop in instructions with no operations (NOPS_RETIRED). Inter-processor communication is visualized with MPE libraries and Upshot. Results show that communication overhead occurs sporadically with the subroutine MPI_Waitall. This delay will be further investigated.

Acknowledgements.
The author would like to thank Kenneth Jansen and Anil Karanam at Rensselaer Polytechnic Institute for explaining the functions of PHASTA. Rick Kufrin and Greg Bauer at NCSA provided vital help in the use of performance analysis tools. The author also had constructive discussions with Faisal Saied on various code optimization issues.

References

1. Intel Fortran compiler user's guide. 2002.
2. GNU gprof homepage. http://www.gnu.org/manual/gprof-2.9.1/gprof.html. 2003.
3. NCSA homepage. http://www.ncsa.uiuc.edu. 2003.
4. PAPI homepage. http://icl.cs.utk.edu/projects/papi.
5. Scientific Computation Research Center homepage. http://www.scorec.rpi.edu. 2003.
6. VProf homepage. http://aros.ca.sandia.gov/~cljanss/perf/vprof.
7. MPE libraries and Upshot homepage. http://www-unix.mcs.anl.gov/perfvis/software/viewers.
8. psrun homepage. http://perfsuite.ncsa.uiuc.edu/psrun.

Performance Instrumentation and Measurement for Terascale Systems

Jack Dongarra[1], Allen D. Malony[2], Shirley Moore[1],
Philip Mucci[2], and Sameer Shende[2]

[1] Innovative Computing Laboratory, University of Tennessee
Knoxville, TN 37996-3450 USA
{dongarra,shirley,mucci}@cs.utk.edu
[2] Computer Science Department, University of Oregon
Eugene, OR 97403-1202 USA
{malony,sameer}@cs.uoregon.edu

Abstract. As computer systems grow in size and complexity, tool support is needed to facilitate the efficient mapping of large-scale applications onto these systems. To help achieve this mapping, performance analysis tools must provide robust performance observation capabilities at all levels of the system, as well as map low-level behavior to high-level program constructs. Instrumentation and measurement strategies, developed over the last several years, must evolve together with performance analysis infrastructure to address the challenges of new scalable parallel systems.

1 Introduction

Performance observation requirements for terascale systems are determined by the performance problem being addressed and the performance evaluation methodology being applied. Instrumentation of an application is necessary to capture performance data. Instrumentation may be inserted at various stages, from source code modifications to compile-time to link-time to modification of executable code either statically or dynamically during program execution. These instrumentation points have different mechanisms which vary in their ease of use, flexibility, level of detail, user control of what data can be collected, and intrusiveness.

To provide insight into program behavior on large-scale systems and point the way toward program transformations that will improve performance, various performance data must be collected. Profiling data show the distribution of a metric across source-level constructs, such as routines, loops, and basic blocks. Most modern microprocessors provide a rich set of hardware counters that capture cycle count, functional unit, memory, and operating system events. Profiling can be based one either time or various hardware-based metrics, such as cache misses, for example. Correlations between profiles based on different events, as well as event-based ratios, provide derived information that can help to quickly identify and diagnose performance problems. In addition to profiling data, capturing event traces of program events, such as message communication events, helps portray the temporal dynamics of application performance.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 53–62, 2003.
© Springer-Verlag Berlin Heidelberg 2003

For terascale systems, a wide range of performance problems, performance evaluation methods, and programming environments need to be supported. A flexible and extensible performance observation framework can best provide the necessary flexibility in experiment design. Research problems to be addressed by the framework include the following: the appropriate level and location in the framework for implementing different instrumentation and measurement strategies, how to make the framework modular and extensible, and the appropriate compromise between the level of detail and accuracy of the performance data collected and the instrumentation cost.

The remainder of the paper is organized as follows. Section 2 describes the instrumentation mechanisms desirable to support in such a framework and Section 3 describes types of measurements. Section 4 explains how the instrumentation and measurement strategies are supported in the PAPI cross-platform hardware counter interface and in the TAU performance observation framework. Section 5 presents our conclusions.

2 Instrumentation

To observe application performance, additional instructions or probes are typically inserted into a program. This process is called *instrumentation*. Instrumentation can be inserted at various stages, as described below.

2.1 Source Code Instrumentation

Instrumentation at the source code level allows the programmer to communicate higher-level domain-specific abstractions to the performance tool. A programmer can communicate such events by annotating the source code at appropriate locations with instrumentation calls. Once the program undergoes a series of transformations to generate the executable code, specifying arbitrary points in the code for instrumentation and understanding program semantics at those points may not be possible. Another advantage of source code instrumentation is that once an instrumentation library targets one language, it can provide portability across multiple compilers for that language, as well as across multiple platforms. Drawbacks of source code instrumentation include possible changes in instruction and data cache behavior, interactions with optimizing compilers, and runtime overhead of instrumentation library calls.

Source code annotations can be inserted manually or automatically. Adding instrumentation calls in the source code manually can be a tedious task that introduces the possibility of instrumentation errors producing erroneous performance data. Some of these difficulties with manual source code instrumentation can be overcome by using a source-to-source preprocessor to build an automatic instrumentation tool. Tools such as Program Database Toolkit (PDT) [10] for C++, C and Fortran 90, can be used to automatically instrument subroutines, code regions, and statements.

2.2 Library Level Instrumentation

Wrapper interposition libraries provide a convenient mechanism for adding instrumentation calls to libraries. For instance, the MPI Profiling Interface [1] allows a tool developer to interface with MPI calls in a portable manner without modifying the application

source code or having access to the proprietary source code of the library implementation. The advantage of library instrumentation is that it is relatively easy to enable and the events generated are closely associated with the semantics of the library routines.

2.3 Binary Instrumentation

Executable images can be instrumented using binary code-rewriting techniques, often referred to as binary editing tools or executable editing tools. Systems such as Pixie, ATOM [5], EEL [9], and PAT [6] include an object code instrumentor that parses an executable and rewrites it with added instrumentation code. The advantage of binary instrumentation is that there is no need to re-compile an application program and rewriting a binary file is mostly independent of the programming language. Also, it is possible to spawn the instrumented parallel program the same way as the original program, without any special modification as are required for runtime instrumentation [12]. Furthermore, since an executable program is instrumented, compiler optimizations do not change or invalidate the performance optimization.

2.4 Dynamic Instrumentation

Dynamic instrumentation is a mechanism for runtime code patching that modifies a program during execution. DyninstAPI [3] provides an efficient, low-overhead interface that is suitable for performance instrumentation. A tool that uses this API is called a *mutator* and can insert code snippets into a running program, which is called the *mutatee*, without re-compiling, re-linking, or event re-starting the program. The mutator can either spawn an executable and instrument it prior to its execution, or attach to a running program. Dynamic instrumentation overcomes some limitations of binary instrumentation by allowing instrumentation code to be added and removed at runtime. Also, the instrumentation can be done on a running program instead of requiring the user to re-execute the application. The disadvantage of dynamic instrumentation is that the interface needs to be aware of multiple object file formats, binary interfaces (32/64 bit), operating system idiosyncrasies, as well as compiler specific information (e.g., to support template name de-mangling in C++ from multiple C++ compilers). To maintain cross language, cross platform, cross file format, cross binary interface portability is a challenging task and requires a continuous porting effort as new computing platforms and multi-threaded programming environments evolve.

3 Types of Measurements

Decisions about instrumentation are concerned with the number and type of performance events one wants to observe during an application's execution. Measurement decisions address the types and amount of performance data needed for performance problem solving. Often these decisions involve tradeoffs of the need for performance data versus the cost of obtaining it (i.e., the measurement overhead). Post-mortem performance evaluation tools typically fall into two categories: profiling and tracing, although some provide both capabilities. More recently, some tools provide real-time, rather than post-mortem, performance monitoring.

3.1 Profiling

Profiling characterizes the behavior of an application in terms of aggregate performance metrics. Profiles are typically represented as a list of various metrics (such as inclusive/exclusive wall-clock time) that are associated with program-level semantics entities (such as routines, basic blocks, or statements in the program). Time is a common metric, but any monotonically increasing resource function can be used, such as counts from hardware performance counters. Profiling can be implemented by sampling or instrumentation based approaches.

3.2 Tracing

While profiling is used to get aggregate summaries of metrics in a compact form, it cannot highlight the time varying aspects of the execution. To study the post-mortem spatial and temporal aspects of performance data, event tracing, that is, the activity of capturing events or actions that take place during program execution, is more appropriate. Event tracing usually results in a log of the events that characterize the execution. Each event in the log is an ordered tuple typically containing a time stamp, a location (e.g., node, thread) an identifier that specifies the type of event (e.g., routine transition, user-defined event, message communication, etc.) and event-specific information. For a parallel execution, trace information generated on different processors may be merged to produce a single trace file. The merging is usually based on the timestamp which can reflect logical time or physical time.

3.3 Real-Time Performance Monitoring

Post-mortem analysis of profiling data or trace files has the disadvantage that analysis cannot begin until after program execution has finished. Real-time performance monitoring allows users to evaluate program performance during execution. Real-time performance monitoring is sometimes coupled with application performance steering.

4 PAPI and TAU Instrumentation and Measurement Strategies

Understanding the performance of parallel systems is a complicated task because of the different performance levels involved and the need to associate performance information to the programming and problem abstractions used by application developers. Terascale systems do not make this task any easier. We need to develop performance analysis strategies and technique that are successful both in their accessibility to users and in their robust application. In this section, we describe our efforts in evolving the PAPI and TAU technologies to terscale use.

4.1 PAPI

Most modern microprocessors provide hardware support for collecting hardware performance counter data [2]. Performance monitoring hardware usually consists of a set

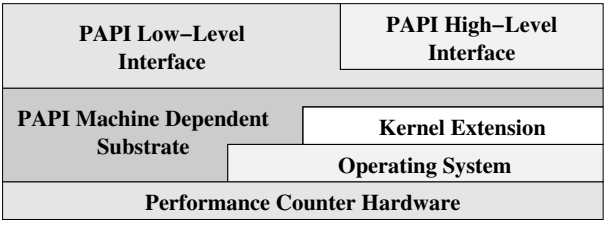


Fig. 1. Layered architecture of the PAPI implementation

of registers that record data about the processor's function. These registers range from simple event counters to more sophisticated hardware for recording data such as data and instruction addresses for an event, and pipeline or memory latencies for an instruction. Monitoring hardware events facilitates correlation between the structure of an application's source/object code and the efficiency of the mapping of that code to the underlying architecture.

Because of the wide range of performance monitoring hardware available on different processors and the different platform-dependent interfaces for accessing this hardware, the PAPI project was started with the goal of providing a standard cross-platform interface for accessing hardware performance counters [2]. PAPI proposes a standard set of library routines for accessing the counters as well as a standard set of events to be measured. The library interface consists of a high-level and a low-level interface. The high-level interface provides a simple set of routines for starting, reading, and stopping the counters for a specified list of events. The fully programmable low-level interface provides additional features and options and is intended for tool or application developers with more sophisticated needs.

The architecture of PAPI is shown in Figure 1. The goal of the PAPI project is to provide a firm foundation that supports the instrumentation and measurement strategies described in the preceding sections and that supports development of end-user performance analysis tools for the full range of high-performance architectures and parallel programming models. For manual and preprocessor source code instrumentation, PAPI provides the high-level and low-level routines described above. The `PAPI_flops` call is an easy-to-use routine that provides timing data and the floating point operation count for the bracketed code. The low-level routines target the more detailed information and full range of options needed by tool developers. For example the `PAPI_profil` call implements SVR4-compatible code profiling based on any hardware counter metric. Again, the code to be profiled need only be bracketed by calls to the `PAPI_profil` routine. This routine can be used by end-user tools such as VProf [1] to collect profiling data which can then be correlated with application source code.

Reference implementations of PAPI are available for a number of platforms (e.g., Cray T3E, SGI IRIX, IBM AIX Power, Sun Ultrasparc Solaris, Linux/x86, Linux/IA-64, HP/Compaq Alpha Tru64 Unix). The implementation for a given platform attempts to map as many of the standard PAPI events as possible to the available platform-

[1] `http:/aros.ca.sandia.gov/čljanss/perf/vprof/`

specific events. The implementation also attempts to use available hardware and operating system support – e.g., for counter multiplexing, interrupt on counter overflow, and statistical profiling.

Using PAPI on large-scale application codes, such as the EVH1 hydrodynamics code, has raised issues of scalability of the instrumentation. PAPI initially focused on obtaining aggregate counts of hardware events. However, the overhead of library calls to read the hardware counters can be excessive if the routines are called frequently – for example, on entry and exit of a small subroutine or basic block within a tight loop. Unacceptable overhead has caused some tool developers to reduce the number of calls through statistical sampling techniques. On most platforms, the current PAPI code implements statistical profiling over aggregate counting by generating an interrupt on counter overflow of a threshold and sampling the program counter. On out-of-order processors the program counter may yield an address that is several instructions or even basic blocks removed from the true address of the instruction that caused the overflow event. The PAPI project is investigating hardware support for sampling, so that tool developers can be relieved of this burden and maximum accuracy can be achieved with minimal overhead. With hardware sampling, an in-flight instruction is selected at random and information about its state is recorded – for example, the type of instruction, its address, whether it has incurred a cache or TLB miss, various pipeline and/or memory latencies incurred. The sampling results provide a histogram of the profiling data which correlates event frequencies with program locations.

In addition, aggregate event counts can be estimated from sampling data with lower overhead than direct counting. For example, the new PAPI substrate for the HP/Compaq Alpha Tru64 UNIX platform is built on top of a programming interface to DCPI called DADD (Dynamic Access to DCPI Data). DCPI identifies the exact address of an instruction, thus resulting in accurate text addresses for profiling data [4]. Test runs of the PAPI `calibrate` utility on the substrate have shown that event counts converge to the expected value, given a long enough run time to obtain sufficient samples, while incurring only one to two percent overhead, as compared to up to 30 percent on other substrates that use direct counting. A similar capability exists on the Itanium and Itanium 2 platforms, where Event Address Registers (EARs) accurately identify the instruction and data addresses for some events. Future versions of PAPI will make use of such hardware assisted profiling and will provide an option for estimating aggregate counts from sampling data.

The `dynaprof` tool that is part of the most recent PAPI release uses dynamic instrumentation to allow the user to either load an executable or attach to a running executable and then dynamically insert instrumentation probes [11]. Dynaprof uses Dyninst API [3] on Linux/IA-32, SGI IRIX, and Sun Solaris platforms, and DPCL [2] on IBM AIX. The user can list the internal structure of the application in order to select instrumentation points. Dynaprof inserts instrumentation in the form of *probes*. Dynaprof provides a PAPI probe for collecting hardware counter data and a wallclock probe for measuring elapsed time, both on a per-thread basis. Users may optionally write their own probes. A probe may use whatever output format is appropriate, for example a real-time data feed to a visualization tool or a static data file dumped to

[2] `http://oss.software.ibm.com/developerworks/opensource/dpcl/`

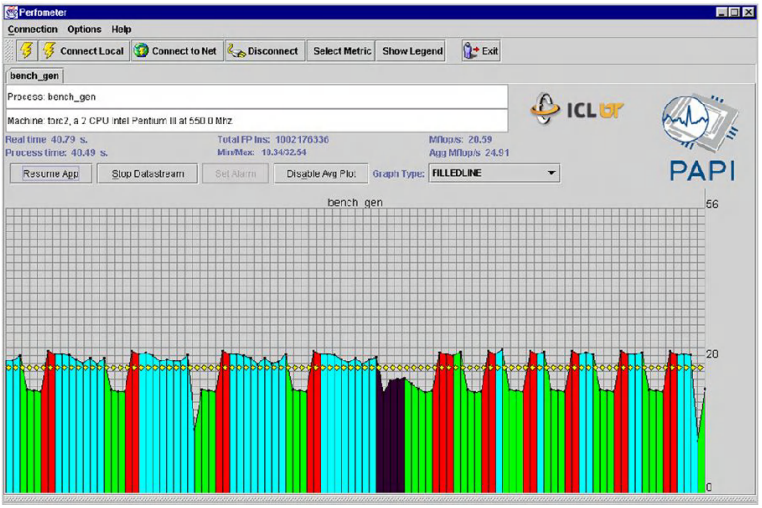


Fig. 2. Real-time performance analysis using Perfometer

disk at the end of the run. Future plans are to develop additional probes, for example for VProf and TAU, and to improve support for instrumentation and control of parallel message-passing programs.

PAPI has been incorporated into a number of profiling tools, including SvPablo [3], TAU and VProf. In support of tracing, PAPI is also being incorporated into version 3 of the Vampir MPI analysis tool [4]. Collecting PAPI data for various events over intervals of time and displaying this data alongside the Vampir timeline view enables correlation of event frequencies with message passing behavior.

Real-time performance monitoring is supported by the *perfometer* tool that is distributed with PAPI. By connecting the graphical display to the backend process (or processes) running an application code that has been linked with the perfometer and PAPI libraries, the tool provides a runtime trace of a user-selected PAPI metric, as shown in Figure 2 for floating point operations per second (FLOPS). The user may change the performance event being measured by clicking on the Select Metric button. The intent of perfometer is to provide a fast coarse-grained easy way for a developer to find out where a bottleneck exists in a program. In addition to real-time analysis, the perfometer library can save a trace file for later off-line analysis. The dynaprof tool described above includes a perfometer probe that can automatically insert calls to the perfometer setup and color selection routines so that a running application can be attached to and monitored in real-time without requiring any source code changes or recompilation or even restarting the application.

[3] `http://www-pablo.cs.uiuc.edu/Project/SVPablo/SvPabloOverview.htm`
[4] `http://www.pallas.com/e/products/vampir/index.htm`

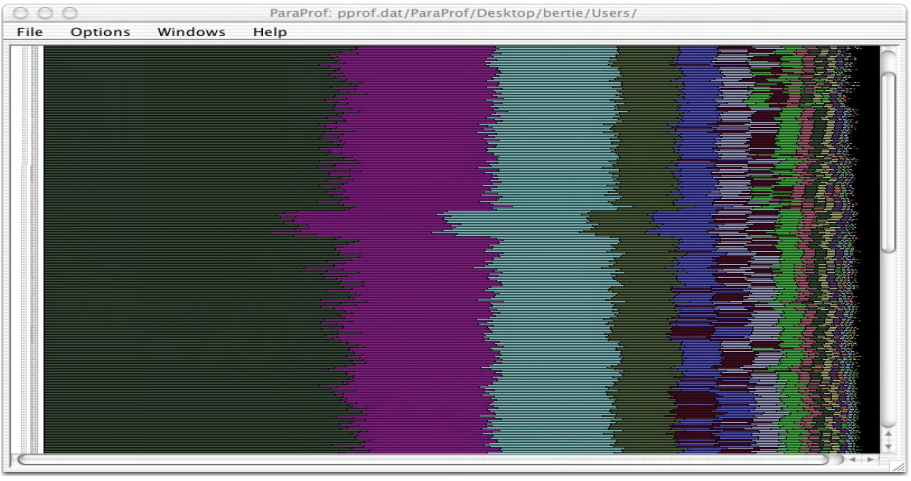


Fig. 3. Scalable SAMRAI Profile Display

4.2 TAU Performance System

The TAU (Tuning and Analysis Utilities) performance system is a portable profiling and tracing toolkit for parallel threaded and or message-passing programs written in Fortran, C, C++, or Java, or a combination of Fortran and C. The TAU architecture has three distinct parts: instrumentation, measurement, and analysis. The program can undergo a series of transformations that insert instrumentation before it executes. Instrumentation can be added at various stages, from compile-time to link-time to run-time, with each stage imposing different constraints and opportunities for extracting program information. Moving from source code to binary instrumentation techniques shifts the focus from a language specific to a more platform specific approach. TAU can be configured to do either profiling or tracing or to do both simultaneously.

Source code can be instrumented by manually inserting calls to the TAU instrumentation API, or by using PDT [10] to insert instrumentation automatically. PDT is a code analysis framework for developing source-based tools. It includes commercial grade front end parsers for Fortran 77/90, C, and C++, as well as a portable intermediate language analyzer, database format, and access API. The TAU project has used PDT to implement a source-to-source instrumentor (`tau_instrumentor`) that supports automatic instrumentation of C, C++, and Fortran 77/90 programs. TAU can also use DyninstAPI [3] to construct calls to the TAU measurement library and then insert these calls into the executable code. In both cases, a selective instrumentation list that specifies a list of routines to be included or excluded from instrumentation can be provided. TAU uses PAPI to generate performance profiles with hardware counter data. It also uses the MPI profiling interface to generate profile and/or trace data for MPI operations.

Recently, the TAU project has focussed on how to measure and analyze larg-scale application performance data. All of the instrumentation and measurement techniques discussed above apply. In the case of parallel profiles, TAU suffers no limitations in

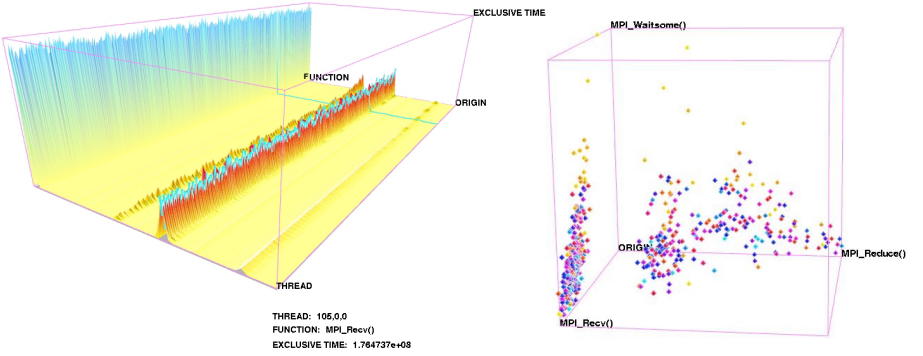


Fig. 4. Performance Profile Visualization of 500 Uintah Threads

the ability to make low-overehead performance measurement. However, a significant amount of performance data can be generated for large processor runs. The TAU ParaProf tool provide the user with means to navigate through the profile dataset. For example, we applied ParaProf to TAU data obtained during the profiling of a SAMRAI [8] application run on 512 processor nodes. Figure 3 shows a view of exclusive wall-clock time for all events. The display is fully interactive, and can be "zoomed" in or out to show local detail. Even so, some performance characteristics can still be difficult to comprehend when presented with so much visual data.

We have also been experimenting with three-dimensional displays of large-scale performance data. For instance, Figure 4 [5] shows two visualizations of parallel profile samples from a Uintah [7] application. The left visualization is for a 500 processor run and shows the entire parallel profile measurement. The performance events (i.e., functions) are along the x-axis, the threads are along the y-axis, and the performance metric (in this case, the exclusive execution time) is along the z-axis. This full performance view enables the user to quickly identify major performance contributors. The `MPI_Recv()` function is highlighted. The right display is of the same dataset, but in this case each thread is shown as a sphere at a coordinate point determined by the relative exclusive execution time of three significant events. The visualization gives a way to see clustering relationships.

5 Conclusions

Terascale systems require a performance observation framework that supports a wide range of instrumentation and measurement strategies. The PAPI and TAU projects are addressing important research problems related to construction of such a framework. The widespread adoption of PAPI by third-party tool developers demonstrates the value of implementing low-level access to architecture-specific performance monitoring hardware underneath a portable interface. Now tool developers can program to a single in-

[5] Visualizations courtesy of Kai Li, University of Oregon

terface, allowing them to focus their efforts on high-level tool design. Similarly, the TAU framework provides portable mechanisms for instrumentation and measurement of parallel software and systems.

Terascale systems require scalable low-overhead means of collecting relevant performance data. Statistical sampling methods, such as used in the new PAPI substrates for the Alpha Tru64 UNIX and Linux/Itanium/Itanium2 platforms, yield sufficiently accurate results while incurring very little overhead. Filtering and feedback schemes such as those use by TAU lower overhead while focusing instrumentation where it is most needed. Both PAPI and TAU projects are developing online monitoring capabilities that can be used to control instrumentation, measurement, and runtime performance data analysis. This will be important for effective performance steering in highly parallel environments.

Together, the PAPI [6] and TAU [7] projects have begun the construction of a portable performance tool infrastructure for terascale systems designed for interoperability, flexibility, and extensibility.

References

1. MPI: A message passing interface standard. *International Journal of Supercomputing Applications*, 8(3/4), 1994.
2. S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *International Journal of High Performance Computing Applications*, 14(3):189–204, 2000.
3. B. Buck and J. Hollingsworth. An API for runtime code patching. *International Journal of High Performance Computing Applications*, 14(4):317–329, 2000.
4. J. Dean, C. Waldspurger, and W. Weihl. Transparent, low-overhead profiling on modern processors. In *Workshop on Profile and Feedback-directed Compilation*, October 1998.
5. A. Eustace and A. Srivastava. ATOM: A flexible interface for building high performance program analysis tools. In *Proc. USENIX Winter 1995*, pages 303–314, 1995.
6. J. Galarowics and B. Mohr. Analyzing message passing programs on the Cray T3E with PAT and VAMPIR. Technical report, ZAM Forschungszentrum: Juelich, Germany, 1998.
7. J. S. Germain, A. Morris, S. Parker, A. Malony, and S. Shende. Integrating performance analysis in the uintah software development cycle. In *High Performance Distributed Computing Conference*, pages 33–41, 2000.
8. R. Hornung and S. Kohn. Managing application complexity in the samrai object-oriented framework. *Concurrency and Computation: Practice and Experience*, special issue on Software Architectures for Scientific Applications, 2001.
9. J. Larus and T. Ball. Rewriting executable files to measure program behavior. *Software Practice and Experience*, 24(2):197–218, 1994.
10. K. Lindlan, J. Cuny, A. Malony, S. Shende, B. Mohr, R. Rivenburgh, and C. Rasmussen. A tool framework for static and dynamic analysis of object-oriented software with templates. In *Proc. SC 2000*, 2000.
11. P. Mucci. Dynaprof 0.8 user's guide. Technical report, Nov. 2002.
12. S. Shende, A. Malony, and R. Bell. Instrumentation and measurement strategies for flexible and portable empirical performance evaluation. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001)*, 2001.

[6] For further information: `http://icl.cs.utk.edu/papi/`

[7] For further information: `http://www.cs.uoregon.edu/research/paracomp/tau/`

Logging Kernel Events on Clusters

Jürgen Reuter and Walter F. Tichy

Institut für Programmstrukturen und Datenorganisation
Fakultät für Informatik
Universität Karlsruhe, 76128 Karlsruhe, Germany
{reuter,tichy}@ipd.uka.de
http://www.ipd.uka.de/~{reuter,tichy}/

Abstract. We present tools for recording and analysing kernel events on Linux clusters. The tools provide cluster-wide event logging at system clock accuracy. We demonstrate the usefulness of our tools by verifying an implementation of a simple remote scheduling feature and present interesting results regarding schedule responsiveness. With our GUI-based Java application, data recorded on multiple hosts is integrated for visualization. These tools can be used for analysis of cluster schedulers such as gang schedulers, cooperative cacheing, network RAM drivers, or parallel file systems.

1 Introduction

Tracking and analysing events on computer clusters is essential for understanding and improving crucial aspects of cluster operating systems, such as gang scheduling, synchronization, parallel file access, managing network RAM, or cooperative cacheing. We provide a set of tools that collect and display kernel events on clusters of Linux computers. The events are logged by kernel modifications on each node individually and integrated in a cluster-wide view for analysis. Our current implementation captures scheduling events and is intended to help understand gang scheduling strategies. Other events can also be captured, such as block lookup and eviction requests in cacheing policies of network RAM systems on globally coordinated cluster caches. Similarly, collecting and analysing information regarding parallel access to files may help choose proper policies for I/O scheduling, cacheing and prefetching. Our event recording approach achieves system clock resolution.

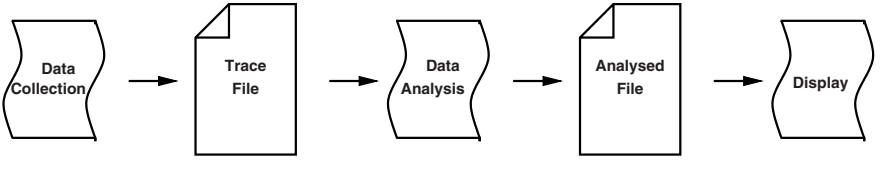


Fig. 1. General Performance Analysis Approach

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 63–72, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Traditionally, post-mortem performance analysis follows a pattern as shown in Figure 1. The first step is to collect data, which can be done in manifold ways. For example, user applications can be instrumented, i.e. additional code may be inserted at selected locations in the application such that when execution comes across such location, timestamp and program status information is recorded and appended to a logfile. This way, the logfile subsumes a trace of selectively recorded events in the course of program execution. Rather than inserting code for instrumentation into a user application, one can also insert such code at relevant locations of underlying libraries. For example, by instrumenting a communication library such as PVM[10] or MPI[5], one can track the communication pattern of a user application that uses the library, but may not be able to find bottlenecks in the communication itself. Our approach is to instrument selected locations of the OS kernel, gaining a better understanding of specific performance issues in the cooperation between the kernel and user applications.

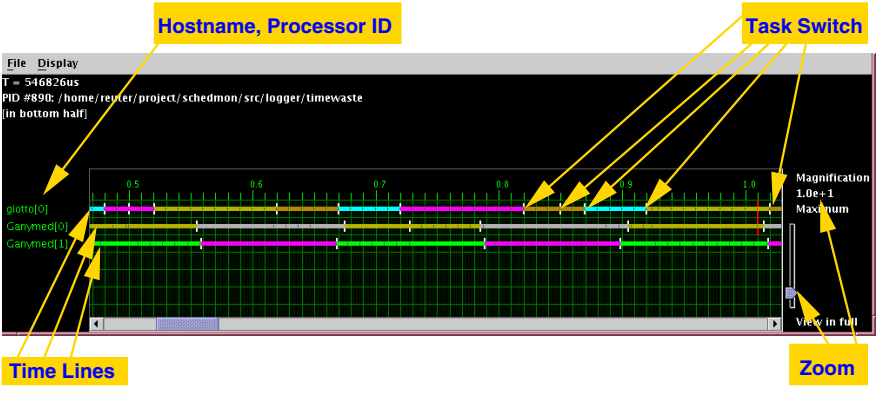


Fig. 2. Data Visualization Tool

The second step in the course of post-mortem performance analysis, the data analysis, consists in filtering events, as well as computing some statistics. The third step is to display the results of the analysis. We integrated these final two steps into a single Java application with a graphical user interface. Data display is based on showing events on per-processor time lines, as shown in Figure 2.

The remainder of this paper is organized as follows: Section 2 gives an overview of the kernel modifications and a companian user level application to record event data. In Section 3, we present the visualization tool for analysing and displaying the data. As an example of usage, in Section 4 we show how we evaluated the performance of a remote scheduling feature with our tools. Section 5 discusses some limitations of our current design and presents possible solutions that we envision for the future. In Section 6, we compare our work with related approaches. Section 7 summarizes the results of this work.

2 Data Collection

The data collection process is realized by recording local time and type of kernel events through instrumentation of the kernel. Our work focuses on instrumenting selected locations in the kernel code rather than instrumenting user level applications. With this approach, we can efficiently log data that is gathered in the kernel without suffering from the additional overhead of switching between kernel and user mode. The possibly most important type of event and relevant motivation for this work is the point of time when the kernel, while executing the `schedule()` function, decides to switch from one task to another one. Similarly, we may want for example to observe when an ICMP message arrives from the network, or how much time the kernel spends in a particular servicing routine, e.g. by recording when the kernel enters and leaves bottom halves[1] as part of the `do_softirq()` kernel function.

For this purpose, we provide a framework consisting of kernel code (based on modifications of a SuSE 8.0 Linux 2.4.18 kernel) that allows for recording such events, and a user level application that is responsible for saving the data to disk. This framework follows the producer-consumer paradigm, but is relaxed in the sense that the producer can not block, but will rather discard data when its buffer overflows.

From the perspective of the kernel code to be instrumented, we simply provide a new kernel function `evtlog_put(struct evtlog_entry_t *entry, int/*bool*/ fill_in_time)`, where `struct evtlog_entry_t entry` consists of an event id, a timestamp and 8 bytes of event specific data. Depending on the boolean flag, the timestamp is either provided by the caller or automatically filled in by calling `do_gettimeofday()`. Upon calling `evtlog_put()`, a copy of the struct is saved in a kernel memory buffer. The function is designed to return quickly, such that the overhead of recording does not affect the collected data too much. Therefore, when the buffer is about to overflow, it does not try to save the data to disk by itself. Instead, the kernel function wakes up a dedicated user level process that is responsible for fetching the data from the kernel and saving it to disk.

Figure 3 illustrates the interactions between the kernel logger code and the user level logging process. It shows an SMP machine with two processors. The left side represents the kernel logger code, the right side the dedicated user level logger process. The user level logging process sleeps by invoking `pause()`, waiting to be notified by the kernel upon arrival of log data. For each processor, the kernel logger code provides memory for two buffers. Upon invocation, the kernel logging function writes any event data into the first buffer of the currently executing processor. When this buffer is full, it swaps the two buffers by exchanging pointer references, then makes the user level logging process runnable by calling the kernel function `wake_up_process()`, and finally returns to the caller. Eventually,

[1] Under Linux, a *bottom half* designates the non-critical part of an interrupt handler that is not executed immediately when the interrupt occurs, but is delayed until all time critical code has been executed.

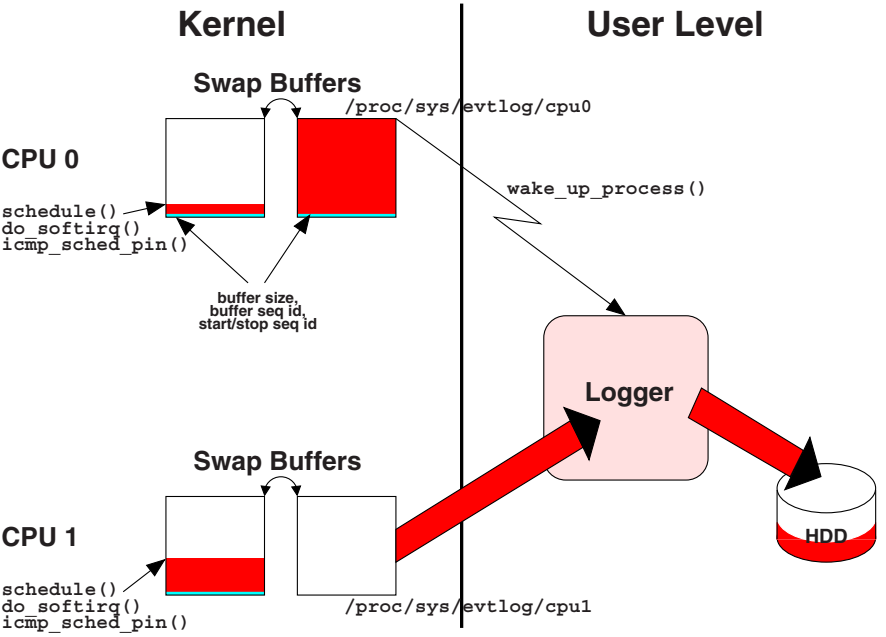


Fig. 3. Interactions between kernel logging code and user level logging process

the kernel scheduler will switch to the user level logging process, since it is now runnable. The user level logging process then looks for any data available for any processor from the kernel logger, fetches it, writes it to disk, and puts itself back to sleep by invoking `pause()` once again. In the current implementation, the data is transferred from kernel to user space by copying it in the course of reading from a dedicated file in the `/proc`[2] filesystem, such that user level applications can initiate read or write access filesystem. When reading from this file, a boolean flag is reset in the kernel, telling it that the buffers may be swapped once again. A future implementation may avoid the extra copy through the `/proc` filesystem by the kernel directly writing the log data into user space memory that has been allocated by the user level logging process.

The user level logging process has to announce itself to the kernel such that the kernel logger code knows which process to wake up upon a full buffer. This is done during startup of the user level logging process through, once more, a special file entry in the `/proc` file system: the user level logging process writes its own process id to that file.

Data may be discarded if the kernel records data faster than the user level logging process is capable of saving, as the process is scheduled a limited amount

[2] The Linux kernel provides pseudo file entries in the `/proc` filesystem, such that user level applications may exchange data with the kernel by writing to or reading from a file in this filesystem.

of time only. To be able to detect such data loss, each buffer contains a per-processor sequence number which is increased whenever the kernel starts writing a new buffer. Checking the sequence numbers for completeness is left to the visualization tool, since the logging itself is designed to produce little overhead.

Kernel logging can be dynamically turned on and off. This mechanism is also done via the `/proc` file system by writing to special files. Whenever kernel logging is restarted, the user level logging process creates a new log file on disk and writes subsequent buffers to it. For that purpose, in addition to the buffer sequence number, there is also a start/stop sequence number that is incremented each time kernel logging is turned on, such that the user level logging process can determine the correct log file. This is necessary, since on an SMP machine, the buffer for one processor may still contain data from an earlier logging phase as compared to another processor.

Furthermore, since upon stopping kernel logging all buffers are flushed, a third special entry is needed that represents the actual size of the current buffer. Choosing a null-terminated array for the buffer implementation is not reasonable, since this would require spending time for scanning through the whole buffer to determine its size when writing it do disk.

With respect to keeping the logging overhead small, the user level logging process just fetches and stores the data in whatever order it retrieves it from the kernel. It only cares for evaluating the start/stop sequence number for writing the data to the correct file.

Since the kernel logger uses per-processor buffers, on an SMP machine, kernel logging can be performed concurrently among all processors without requiring a time-wasting SMP kernel lock: the kernel logging function accesses always the buffers of the currently executing processor, thus scaling fine even on fat SMP nodes. Calling the kernel logging function from multiple locations including bottom half code is possible, since the kernel itself is not preemptive, such that calls to the kernel logging function are executed sucessively without overlapping in time. However, with the current implementation, the kernel logging function should not be called from within a hardware interrupt handler, since the handler might have interrupted a pending call of the kernel logging function. Overlapping calls to the kernel logging function effectively have the same effect as unsynchronized concurrent access to the log buffers, thereby potentially damaging its consistency. Preventing such a scenario requires additional provision such as queuing logging requests. This may be added in a future version.

Our kernel modifications include a call to the kernel logging function whenever the Linux scheduler decides to switch to another process on any processor. This way, the processing time of the user level logging process itself is logged and can be used for determining the logging overhead caused by the user level logging process. This includes the time needed for copying the log data from kernel space to user space, since the read access on the `/proc` file system is executed in the context of the user level logging process.

The overhead in the kernel itself can not be tracked this way. But since the kernel logging function mostly consists of determining the timestamp and writing

a log entry into kernel memory, it can be well estimated by assuming a constant cost of time per log entry for a particular machine. Our experiences suggest that determining the timestamp is a bottleneck, but that obviously depends on the particular hardware.

Since logging is performed locally on each node (assuming that the data is written to a local file system), the system scales fine with an increasing number of cluster nodes. On fat SMP nodes, writing the data to disk may become a bottleneck (though Linux supports only up to 32 processors). Still, if the logging overhead is roughly the same for all nodes, scheduling all logging processes simultaneously as a gang can be used to minimize the effect of the logging overhead onto any other processes of the system.

3 Data Analysis and Display

The user level logging process creates log data files by retrieving data from the kernel via the `/proc` file system and writing it to disk without further processing. It is the task of separate tools to reorder, filter, evaluate and display the data. For this purpose, we provide a standalone Java application with a graphical user interface. Since it is written in a platform independent manner, it runs on several platforms. This is handy, when, for example, the log data is collected on a Linux cluster, but the perfomance analysis will be done on a separate working station that potentially runs under a different OS.

Figure 2 shows a screenshot that depicts typical use of the Java tool. For each processor, grouped on a per host basis, a graphical display panel shows a horizontal time line with the currently running processes displayed in different colors. Some colors are dedicated (black = idle process, red = logger process); all other process are currently mapped to a fixed set of colors in a round-robin way, but support for user-defined color mappings may be added in a future version. Events are marked as vertical ticks across the time line, using different length and color.

The panel's view can be scrolled by either using the scrollbar below the panel or by dragging and dropping its content with the mouse. The latter is handy especially when zooming to high resolution makes the effect of scrollbar movements too coarse. The zoom of the display can be easily controlled by the logarithmic-scaled slider at the right side and allows for zooming from full view to microsecond resolution. Care has been taken to ensure fast and smooth scrolling and zooming by proper design of the data structures for internal log data representation and by cacheing pointers to the data of the currently displayed location. Display of particular event types (`schedule()`, `schedule_task()`, bottom half enter/leave, ICMP schedule requests (see Section 4) and other events) can be turned on and off.

When pointing the mouse cursor on a processor's time line, the ID of the current process (and its name, if available) on that processor is textually displayed at the left side above the display panel. When positioning the mouse cursor on an event tick, additional information about the event such as the event type and its

exact time in microsecond resolution are textually shown. Processor utilization, idle time and other general statistics are shown at program startup.

Log data from multiple hosts can be gathered via NFS: Each host writes its log file to a local disk, in order to minimize the logging overhead, and exports the directory with this file to the workstation on which the visualization tool is run. The tool may then read all log data from all hosts and display it in a single window. It is capable of displaying a large number of processor time lines, practically limited only by the amount of main memory. If memory is limited, our tool can be run with data of only a selected number of nodes that are of interest for the analysis. Filters for data reduction or extraction and combination of data from multiple processor time lines may be essential for a better understanding of huge amount of data. We are planning to implement such filters and proper visualization in a future version of the tool.

4 Preliminary Experience

Originally, our work was motivated by observing and controlling cluster-wide scheduling. For this purpose, we implemented a remote scheduling request feature that, for example, may be used for gang scheduling sets of related tasks of a distributed parallel application in a cluster environment. In our implementation, a host sends a special ICMP[12] message to one or several hosts to select a process group for immediate scheduling. We put additional logic into the ICMP network bottom half code of the Linux kernel such that upon reception of this special ICMP message the priority of the affected processes is set to the maximal possible value. When all bottom halves have been serviced, the Linux kernel reruns the scheduler, such that the time between receiving the ICMP message and scheduling the proper process is mostly determined by the time for processing all pending bottom halves, provided that the chance for further interrupts to intercept is negligible. On an SMP machine, the incoming ICMP message is processed by a single processor that initiates a reschedule on every other processor via a call to the kernel function `smp_send_reschedule()`.

We used our tools to examine how much time is spent from the incoming ICMP message until scheduling the requested processes on all processors. The analysis was performed on a Dual 500MHz Pentium III (Katmai) SMP machine. We naively expected that the processor that handles the incoming ICMP message would perform the requested schedule first, since the `smp_send_reschedule()` would take some time before affecting all other processors.

Figure 4 shows a screenshot of our visualization tool that depicts a region of time between a particular receipt of an ICMP message and the resulting task switch. Surprisingly, the processor that handles the incoming ICMP message is often the last one that switches to the requested process. The reason may be that this processor is still in the midst of executing the network bottom half code, while the other processor on our two processor machine receives the reschedule interprocessor signal and almost immediately switches to the requested process. In any case, we could show that, when no further time is spent in servicing

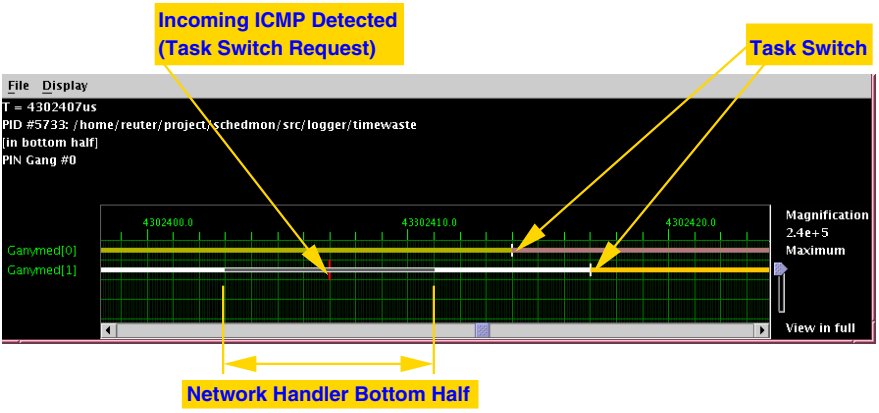


Fig. 4. Incoming ICMP message and resulting task switches

other interrupts, on our hardware the reschedule takes effect within roughly 10 microseconds after receiving the ICMP message.

5 Future Work

Our approach of instrumenting selected locations in the kernel is essential for performance issues on the OS level. Future versions of our tools may consider more process related data or additional event types. For a comprehensive understanding, one needs insight from the application point of view. For example, better support for MPI applications or multithreaded Java programs is desirable, e.g. in order to recognize threads of execution or process groups and observe effects of communication patterns between them. Ideally, one would like to combine our approach with one or more of the many application level oriented performance analysis tools. The process IDs that we record on the OS level offer the natural basis for an integration with log file data collected in user level space. Integration of the data either would be done by a separate tool or integrated into our visualization tool.

When combining log files from multiple hosts into a single display, we assume that all logs were started at the same time and the drift between the clocks is negligible. Otherwise, the display may show a horizontal offset error on the time line. Our modifications to the kernel allow for broadcasting special ICMP messages that remotely start/stop the logging mechanism, using the same infrastructure as for the remote schedule feature. First experience shows that this way, the skew is not worse than 10 microseconds even via a slow 10 MBit/s Ethernet connection. However, intercepting hardware interrupts may contribute to further delay. To eliminate this source of error, we may follow recent work on network time synchronization[2]. This work shows that with broadcasts, the difference in the arrival time is typically small and OS related skew can be minimized by recording the system clock's current time immediately at interrupt

time upon receiption of a packet. Consequently, we would try to record the system time clock upon receipt of an ICMP message as early as possibly and use it as reference start time for the logging process.

6 Related Work

Almost all software performance analysis tools that we know of focus on either application or communication library level. The classical UNIX tool `gprof`[7] and the Java tool `hprof`[8] follow the stochastical profiling approach by interrupting program execution in regular intervals and recording program status information such as the current execution stack.

DiP[9] is a collection of tools for recording a stream of events at PVM or MPI level by instrumenting the underlying communication library. DiP's visualization tool PARAVER uses time lines for visualization and supports textual and graphical display of user defined functions on lists of events.

SCALEA[14] supports automatic and manual instrumentation of user level source code, using a novel representation of code regions in the course of call graph analysis that also covers loops, I/O and communication statements.

Recent versions of Paradyn[1] also follow the approach of examining call graphs. Paradyn uses dynamic instrumentation to insert and delete measurement instrumentation as a program runs. Visualization under Paradyn is mostly based on statistical diagrams and search history graphs rather than time lines.

Due to a growing number of performance analysis tools, some work focuses on standardization. MPI[5] provides a built-in interface for profiling tools. Similar work has been suggested for OpenMP[11]. JavaPSL[3] is a generic language for describing experiment-related data and performance properties, serving as a substitute for specific log file formats and including data of the experiment's environment. Gerndt et al.[6] present an aproach with automatic performance problem search based on the performance properties specification language ASL.

Tamches and Miller[13] present a framework for dynamically instrumenting the Solaris kernel by performing control flow analysis on the kernel machine code and patching it at runtime.

Feng et al.[4] present a system that conceptionally comes pretty close to our approach. Still, their implementation differs in some important details such as the use of SMP locks which our design avoids.

7 Conclusion

In this paper, we presented a set of tools for performance analysis that collects OS level data by instrumenting selected code in the OS kernel. We showed that the logging overhead can be kept small and taken into account in the analysis. Particular challenges and solutions on this approach inlude peculiarities of SMP machines. Using our Java-based visualization tool, we could gain first experience with our tools upon verifying the effect of a remote schedule feature that we

implented in the kernel. We discussed how our approach may be combined with user level space performing analysis and how the synchronization of time lines across multiple machines may be further improved. The results of our work are publicly available at `http://www.ipd.uka.de/~reuter/tools/`.

References

1. Harold W. Cain, Barton P. Miller, and Brian J.N. Wylie. A callgraph-based search strategy for automated performance diagnosis. In *Proc. of Euro-Par 2000 Parallel Processing*, pages 108–122, 2000.
2. Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA., December 2002.
3. Thomas Fahringer and Clóvis Seragiotto Júnior. Modeling and detecting performance problems for distributed and parallel programs with JavaPSL. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, November 2001.
4. W. Feng, M. Gardner, and J. Hay. The magnet toolkit: Design, evaluation, and implementation. *Journal of Supercomputing*, 23(1):67–79, August 2002.
5. Message Passing Interface Forum. MPI: A message-passing interface standard, June 1995. `http://www.mpi-forum.org/docs/mpi-11.ps`.
6. M. Gerndt, A. Schmidt, M. Schulz, and R. Wismüller. Performance analysis for teraflop computers - a distributed automatic approach. In *Euromicro Workshop on Parallel, Distributed, and Network-based Processing*, pages 23–30, Gran Canaria, January 2002.
7. Susan L. Graham, Peter B. Kessler, and Marshall K. McKusick. Gprof: A call graph execution profiler. In *Proceedings of the SIGPLAN '82 symposium on Compiler construction*, pages 120–126, Boston, Massachusetts, United States, June 1982.
8. Stuart Halloway. Using HPROF to tune performance, January 2000. `http://developer.java.sun.com/developer/TechTips/2000/tt0124.html#tip2`.
9. Jesús Labarta, Sergi Girona, Vincent Pillet, Toni Cortes, and Luis Gregoris. DiP: A parallel program development environment. In *Proc. of the 2nd International Euro-Par Conference*, 1996.
10. Oak Ridge National Laboratory. PVM: Parallel virtual machine. `http://www.epm.ornl.gov/pvm/pvm_home.html`.
11. Bernd Mohr, Allen D. Malony, Sameer Shende, and Felix Wolf. Towards a performance tool interface for OpenMP: An approach based on directive rewriting. In *Proc. of Third European Workshop on OpenMPI (EWOMP 2001)*, Barcelona, September 2001.
12. John Postel. Request for comments RFC 792: Internet control message protocol, September 1981. `http://www.ietf.org/rfc/rfc0792.txt`.
13. Ariel Tamches and Barton P. Miller. Fine-grained dynamic instrumentation of commodity operating system kernels. In *Proc. of Operating Systems Design and Implementation (OSDI 1999)*, 1999.
14. Hong-Linh Truong and Thomas Fahringer. SCALEA: A performance analysis tool for distributed and parallel programs. In *Proc. of the 8th International Euro-Par Conference, LNCS 2400*, Paderborn, Germany, August 2002.

Recent Advances in QM and QM/MM Methods

Mark S. Gordon and Michael W. Schmidt

Department of Chemistry and Ames Laboratory, Iowa State University, Ames, IA 50011
mark@si.fi.ameslab.gov

Abstract. Recent advances in advanced quantum chemistry and quantum chemistry interfaced with model potentials are discussed, with the primary focus on new scalable implementations in the GAMESS electronic structure suite of programs. Applications to solvent effects and surface science are discussed.

1 Introduction

In view of the limited space and the impressive array of recent advances in electronic structure theory, this summary will focus on new methods that have recently been implemented into GAMESS (General Atomic and Molecular Electronic Structure System)[1]. The following discussion is divided into three general topics: Recently developed and implemented methods in quantum mechanics (QM), New scalable methods for correlated wavefunctions, and Approaches for interfacing quantum mechanics with molecular mechanics (MM) in order to treat solvation and surface science.

2 QM Methods

In the 1950s Löwdin showed that a wavefunction that includes all possible excitations from the reference wavefunction (usually the electronic ground state) is the exact wavefunction for the given atomic basis. Therefore, this level of theory, commonly called full configuration interaction (full CI), is the benchmark against which all advanced QM methods that include electron correlation may be measured. Indeed any level of CI, perturbation theory, or coupled cluster theory can be extracted from a full CI wavefunction and compared with the exact result. It is therefore very useful to develop and implement a full CI method that can be applied to as large an array of atomic and molecular species as possible. Such a full CI code based on a determinant, rather than a configuration, expansion has been developed by Ivanic and Ruedenberg[2] and implemented into GAMESS.

A special case of full CI is the CASSCF (complete active space self-consistent field) or FORS (fully optimized reaction space) approach in which one defines an active space of orbitals and corresponding electrons that are appropriate for a chemical process of interest[3]. The FORS wavefunction is then obtained as a linear combination of all possible electronic excitations (configurations) from the occupied to the unoccupied (virtual) orbitals in the active space. Since the FORS wavefunction generally corresponds to an incomplete CI, one also optimizes the molecular orbital coefficients to self-consistency. The resulting FORS/CASSCF multi-configurational

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 75-83, 2003.
© Springer-Verlag Berlin Heidelberg 2003

self-consistent-field (MCSCF) method is very powerful for a variety of application but the size of the active space for actual calculations is generally limited to roughly 14 electrons in 14 orbitals, or a full valence active space for a molecule the size of ethane.

Of course, even with the most efficient code, a full CI calculation or a full valence FORS calculation is limited to very small atoms and molecules. It is therefore very important to think about ways in which one can approach the accuracy of a full CI wavefunction with a much smaller effort than that required for full CI. In principle, one would expect that a great many of the configurations in a full CI or a FORS wavefunction have little effect on the total molecular energy, but it is not obvious how one would identify the important contributors to the total wavefunction. Ruedenberg, Ivanic and Bytautas have used the full CI code and a systematic analysis of single, double, triple, … excitations in order to develop a general method for eliminating the "deadwood" from the full CI wavefunction[4]. Making use of localized MCSCF orbitals (LMOs), they have shown for several test cases that roughly 90% of the configurations in a full CI list can be eliminated while retaining millihartree accuracy. Fig. 1 illustrates the effectiveness of this approach by comparing the model energies vs. the known CCSD(T) correlation energies for 38 small to moderate size molecules,

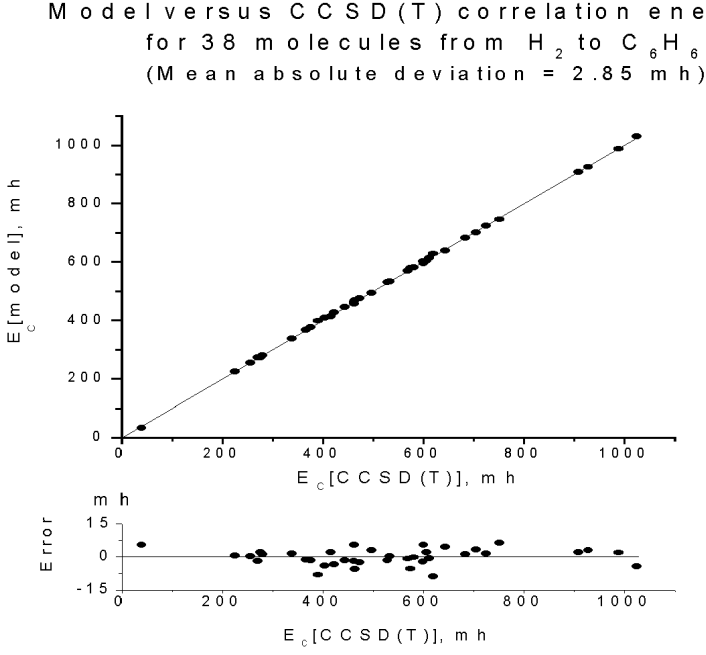


Fig. 1

with a mean absolute deviation of less than 3 mh. The error in *relative* energies, for example for chemical reactions, is likely to be much less.

Once one has implemented a general CI such as that described above, the next logical step is to use this same approach for a MCSCF *ansatz*. The extension of a general CI method to a general MCSCF method is non-trivial, but it has been accomplished and implemented into GAMESS in collaboration with the Ruedenberg group[5]. It is important to recognize that there are clear advantages and some disadvantages to the general MCSCF approach. The obvious advantage is the dramatic reduction in computation time that one attains by eliminating most of the configurations. What one gives up in this approach is the built-in size-consistency that is guaranteed by the complete active space approach. Until the method has been extensively tested, it is not clear how serious a matter this is. Indeed, it is possible that eliminating essentially non-contributing configurations has only a small effect on size-consistency. Similarly, it is not clear how the MCSCF convergence will be affected when a complete active space is not used.

Since a prime motivation for the general MCSCF approach is to expand the size of a chemical system that one can study at this level of theory, an especially exciting development is the very new ORMAS (Occupation Restricted Multiple Active Space) method developed by Ivanic[6]. This direct CI method, recently implemented into GAMESS, permits one to subdivide a FORS/CASSCF active space into multiple subspaces in a completely general way, and then imposes limits on the electron occupancies in each subspace. Since each subspace is treated individually as a complete active space, the ORMAS method enjoys all of the advantages of any FORS/CASSCF method (e.g., size-extensivity, good SCF convergence, straightforward formulation of energy gradients), while having the potential to greatly expand the accessible size of chemical systems. The method has already been applied with success to the very difficult N_2O_4 system, as well as a complicated transition metal complex[6]. Since MCSCF provides the correct zeroth order wavefunction, but does not include dynamic correlation, Ivanic has also developed an analog to the second order CI method; that is, single and double excitations out of all determinants in the ORMAS wavefunction. It is anticipated that this suite of methods will be heavily used in the future.

All of the methods discussed above are based on a multi-reference (MR) approach to obtaining wavefunctions and properties. Such MR approaches are often necessary, because many chemical problems involve species with considerable configurational mixing (frequently referred to as diradical character). However, the amount of diradical character in a chemical system can span a very broad range, from essentially zero (e.g., HOMO occupancy ~2, LUMO occupancy ~0) to fully diradical (HOMO occupancy ~1, LUMO occupancy ~1). As one approaches fully diradical character, all single reference methods break down, but they do not break down at the same rate as one approaches this limit. In particular, there is considerable evidence that coupled cluster (CC) methods, particularly those like CCSD(T) that incorporate a triples correction, can overcome the deficiency of a single reference wavefunction for problems with non-trivial diradical character. This was demonstrated, for example, by examining the N_2 dissociation curves for MP2 and CCSD(T) vs. various MR methods[7]. The breakdown in the CCSD(T) calculation appears much later in the dissociation process than does the MP2 breakdown. Recent developments by Piecuch

and co-workers[8] are particularly exciting, since they extend this breakdown even further out in the dissociation curve.. Termed re-normalized and completely re-normalized methods (e.g., R-CCSD(T) and CR-CCSD(T)), these methods are designed to account for an increasing amount of diradical character. Although they do eventually break down at large distances for multiple bonds, they are clearly more robust for intermediate cases. The full suite of closed shell CC, R-CC and CR-CC methods are now available in GAMESS, and their equations-of-motion (EOM) analogs (especially important for investigating electronically excited states) will become available within the next six months.

3 Scalable Methods

One approach to growing the size of a chemical system that can be realistically treated by the most sophisticated methods is to devise new methods that are inherently more efficient, as discussed in the previous section. Another, complementary approach is to devise algorithms in such a manner that the calculations are scalable; that is, the computationally most demanding tasks may be distributed among whatever processors are available. Often referred to as parallel programming, this approach is *relatively* straightforward for low-level methods like Hartree-Fock and density functional theory energies and gradients, but become increasingly complicated for the more sophisticated correlated methods. Early approaches to parallel methods relied on replicated data (RD) algorithms, in which the data sets, such as Fock and density matrices, are replicated on each compute node, while the two-electron integrals are computed in a "direct" manner, on-the-fly. The disadvantage to the RD approach is that although a calculation proceeds more rapidly than it would on a single processor, the feasible size of a chemical system is limited by the amount of memory and disk on the smallest node. Therefore, the RD approach is sensible when only two-dimensional matrices are involved, but becomes much less viable for correlated methods for which the four-dimensional electron repulsion integrals must be manipulated (i.e., transformed between the AO and MO basis).

A major advance in the manner in which QM (especially correlated QM) calculations may be performed on parallel computers was provided by the development at PNNL of the global array (GA) tools[9], a one-sided message passing library that facilitates the distribution of large sets of data across all available nodes. The development of the distributed data interface (DDI)[10] in GAMESS benefited considerably from the prior development of GA technology. DDI performs best when it can take advantage of the SHMEM library, especially on Cray systems, but it has also been very successful on clusters of UNIX and Linux computers. The point-to-point messages required for the implementation of DDI on such hardware are carried by TCP/IP socket messages or, sometimes, an MPI-1 library.

The initial implementation of DDI was for closed shell MP2 energies and gradients[11]. This has been extremely successful. As long as the size of the system of interest is increased as the number of CPUs is increased, the method scales almost linearly up through 512 T3E processors[12]. For species with unpaired electrons, the implementation of restricted open shell energies is equally efficient, and it is anticipated that UMP2 energies and gradients (currently in progress) will scale as well as the closed shell analog. Restricted open shell gradients using the ZAPT *ansatz*

have been derived[13], and the coding of both sequential and parallel codes will begin shortly. DDI has also been used to develop a distributed parallel MR perturbation method in collaboration with the Koseki group[14]. It appears that the parallel MRMP2 method currently scales well up to about 32 processors.

Since MCSCF is an important starting point for so many chemical problems, it is very important to develop parallel MCSCF methods as well. The initial attempt at this was a RD approach which scaled well only to ~4-8 processors[15]. Very recently, a DD parallel MCSCF algorithm has been developed using the full Newton-Raphson convergence algorithm[16]. This DD MCSCF method addresses the integral transformation and orbital rotation steps, but not the CI coefficient optimization, which is treated in the next paragraph. Initial tests suggest that this algorithm will scale well up to ~32-64 processors, a major advance over the RD algorithm.

As noted in the Introduction, the ultimate wavefunction for a given basis is the full CI wavefunction, so it is important to extend the sizes of chemical species that can be realistically approached using full CI. Equally important is the recognition that a full CI within a specified set of orbitals and corresponding electrons is just a FORS/CASSCF wavefunction. So, the development of a scalable Full CI method serves a dual purpose. Both RD and DD full CI codes have been developed and implemented into GAMESS[17]. The algorithm uses a CI driven approach, in which communication is controlled by a string-driven method. The success of the DD/FCI method is especially encouraging, as is illustrated in Fig. 2.

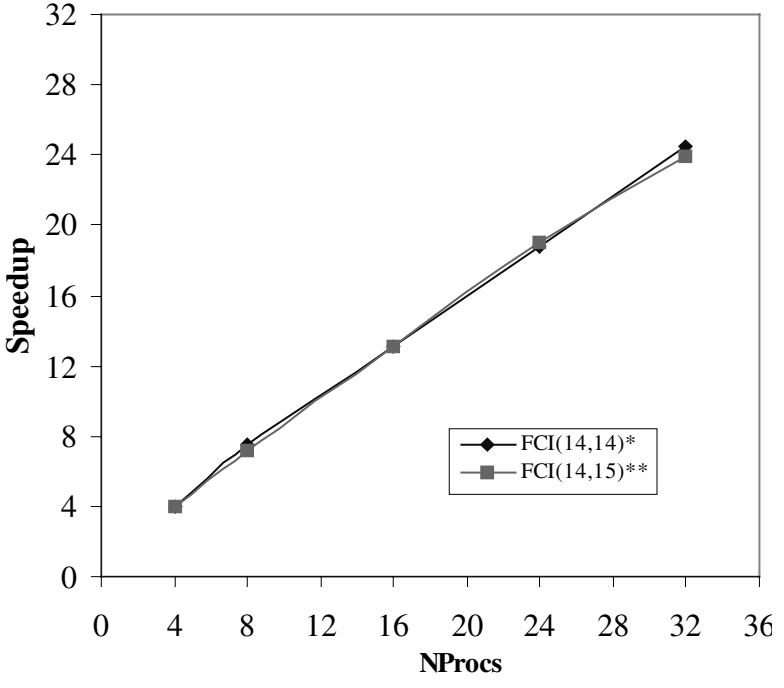


Fig. 2

This figure demonstrates a test on a cluster of 64-bit IBM Power 3II dual processor computers running AIX. The illustrated problems are CH_3OH (14,14) and H_2O_2 (14,15), where the numbers in parentheses signify the number of electrons and orbitals, respectively. These problems include ~11,800,000 and 40,400,000 determinants, rspectively, and the scalability through 32 processors is excellent. Similar performance is observed on Linux clusters up through the 16 processors that were available for testing.

One can think of the parallel methods discussed above as fine-grained parallelism, in that each subtask in a single energy or energy+gradient evaluation is individually distributed among available processors. There are also problems for which a very coarse-grained approach is appropriate. Examples are the computation of numerical derivatives (e.g., gradients and hessians) for which each displaced geometry is separate from the others, and all displacements may be identified at the beginning of the calculation. Another example is a Monte Carlo calculation, since again, the energy evaluations are independent of one another. A development underway in GAMESS is the GDDI (generalized DDI) method which makes use of the concept of groups and subgroups (in a computational science sense) to make use of both fine-grained and coarse-grained parallelism[18]. For example, if one wishes to perform a MP2 Monte Carlo study, one can distribute the large number of MP2 energy evaluations among all available nodes. At the same time, if each node is a multi-processor (e.g., SMP) computer, each MP2 energy calculation can itself be distributed among the processors on its node.

4 QM/MM Methods

Even with the most clever and efficient methods and scalable algorithms, as the size of the system of interest grows, sooner or later the compute power is not up to the task if one uses fully QM methods, especially correlated ones. Two important areas of research that fall into this category are solvent effects (more generally liquid behavior) and surface science. An effective alternative to fully QM methods is the combination of QM with molecular mechanics (MM) methodology. MM is a term that generally suggests that one is using classical techniques with no wavefunction; such methods vary broadly in sophistication. Two types of MM methods that are very different in their level of sophistication are discussed here.

The approach taken in GAMESS to study solvation is a multi-layer one in which the innermost layer consists of the solute plus some number of solvent molecules that one feels must be treated with explicit QM. Examples of the latter are water molecules that act as conduits in H-transfer reactions. The second layer consists of a sophisticated potential, the effective fragment potential (EFP) that is derived from rigorous QM[19]. The outermost layer is represented by a reliable continuum method to represent the bulk liquid. In its original EFP1/HF version, this method described solvent molecules (i.e., water) by three terms that are added to the QM (i.e., HF) Hamiltonian. The first term represents the Coulomb interactions by a distributed multipole analysis (DMA) expanded through octopoles. The entire Coulomb term is multiplied by a distance-dependent cutoff to account for overlapping charge distributions. The second, induction, term accounts for the polarization of charge densities in a self-consistent manner using localized molecular orbitals (LMOs). The

third term is fitted to the remainder of the HF dimer potential and represents exchange repulsion and (implicitly) charge transfer. This method has been very successful for problems that are well described by the HF method, but it is limited in two respects. First, HF includes no electron correlation which invades all of the terms mentioned above and introduces entirely new interactions, most notably dispersion. Second, the process of fitting to obtain the exchange repulsion/charge transfer term is not something one wants to do for every solvent or liquid of interest.

The first of these limitations has been partially addressed by reformulating the EFP1 approach with density functional theory (DFT), using the popular B3LYP functional[20]. Denoted EFP1/DFT, this method includes some correlation, although not long-range dispersion, and therefore produces much better binding energies, for example, in waer clusters. So, this approach only partially accounts for the correlation problem and does not address the issue of fitting the repulsion term. In this sense, it is desirable to derive the exchange repulsion and charge transfer from "first principles" instead of employing fitting procedures. This has been accomplished for the exchange repulsion by expanding this interaction as a power series in the intermolecular overlap. This is not a new idea, but combining this approach with highly transferable LMOs to calculate these integrals and the related intermolecular kinetic energy integrals has been very successful for a wide variety of solvents[19]. The exchange repulsion calculated by this method maps the exact HF intermolecular exchange typically to within 0.5 kcal/mol. It remains to develop analogous expressions for charge transfer, but this EFP2/HF method is already a success, and it has been extended by Jensen and co-workers[21] to the treatment of intramolecular covalent, rather than intermolecular interactions. Neither EFP1 nor EFP2 include dispersion interactions at present, but the development of dispersion terms for both methods is in progress.

Although the cost of an EFP calculation is several orders of magnitude smaller than that of a corresponding (e.g., HF, DFT, MP2) calculation, the cost can rise considerably if one incorporates a large number of solvent molecules. This cost reflects not only the inherent cost of a single energy + gradient calculation, but also the fact that the number of arrangements of solvent or liquid molecules expands rapidly with the number of molecules. This means to find the global minimum, for example, one requires a Monte Carlo or similar calculation that requires a great many energy evaluations. Likewise, to predict bulk properties, one would employ a molecular dynamics scheme that generates a great many energy + gradient evaluations. Both Monte Carlo/simulated annealing[22] and molecular dynamics[23] codes have been implemented in GAMESS, combined with the EFP methods. To make such calculations more feasible for several hundred fragments, each term in the EFP method has been made scalable. As for any other application, the scalability relies on the size of the problem, but the scalability looks promising up through the 16 Linux nodes that have been available for testing.

For surface chemistry, a more traditional QM/MM approach, SIMOMM[24] (surface integrated molecular orbital molecular mechanics), has been developed and implemented in GAMESS. SIMOMM is an embedded cluster approach in which the QM part of the system is embedded into a much larger MM cluster to represent the bulk. Any level of QM theory in GAMESS can be used for the QM part, while the TINKER code is used for the MM part. The interface between the QM and MM parts

is represented by link atoms that appear in the QM part as hydrogens and in the MM part as the actual surface atoms of interest. Gradients for both the QM and MM methods are generally available, so full geometry optimizations are both feasible and recommended. The method has been most extensively applied to problems that involve the Si(100) surface, including addition of organic molecules to the surface, etching, and diffusion of metal atoms along the surface. More recently, it has been applied to the growth of diamond and silicon carbide surfaces.

Acknowledgements. The work described in this paper was supported by the Air Force Office of Scientific Research via a software development (CHSSI) grant and by the Department of Energy through a SciDAC grant. The authors are most grateful to our many collaborators who have contributed their efforts to GAMESS, most notably Professors Klaus Ruedenberg, Jan Jensen, Shiro Koseki and Piotr Piecuch, and Drs. Joe Ivanic and Graham Fletcher, and of course all the present and past graduate students.

References

1. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J., Koseki, S., Matsunaga, N., Nguyen, K. A., Su, S., Windus, T.L., Dupuis, M., Montgomery, J.A.: *J. Comput. Chem.* **14**, 1347-1363 (1993).
2. Ivanic, J., Ruedenberg, K.: *Theoret. Chem. Acc*. **106**, 339-351 (2001).
3. Schmidt, M.W., Gordon, M.S.: *Annu. Rev. Phys. Chem.* **49**, 233-266 (1998).
4. a) Bytautas, L., Ruedenberg, K.: *Mol. Phys*., **100**, 757-781 (2002).
   b) Ivanic, J., Ruedenberg, K.: *Theoret. Chem. Accts*., **107**, 220-228 (2002).
5. Ivanic, J., Ruedenberg, K.: *J. Comp. Chem*., in press.
6. Ivanic, J.: *J. Chem. Phys*., submitted.
7. Gordon, M.S., Schmidt, M.W., Chaban, G.M., Glaesemann, K.R., Stevens, W.J., Gonzalez, C.: *J. Chem. Phys.* **110**, 4199-4207 (1999).
8. Piecuch, P., Kucharski, S.A., Kowalski, K., Musial, M.: *Comput. Phys. Commun.*, **149**, 71-96 (2002).
9. Nieplocha, J., Harrison, R.J., Littlefield, R.J.: *Proc. Supercomputing'94*, 340-349 (1994).
10. Fletcher, G.D., Schmidt, M.W., Bode, B.M., Gordon, M.S.: *Comput. Phys. Commun.* **128**, 190-200 (2000).
11. Fletcher, G.D., Schmidt, M.W., Gordon, M.S.: *Adv. Chem. Phys.,* **110**, 267-294 (1999).
12. Kudo, T., Gordon, M.S.: *J. Phys. Chem. A*, **105**, 11276-11284 (2001).
13. Fletcher, G.D., Gordon, M.S., Bell, R.S.: *Theoret. Chem. Accts*., **107**, 57-70 (2002).
14. Umeda, H., Koseki, S., Nagashima, U., Schmidt, M.W.: *J. Comput. Chem*., **22**, 1243-1251 (2001).
15. Windus, T.L., Schmidt, M.W., Gordon, M.S.: *Theoret. Chim. Acta* **89**, 77-88 (1994).
16. Fletcher, G.D.: to be published.
17. Gan, Z., Alexeev, Y., Kendall, R.A., Gordon, M.S.: *J. Chem. Phys.,* in press.
18. Olson, R.M., Fedorov, D.G., Schmidt, M.W., Gordon, M.S.: to be published.
19. Gordon, M.S., Freitag, M.A., Bandyopadhyay, P., Jensen, J.H., Kairys, V., Stevens, W.J.: *J. Phys. Chem. A*, **105**, 293-307 (2001).
20. Adamovic, I., Freitag, M.A., Gordon, M.S.: *J. Chem. Phys.*, in press.

21. Kairys, V., Jensen, J.H.: *J. Phys. Chem. A*, **104**, 6656-6665 (2000).
22. Day, P.N., Pachter, R., Gordon, M.S., Merrill, G.N.: *J. Chem. Phys.,* **112**, 2063-2073 (2000).
23. Netzloff, H.M., Gordon, M.S.: to be published.
24. Shoemaker, J., Burggraf, L.W., Gordon, M.S.: *J. Chem. Phys*., **112**, 2994-3005 (2000).

UTChem – A Program for *ab initio* Quantum Chemistry

Takeshi Yanai[1], Haruyuki Nakano[2], Takahito Nakajima[2,3], Takao Tsuneda[2], So Hirata[4], Yukio Kawashima[5], Yoshihide Nakao[2], Muneaki Kamiya[2], Hideo Sekino[6], and Kimihiko Hirao[2, 5]

[1] Oak Ridge National Laboratory, PO Box 2008 MS6367
Oak Ridge TN 37831-6367, USA
[2] Department of Applied Chemistry, School of Engineering,
University of Tokyo, Tokyo, Japan 113-8656
[3] PREST, Japan Science and Technology Corporation (JST)
[4] William R Wiley Environmental Molecular Sciences Laboratory
Battelle, Pacific Northwest National Laboratory
K1-96, P.O.Box 999, Richland, WA 99352, USA
[5] Institute for Molecular Science
Myodaiji, Okazaki, Japan 444-8585
[6]Department of Knowledge-based Information Engineering
Toyahashi University of Technology
Toyohashi, Japan 441-8580

Abstract. UTChem is a quantum chemistry software developed by Hirao's group at the University of Tokyo. UTChem is a research product of our work to develop new and better theoretical methods in quantum chemistry.

1 Introduction

Present molecular quantum theory is highly sophisticated, and has evolved dramatically. Software forms a basis for computational chemistry. However, it is not an easy task for an individual/group to develop a comprehensive new program package in *ab initio* quantum chemistry from scratch. Several years ago, we decided to accept this challenge. In view of the availability of such good programs as Gaussian, Gamess, Molcas, NWChem, etc., one may question the relevance of a new program package. We have three arguments for our project. (1) First, we believe that healthy competition is very important in science. (2) Second, we can have a good harvest by doing research using other programs, but it is an abortive flower. We could not make a true breakthrough if we were circumscribed by current software limitations. (3) Third, in spite of the excellent performance of other programs, there are important and powerful methods that others cannot yet handle. We have developed new methodologies in quantum chemistry, particularly the multireference-based perturbation theory for describing chemical reactions and excited states, relativistic molecular theory to treat heavy elements, parameter-free (less) exchange and correlation functionals in DFT, highly efficient algorithms for calculating molecular integrals over generally contracted Gaussians, etc. UTChem is a research product of our work to develop new and better theoretical methods in quantum chemistry. Most of the codes have been developed recently by Hirao's group at the

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 84–95, 2003.
© Springer-Verlag Berlin Heidelberg 2003

University of Tokyo. The basic philosophy behind UTChem is to develop methods that allow an accurate and efficient computational chemistry of electronic structure problems for molecular systems in both the ground and excited states. We have started joint developments with a few coalition partners, who are contributing their codes to UTChem. Their contributions have enhanced the capabilities of UTChem to carry out well-developed methods such as RPA (by Sekino), MPn, CI, CC, etc (by Hirata on TCE project),[1] which are standard in most quantum chemistry programs. We are aiming ultimately at better performance than other programs. UTChem will soon be ready for distribution. Here you will be able to see the features of UTChem. UTChem contains a large number of improvements and some interesting new features, which others cannot match.

2 Synthetic Design and Conception for Implementation of UTChem

A packaging of UTChem got started in a situation that researchers and students in Hirao laboratory had accomplished their implementations individually on independent developmental environments. The accomplishments are covering wide range of capabilities enough to do a sequence of *ab initio* calculations without using other extant distributions as follows,

1) multireference perturbation method (MRMP, MCQDPT),
2) multiconfigurational self-consistent field method (QCAS, GMC, CASVB),
3) relativistic 1/2-component method (RESC, DK3),
4) relativistic 4-component method (DHF, DKS),
5) one- and two-electron integrals,
6) direct and conventional self-consistent field method,
7) density functional theory (OP, PFREE),
8) dynamics and simulation involving QM/MM,
9) response theory for Hartree-Fock and density functional theory.

Our synthetic design and conception are basically coming from a practical necessity on how smooth and stressless compilation of the above existing program codes can be carried out. The fundamental structure we desire is that UTChem should be a program suite which enables developer groups to develop separately executable programs of their owns individually with no consolidation into one binary. This approach is based on a development philosophy that programmer groups can program as freely as possible with no disturbance of serious problems and fatal bugs of the damaged codes the other programmer groups have committed, and no intruder of the other developer groups into their own pretty codes and motivations.

Technically, through a sequence of calculations using separated binaries, what the individual program binaries share with each other is data files compliant with the file formats the developers have specified. UTChem itself just governs a number of rules of the file interfaces, (i.e., no subroutine interfaces, no program rules, ...), which are an exclusive way to connect the individual executable binaries. This design is simple and widely used in other program systems, meanwhile a method to control several program binaries has to be implemented so as to distribute them as a reliable and usable program suite. In UTChem, Python language is used to provide a high-level

and flexible script to control and manage the calculation flows consisted of separately executed programs. We believe that this packaging approach also widens accessibility, in which various developers also easily contribute their own program to UTChem.

As to practical calculation flows, figure 1 depicts a flowchart in UTChem. After the frontend program reads a given input file, required one- and two-electron AO integrals are calculated. HF/KS-SCF as well as TD-HF/DFT for excitation energies and Local MP2 are carried out with 2e AO integrals directly-driven or restored from disk. The single-reference electron correlation theories such as configuration interaction theory (CISD, CISDT, CISDTQ), coupled-cluster theory (CCD, LCCD, CCSD, LCCSD, QCISD, CCSDT, CCSDTQ), and Møller-Plesset perturbation theory (MP2, MP3, MP4) achieved by tensor contraction engine (TCE) are following the integral transformation. Multiconfigurational calculations including MCSCF and perturbation (MRMP and MCQDPT) method are connected with AO integral part and HF-SCF part. After the above single-point energy calculations, if necessary and executable, the derivative energies with respect to the nuclear coordinates are derived. The gradients are used in the geometry optimization code and dynamics simulation including QM/MM.

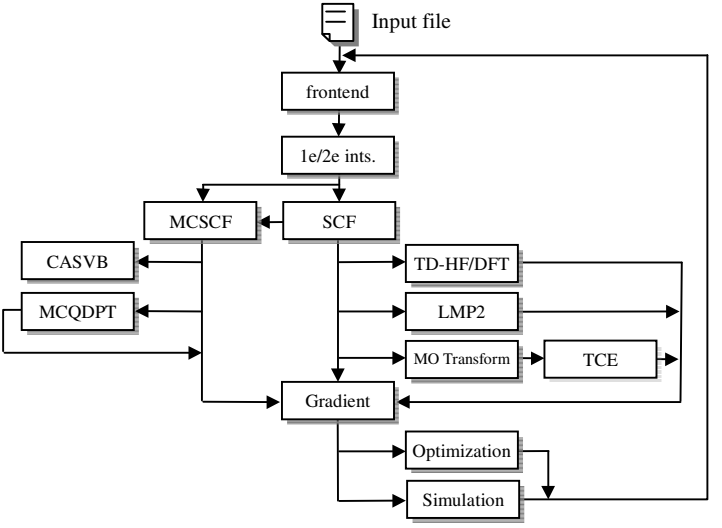


Fig. 1. Schematic calculation flowchart in UTChem

2. 1 INTEGRA

INTEGRA is currently a primal module of UTChem program suite, which we started to write from scratch in Jun 1999. Fortran 90 language is chosen for writing codes in point of its simplicity and flexibility even for quantum chemists reluctant to overcome the computational complexity. Our scientific motivation of the initial implementation of INTEGRA was to develop an efficient integral method for generally contracted Gaussian-type orbitals (GTOs). The successful achievement of the correlation consistent basis sets developed by Dunning[2] and the atomic natural orbital (ANO) basis sets by Almlöf and Taylor[3] has spurred demands of the efficient two-electron integral program based on the general contraction scheme. The difficulty in efficient calculations using general contraction scheme is that the generally contracted GTOs are in nature composed of quite a few primitive GTOs, and the conventional *ab initio* programs and algorithms are designed to calculate the segmentally contracted GTOs. Recently Ishida has reported an efficient formulation for electron repulsion integrals (ERIs) using accompanying coordinate expansion (ACE) method.[4] His formulation can yield a formulation of ERIs requiring the lowest FLOP counts in the ever reported formulations, and especially the reduction of FLOPs for ERIS over highly contracted GTOs is outstanding. We have developed a new integral method oriented to the generally contracted Gaussian-type orbitals (GTOs), and succeeded to implement an efficient integral code SPHERICA onto INTEGRA.[5] Our algorithm successfully specialized to the general contraction scheme with exploiting the efficiency of Ishida's formulation could lead to FLOP count dependence against the degree of contraction K and the number of contracted GTOs N as

$$\text{FLOPs} = x_1 K^4 N + x_2 K^3 N^2 + y_1 K^2 N^3 + y_2 K N^4 + z N^4. \quad (1)$$

Comparing with the segmental contraction scheme, which gives FLOP count dependence as,

$$\text{FLOPs} = N^4 (x K^4 + y K^2 + z), \quad (2)$$

our algorithm gains a drastic efficiency in $O(N^3 + KN^2)$. The name of the code SPHERICA is coming from a capability to calculate ERIs over spherical harmonics generally contracted GTOs efficiently without transformation from Cartesian indices. Table I shows the CPU times to calculate benzene molecule using ANO basis sets on the single CPU of IBM RS/6000 260 model. Currently the integral code SPHERICA is contributing to a direct and conventional SCF routine, in which a robust orbital optimization using direct inversion in the iterative subspace, DIIS, the symmetry usage within D_{2h} and subgroup, DFT numerical grid-quadrature routine for the exchange-correlation potentials, and the parallelization using Global Array[6] have been implemented.

Historically INTEGRA has got an integrated platform for other implementations toward post-SCF routine, such as the integral transformation routine, the time-dependent Hartree-Fock / density functional theory (TD-HF/DFT) routines, the single-reference many-electron correlation routines achieved by TCE, and so on, except MR2D and S&D. The relativistic calculation code REL4D is also programmatically included in INTEGRA. Even though several applications are

consolidated into one binary INTEGRA, a switch specified in a command option allows each application to execute independently, and therefore schematically each application can be regarded as an independent binary. The reason why developers have been working to write codes mainly in INTEGRA without getting their codes an individual binary is just for convenience of the program interface of INTEGRA and Fortran90 environment. Remarkably, the contributors of RPA and TCE were willing to work on cooperative coding on INTEGRA.

To summarize, INTEGRA efficiently provides the following basic integrals necessary for *ab initio* calculations with Cartesian and spherical harmonics basis,

1) one-electron integrals and first derivatives (overlap, core Hamiltonian and first dipole moment)
2) two-electron integrals and first derivatives,
3) one- and two-electron integrals transformed into MO indices.

The parallelized HF/KS-SCF and parallelized TD-HF/DFT can be calculated using directly driven integrals. The detail features on DFT, TCE and REL4D modules are mentioned in other sections.

Table 1. CPU times (sec.) to calculate Fock matrix in SCF calculation for benzene molecule.

	Primitive / Contracted GTOs	Number of basis	SPHERICA	HONDO	MOLCAS
ANO-S	C: 10s6p3d / 3s2p1d H: 7s3p / 2s1p	114	379	2,710	237
ANO-S	C: 10s6p3d / 4s3p2d H: 7s3p / 3s2p	192	870	17,500	358
ANO-L	C: 14s9p4d / 3s2p1d H: 8s4p / 2s1p	114	1,003	4,330	796
ANO-L	C: 14s9p4d3f / 4s3p2d1f H: 8s4p3d / 3s2p1d	264	6,080	125,000	4,910

2.2 MR2D – Multi-Reference 2nd-Order Perturbation Theory and Its Derivatives

MR2D is a program for a multistate multireference perturbation theory, the quasi-degenerate perturbation theory with MC-SCF reference functions (MC-QDPT) [7]. In this PT, state-averaged CAS-SCF is first performed to set reference functions, and then an effective Hamiltonian is constructed, which is finally diagonalized to obtain the energies of interest. This theory includes multireference Møller–Plesset (MRMP) PT (the case that the set of reference functions reduces to a single function).[8]

Let the reference functions be CAS-SCF wavefunctions,

$$|\alpha\rangle = \sum_A C_A |A\rangle . \tag{3}$$

The effective Hamiltonian to second order is given by

$$\left(H_{\text{eff}}^{(0-2)}\right)_{\alpha\beta} = \langle\alpha|H|\beta\rangle + \frac{1}{2}\sum_I \left\{ \frac{\langle\alpha|H|I\rangle\langle I|H|\beta\rangle}{E_\beta^{(0)} - E_I^{(0)}} + \frac{\langle\beta|H|I\rangle\langle I|H|\alpha\rangle}{E_\alpha^{(0)} - E_I^{(0)}} \right\}. \tag{4}$$

Substituting the second-quantized operator into H, we obtain an explicit formula using molecular integrals and orbital energies instead of matrix elements,

$$\begin{aligned}\left(H_{\text{eff}}^{(0-2)}\right)_{\alpha\beta} &= E_\alpha^{\text{CAS}}\delta_{\alpha\beta} - \sum_{pq,B}\langle\alpha|E_{pq}|B\rangle C_B(\beta)\sum_e \frac{u_{pe}u_{eq}}{\varepsilon_e - \varepsilon_q + \Delta E_{B\alpha}} \\ &\quad - \sum_{pqrs,B}\langle\alpha|E_{pq,rs}|B\rangle C_B(\beta)\left[\sum_e \frac{u_{pe}g_{eqrs}}{\varepsilon_e - \varepsilon_q + \varepsilon_r - \varepsilon_s + \Delta E_{B\alpha}}\right. \\ &\quad \left. + \sum_e \frac{g_{pers}u_{eq}}{\varepsilon_e - \varepsilon_q + \Delta E_{B\alpha}} + \frac{1}{2}\sum_{(a,b)} \frac{g_{parb}g_{aqbs}}{\varepsilon_a - \varepsilon_q + \varepsilon_b - \varepsilon_s + \Delta E_{B\alpha}}\right] \\ &\quad - \sum_{pqrstu,B}\langle\alpha|E_{pq,rs,tu}|B\rangle C_B(\beta)\sum_e \frac{g_{pers}g_{eqtu}}{\varepsilon_e - \varepsilon_q + \varepsilon_t - \varepsilon_u + \Delta E_{B\alpha}} + (\alpha \leftrightarrow \beta)\,,\end{aligned} \tag{5}$$

where u_{pq} and g_{pqrs} and are one and two electron integrals, respectively, and $\Delta E_{B\alpha}$ is the difference between the energies of the zeroth order state and configuration $E_B^{(0)} - E_\alpha^{(0)}$. The orbital labels $\{i, j\}$, $\{a\}$, and $\{e\}$ are for doubly occupied, active, and external orbitals, respectively, and $\{a', b'\}$ run over both active and external orbitals, and the suffix of the generator $\{p, q, r, s, t, u\}$ run over only active orbitals. The terms including doubly occupied orbitals are omitted in this equation. See Ref. [7]for the full formula. Note that the MRMP energy is also calculated with the formula by setting the number of the states to one.

The formula including doubly occupied orbitals might look tedious. However, the energy can be calculated just a sum of simple terms, hence rather simple. The computation is done with the coupling coefficient driven method. These coupling coefficients are sparse and can be pre-screened according to the condition,

$$\left(v_B^{pq\cdots rs}\right)_{\alpha\beta} = \langle\alpha|E_{pq,\ldots,rs}|B\rangle C_B(\beta) > \delta\,, \tag{6}$$

where $\delta = 1 \times 10^{-8}$ is usually sufficient to keep the energy accuracy better than 10^{-5} hartree. Thus the multiple summations for active orbitals in Eq. (6), which seemingly scales as the power of the number of active orbitals, are actually diminished considerably.

The computational time strongly depends on the systems calculated: in large CAS cases, the coupling coefficient computation occupies the most of CPU time, and in large basis set cases, integral transformation occupies the time. However, roughly, we can say that MR2D runs in memory, disk, and CPU time comparable to the CAS-SCF computation itself.

MR2D has —not only the option for CAS— also options for general configuration spaces.[9] For general configuration space cases, additional non-CAS terms appear as well as CAS terms, Eq. (6). The non-CAS terms may be calculated through

$$\left(H^{(2)}_{\text{non-CAS}}\right)_{\alpha\beta} = \mathbf{v}^T(\alpha)\,\mathbf{w}(\beta) \tag{2}$$

with

$$\mathbf{v}_I(\alpha) = \sum_{A\in \text{ref}} \langle I|H|A\rangle C_A(\alpha)\,, \tag{3}$$

$$\mathbf{w}_I(\beta) = \sum_{B\in \text{ref}} \langle I|H|B\rangle C_B(\beta) \Big/ \left(E^{(0)}_\beta - E^{(0)}_B\right). \tag{4}$$

The intermediate determinants/CSFs I are constructed by exciting one or two electron(s) from the reference determinants/CSFs within the active orbital space. In general, the number of I is not large, and thus they may be managed in computer memory.

Analytic gradients are not available in the current version.

2.3 TCE – Combination with Tensor Contraction Engine

UTChem also supports a spectrum of single-reference many-electron theories such as configuration-interaction theory (CISD, CISDT, CISDTQ), coupled-cluster theory (CCD, LCCD, CCSD, LCCSD, QCISD, CCSDT, CCSDTQ), and Møller-Plesset perturbation theory (MP2, MP3, MP4). To expedite the time-consuming and error-prone processes of deriving the working equations of these complex theories and implementing efficient computer programs on the basis of these equations, we have taken a unique approach --- we have developed a symbolic manipulation program (Tensor Contraction Engine or TCE)[1] that automated the both processes of the formula derivation and program synthesis. Provided a well-defined ansatz of a second-quantized many-electron theory, TCE performs valid contractions of creation and annihilation operators according to Wick's theorem, consolidates identical terms, and reduces the expressions into the form of tensor contractions acted upon by permutation operators. Subsequently, it determines the order of contractions having the minimal operation and memory cost, factorizes common contractions (defines intermediate tensors), and identifies reusable intermediates. The resulting ordered list of binary tensor contractions, additions, and index permutations is translated into an optimized program that is combined with UTChem. The programs synthesized by TCE take advantage of spin, spatial, and index permutation symmetries simultaneously to minimize the number of arithmetic operations and storage requirement, adjust the peak local memory usage by index range tiling, and support parallel I/O interfaces and dynamic load balancing for parallel executions. TCE not only expedites the daunting task of deriving and implementing various many-electron theories, but also enhances portability, maintainability, and extensibility of the synthesize program, facilitates parallelization and other laborious optimization of the programs which may be tailored to a particular computer architecture, and helps design and tests new many-electron theories which may be too complex to be hand-coded. Currently, the TCE-generated many-electron theories are limited to non-relativistic Hartree-Fock reference wave functions, but the extension to relativistic 2- and 4-component reference wave functions is underway.

2. 4 LMP2 – An MP2 Method with Localized Molecular Orbitals

Local MP2 (LMP2), which is based on the second-order Møller–Plesset perturbation theory, is an electron correlation method with localized molecular orbitals. Local correlation methods efficiently estimate the dynamical electron correlation in comparison with conventional canonical methods.

Using orthogonal localized occupied orbitals we have developed and implemented the local MP2 method based on the idea developed by Head-Gordon and coworkers. A subset of non-orthogonal correlation functions (the orbital domain) is assigned to each of the localized occupied orbitals using a distance criterion and excitations from localized occupied orbitals that are arranged into subsets. The correlation energy is estimated using a partial diagonalization and an iterative efficient method for solving large-scale linear equations. The orbital domain sizes are found to be independent of the molecular size, and the present local MP2 method covers about 98–99% of the correlation energy of the conventional canonical MP2 method.

Generally, the correlation energy of the second-order Møller-Plesset method is given as

$$E_2 = \frac{1}{4}\vec{\mathbf{V}}^{\mathrm{t}}\vec{\mathbf{T}} \qquad \left(\vec{\mathbf{T}} = \mathbf{\Delta}^{-1}\vec{\mathbf{V}}\right), \tag{5}$$

where the vector $\vec{\mathbf{T}}$ is the electronic excitation amplitude and vector $\vec{\mathbf{V}}$ is a vector described by the two-electron repulsion integrals as

$$V_{iajb} = \left(ia\middle|jb\right) - \left(ib\middle|ja\right). \tag{6}$$

The terms *i* and *j* denote occupied orbitals, and the terms *a* and *b* denote the non-orthogonal virtual orbitals. The matrix Δ is described by

$$\Delta_{iajb,\ i'a'j'b'} = (\mathbf{F_2})_{ia,i'a'}(\mathbf{S_2})_{jb,j'b'} + (\mathbf{S_2})_{ia,i'a'}(\mathbf{F_2})_{jb,j'b'}. \tag{7}$$

Using the overlap and Fock matrices, the matrices $\mathbf{F_2}$ and $\mathbf{S_2}$ in Equation (6) are described as follows

$$\begin{aligned} (\mathbf{F_2})_{ia,i'a'} &\equiv F_{ii'}S_{aa'} - S_{ii'}F_{aa'}, \\ (\mathbf{S_2})_{ia,i'a'} &\equiv S_{ii'}S_{aa'}. \end{aligned} \tag{11}$$

The non-iterative local MP2 method that was developed by Head-Gordon et al. has to simultaneously diagonalize both the $\mathbf{F_2}$ and $\mathbf{S_2}$ matrices to obtain the matrix Δ^{-1}. Since the side length of both of these two matrices is the number of occupied orbitals times the number of neighboring virtual orbitals, then these diagonalizations require a high computational cost for large-sized molecules.

In our improved method, this particular step is divided into two steps, which involve the diagonalization of the $\mathbf{S_2}$ matrix, and the iterative solving of the $\vec{\mathbf{T}}$ vector without diagonalizing matrix $\mathbf{F_2}$. Since $\mathbf{S_2}$ is a block diagonal matrix within subspaces consisting of same occupied orbitals, then it is easy to diagonalize it. After diagonalizing the $\mathbf{S_2}$ matrix, the vectors of small norms are excluded to avoid a linear dependence in the virtual space. In the next step, the $\vec{\mathbf{T}}$ vector is solved by an iterative method without diagonalizing matrix $\mathbf{F_2}$.

2.5 DFT – Density Functional Theory

Density functional theory (DFT) is a methodology that usually solves the Kohn-Sham (KS) self-consistent field (SCF) equation. In this program, as usual in packages of computational chemistry, the matrix algorithm of the KS method is implemented with a basis set.

The main characteristics of this DFT program are as follows;

1. Johnson's linear-scaling method[10] is implemented.
   Numerical calculations are carried out only for grid points where the adopted Gaussian-type basis functions provide inneglegible values.
2. A screening scheme is carried out for the weight and loop of grids.
   Stratmann-Scuseria-Frisch's weight and microbatch schemes of grid points[11] are also employed. Screenings are also performed for the values of Fock matrix element and electron density.
3. Calculational process is optimized for general contraction Gaussian basis functions.
4. OP correlation functional[12,13] and long-range exchange correction (LRXC) scheme[14,15] are implemented.
5. Data of grids are stored in core memory.

In this program, various types of functionals are available besides OP correlation.

2.6 REL4D – Relativistic Molecular Theory

One of the main features in the UTChem program package is to be able to consider the relativistic effects. The relativistic effect has been considered as an essential factor to figure out molecular structures, chemical activities, or various properties of heavy-element systems theoretically. Recently many quantum chemists have dedicated a lot of efforts to the calculation and treatment of the electronic structures of polyatomic systems including heavy elements, which are involved in many interesting chemical and physical phenomena. They still present unique difficulties to the theoretical study. Before recent years the relativistic effect had ever been thought less important for chemical properties because the relativity appears primarily in the core electrons, which had been believed to be unlikely to affect chemically active valence regions dramatically. Recent studies, however, have revealed not only quantitatively but also qualitatively that the relativistic effect plays essential and comprehensive roles in total natures of molecular electronic structures for heavy-element systems. We are nowadays convinced that the relativistic effect is definitely important for the accurate theoretical treatment of heavy-element systems as well as the electron correlation effect.

We have recently developed several relativistic molecular theories including four-component relativistic and approximate quasi-relativistic approaches. For the four-component relativistic approach, which is the rigorous relativistic treatment using the four-component spinors explicitly, our development starts from derivation of the analytical form of the basis spinor. In the four-component case we have to manage to get rid of the variational collapse coming from using the finite basis sets. We have proposed the efficient four-component basis spinors using the generally contracted kinetically balanced Gaussian-type spinors, which are taken from the analytical spinor forms corresponding to the one-electron hydrogenic system, and maintain the

variational stability through the kinetic balance condition. Furthermore, our contraction basis spinors have merits to give the correct behavior to the atomic limit, and require significantly less number of small-component basis spinors than the conventional decoupled scalar spin-orbitals used in the pioneering four-component programs MOLFDIR and DIRAC. We have formulated and implemented Dirac-Hartree-Fock (DHF)[16] and Dirac-Kohn-Sham (DKS)[17] methods, which have been made computationally feasible by exploiting the highly efficient integral method for two-electron repulsion integrals over the two-component basis spinors. Adoption[18] of the ACE formula proposed by Ishida[3] and the transfer relation proposed by Head-Gordon and Pople[19] make our algorithm efficient. It is shown that our four-component method is the most straightforward and efficient way by comparing with the other four-component relativistic methods and programs.

Even though we have spent much effort to develop the four-component method, still, the relativistic method using four-component spinors demands severe computational efforts to solve, and has a lot of difficulties to be applied to the large-size molecules. Thus, we have been proposed as the quasi-relativistic approximation instead of solving Dirac equation with the fully four-component spinors. Chemists' interest is devoted to the electronic solution, although the four-component calculations provide the eigenstates for not only electronic states but also positronic ones. The motivation of two-component approximation is to obtain the electronic states without treating the positonic solution explicitly. However, since the relativistic effect appearing in the electronic states is closely related to the positronic states in the four-component language, the problem of two-component method is how we can effectively retrieve the effect lost by neglecting the positronic solutions within the framework of the two-component Hamiltonian. In our recent works, we have developed the two kinds of two-component quasi-relativistic method using the RESC[20,21] and higher-order Douglas-Kroll (DK)[22,23] Hamiltonian. The RESC Hamiltonian has been derived from the eliminating small component equation with the relativistic substitutive correction. The higher-order DK Hamiltonian is the effective Hamiltonian obtained by transforming the Dirac Hamiltonian by an exponential-type unitary operator based on the philosophy of DK trans-formation[24,25] The exponential-type operator, which we have proposed, can simplify the formulation and implementation of the higher-order transformation by taking full advantage of the Baker-Cambell-Hausdorff expansion. Both two-component methods have the excellent features that they contain no singularity, therefore, and variationally treat the relativistic effect on the same footing as the electron correlation.

The resulting four-component relativistic and approximate quasi-relativistic codes have been packed in a program suite named REL4D, which is currently a part of the UTChem program package. The efficient ERI engine contributes to the performance for generating relativistic molecular integrals. In the self-consistent field (SCF) routine, several techniques have been implemented; the restricted open-shell treatment proposed by Davidson, the direct and conventional SCF methods, and the direct inversion in the iterative subspace (DIIS) method. The numerical grid-quadrature routine for the exchange-correlation potentials is implemented for our DKS scheme. The (SS|SS) integrals can be treated near the SCF convergence in the direct SCF approach.

2.7 S&D – Simulation and Dynamics

S&D is a program for executing molecular dynamics and Monte Carlo simulations. *Ab initio* molecular dynamics and Monte Carlo simulations can be carried out. Molecular dynamics simulations are carried out in microcanonical (NVE) ensemble. Verlet's algorithm and Gear's algorithm can be selected for integrating the equation of motion. *Ab initio* molecular dynamics can be performed with any approach in UTChem, in which the energy gradients are available. Monte Carlo simulations are performed in canonical (NVT) ensemble. In Monte Carlo simulation using *ab initio* energies, the energies are calculated in other programs. Simulated annealing and the replica exchange method (REM)[26,27,28,29] are also available for both classical and *ab initio* Monte Carlo simulations. In REM, a number of non-interacting replicas of the original system at different temperatures are simulated independently and simultaneously. REM is a powerful tool to obtain the global minimum in systems with many local minima, and difficult to optimize. Periodic boundary condition or spherical (image) boundary condition can be selected in the simulations.

The Quantum Mechanical (QM) / Molecular Mechanical (MM) method is available in UTChem. Simulation using QM/MM energies and gradients can be performed as well. QM/MM calculation can be carried out in Hartree-Fock methods and density functional theories. Geometry optimization using QM/MM is not available in the current version.

Acknowledgments. This research was supported in part by a grant-in-aid for Scientific Research in Specially Promoted Research "Simulations and Dynamics for Real Systems" from the Ministry of Education, Science, Culture, and Sports of Japan, and by a grant from the Genesis Research Institute. Tensor Contraction Engine is a product of Battelle and Pacific Northwest National Laboratory, and its development has been funded by the U.S. Department of Energy, the Division of Basic Energy Science, Office of Science under contract DE-AC06-76RLO 1830 with Battelle.

References

1. Webpage of the Tensor Contraction Engine (TCE) project is placed on http://www.cis.ohio-state.edu/~gb/TCE/.
2. T. H. Dunning Jr., J. Chem. Phys. 90, 1007 (1989).
3. J. Almlöf and P. R. Taylor, J. Chem. Phys. 86, 4070 (1987).
4. K. Ishida, Int. J. Quantum Chem. 59, 209 (1996); K. Ishida, J. Chem. Phys. 109, 881 (1998); K. Ishida, J. Comput. Chem. 19, 923 (1998); K. Ishida, J. Chem. Phys. 111, 4913 (1999).
5. T. Yanai, K. Ishida, H. Nakano, and K. Hirao, Int. J. Quantum Chem. 76, 396, 2000.
6. Global Aarray (GA) toolkit developed in Pacific Northwest National Laboratory, http://www.emsl.pnl.gov:2080/docs/global.
7. H. Nakano, J. Chem. Phys. 99, 7983 (1993); H. Nakano: Chem. Phys. Lett. 207, 372 (1993).
8. K. Hirao, Chem. Phys. Lett. 190,374 (1992); K. Hirao, Chem. Phys. Lett. 196, 397 (1992); K. Hirao, Intern. J. Quantum Chem. Symp. 26,517 (1992).

9. H. Nakano, J. Nakatani, and K. Hirao, J. Chem. Phys. 114,1133 (2001); H. Nakano, R. Uchiyama, and K. Hirao, J. Comput. Chem. 23, 1166 (2002).
10. C. A. White, B. G. Johnson, P. M. W. Gill and M. HeadGordon, Chem. Phys. Lett. 253, 268 (1996).
11. R. E. Stratmann, G. E. Scuseria and M. J. Frisch, Chem. Phys. Lett. 257,213 (1996).
12. T. Tsuneda, T. Suzumura and K. Hirao, J. Chem. Phys. 110, 10664 (1999).
13. T. Tsuneda, T. Suzumura and K. Hirao, J. Chem. Phys., 111, 5656 (1999).
14. H. Iikura, T. Tsuneda, T. Yanai, and K. Hirao, J. Chem. Phys. 115,3540 (2001).
15. M. Kamiya, T. Tsuneda and K. Hirao, J. Chem. Phys.,117, 6010 (2002).
16. T. Yanai, T. Nakajima, Y. Ishikawa, and K. Hirao, J. Chem. Phys. 114, 6526 (2001).
17. T. Yanai, H. Iikura, T. Nakajima, Y. Ishikawa, and K. Hirao, J. Chem .Phys. 115, 8267 (2001).
18. T. Yanai, T. Nakajima, Y. Ishikawa, and K. Hirao, J. Chem. Phys. 116, 10122 (2002).
19. M. Head-Gordon and J. A. Pople, J. Chem. Phys. 89, 5777 (1988).
20. T. Nakajima and K. Hirao, Chem. Phys. Lett. 302, 383 (1999).
21. T. Nakajima and K. Hirao, J. Chem. Phys. 113, 7786 (2000).
22. T. Nakajima, T. Suzumura, and K. Hirao, Chem. Phys. Lett. 304, 271 (1999).
23. T. Nakajima and K. Hirao, Chem. Phys. Lett. 329, 511 (2000).
24. M. Douglas and N. M. Kroll, Ann. Phys. (N.Y.) 82, 89 (1974).
25. B. A. Hess, Phys. Rev. A 33, 3742 (1986).
26. K. Hukushima and K. Nemoto, J. Phys. Soc. Jpn. 65, 1604 (1996).
27. C. J. Geyer and E. M. Keramidas (Ed.), Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface, Interface Foundation, Fairfax Station 156 (1991).
28. R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. 57, 2607 (1986).
29. Y. Ishikawa, Y. Sugita, T. Nishikawa, Y. Okamoto, Chem. Phys. Lett. 333, 199 (2001).

Local Correlation Models

Martin Head-Gordon, Troy Van Voorhis, Gregory J.O. Beran, and Barry Dunietz

Department of Chemistry, University of California, and,
Chemical Sciences Division, Lawrence Berkeley. National Laboratory,
Berkeley, CA 94720, USA.
mhg@cchem.berkeley.edu

Abstract. Local correlation models that meet the requirements of a theoretical model chemistry are discussed. Two types of models are considered. The first class uses a valence active space that associates one correlating orbital for each occupied valence orbital. In these models the fundamental quantity is the electron pair; even the simplest local approximation (perfect pairing) exactly treats an isolated electron pair in this active space. The second class of models uses no active space. In this case we argue that the most appropriate fundamental quantity is the atom; even the simplest local approximation (atoms-in-molecules) exactly treats an isolated atom.

1 Introduction

Theoretical chemical models are systematic approximations to exact quantum mechanics that traditionally involve two approximations. The first one is a limited treatment of electron correlation. Within wavefunction-based theory this has evolved into a hierarchy that goes from Hartree-Fock (HF) theory (no correlation) to MP2 theory (perturbative 2-electron substitutions), to singles and doubles coupled cluster theory (CCSD; self-consistent 2-electron substitutions), and finally to perturbative inclusion of 3-electron correlations via CCSD(T). The second approximation, of course, is the selection of a limited atomic orbital expansion space.

While the correlation treatments mentioned above are convergent in accuracy (at least in most usual applications), they increase in cost as M^5 (MP2), M^6 (CCSD), and M^7 (CCSD(T)), as a function of molecule size M. It is clearly highly desirable to reduce these high computational cost scalings. As electron correlation is primarily a short-range phenomenon, arising from the electron-electron cusp, this is possible by imposing spatial truncations, forming the basis of low-scaling local correlation methods. The modern era of local correlation research was initiated with the work of Pulay, and Pulay and Sæbø [1], with subsequent important extensions by many others.

In principle, local correlation approximations can be made at a sufficiently conservative level that numerical results (and thus the theoretical chemical model) are unaffected. In practice, however, this approach is probably too conservative, and thus, led by the ideas of Pulay and Sæbø, much work has been performed on local correlation

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 96–102, 2003.
© Springer-Verlag Berlin Heidelberg 2003

approaches that recover a large percentage, though not all, of the correlation energy of a given approximated theory. Great numerical success has been achieved, including linear scaling of computation with molecule size [2]. However, the Pulay-Sæbø approach fails to be a well-defined theoretical chemical model, because it depends on one or more numerical parameters to define which occupied orbitals are correlated by which unoccupied orbitals. Connected with this issue, it fails to yield properly continuous potential energy surfaces, and therefore has not found major application in geometry optimizations.

In this article we discuss alternative local correlation models that do satisfy the requirements of a well-defined theoretical model chemistry. In our opinion, these are:

i) The theory should be uniquely defined for any nuclear configuration without any adjustable parameters, such as thresholds, cutoff lengths, or selection criteria, that affect the result.

ii) The theory should yield potential energy surfaces that are strictly continuous so that chemical reactions can be studied.

iii) The theory should yield additive energies when applied to non-interacting systems (size consistency property).

iv) The theory should possess a reasonable degree of accuracy and feasibility. Clearly more accurate models tend to be less feasible for application to large molecules.

2 Valence Active Space Models

We choose to begin our discussion of local correlation models by assuming that electron correlation is treated not in the full correlating space of all empty (virtual) orbitals, but rather in a limited space of valence orbitals that are divided into a nominally occupied set and a matching set of nominally empty orbitals [3]. The occupied and empty orbitals are paired such that there is a 1:1 association between each occupied orbital j and each corresponding empty (correlating) orbital j^*. As a result an isolated electron pair can be exactly described by a 2-configuration wavefunction, where the two orbitals, and the excitation amplitude, are to be optimized.

The important thing is that a valence active space of this type (or simple generalizations that associate 2 or 3 or n correlating orbitals with each occupied orbital) partitions the orbital set into spatially localized functions that describe individual electron pairs separately without repetition or duplication. This contrasts with the situation when no active space is introduced: then the same virtual functions are usually called on to describe intrapair correlations in a number of nearby pairs.

The simplest valence active space local correlation model is the one that treats intra-pair correlations exactly, and at the same time completely neglects all interpair excitations. This is the perfect pairing (PP) model; where the number of 2-electron substitutions is just linear in the size of the system:

$$\hat{T}_{PP} = \sum_{i}^{\substack{valence \\ pairs}} t_i a_{i^*}^{\dagger} a_{\bar{i}^*}^{\dagger} a_{\bar{i}} a_i \qquad (1)$$

In the context of coupled cluster theory, it was first explored by Cullen in the chemical literature [4]. PP can also be generalized to permit nonorthogonal orbitals [5], which can overcome some symmetry-breaking artifacts for molecules with multiple resonance structures. Note that the PP model is defined completely without parameters. Furthermore all orbitals are optimized to minimize the energy. Physically the merit of the PP model is that it exactly describes the making and breaking of any (or each) individual electron-pair bond within the valence active space. As molecules are collections of interacting electron pairs, this is clearly a logical starting point for local correlation modeling.

The next logical level of local correlation must include 2-electron excitations that also couple together different electrons pairs. 2-electron excitations that couple together two electron pairs without net transfer of electrons are the leading contributions of this type. Such excitations are responsible for the long-range van der Waals interactions between separated electron pairs. It is likely also to be the dominant part of short-range pair-pair correlations also. The model which adds these correlations has been somewhat whimsically defined as "imperfect pairing' (IP) [6]. It increases the number of 2-electron correlations from linear in the PP model (intra-pair only) to quadratic, and thus also necessarily increases the computational complexity. At the present stage [7], its computational cost scales approximately with M^3, which is roughly the square root of growth of the full theory including all 2-electron correlations in the valence active space.

It remains somewhat less certain as to how best to define a third level of local correlation modeling in the valence space. At present we are exploring the inclusion of inter-pair correlations that account for the two possible types of single electron transfers between pairs, as a result of correlated fluctuations. The first type, which might be called "charge coalescence", begins with separate electrons in occupied levels of two different pairs, and excites them into the correlating orbital of just one pair. The second type, "charge splitting", begins with both electrons in the occupied level of a single pair and promotes them into two correlating levels one of which is on the same pair, and the other on a different pair. This level of theory could be defined as "singly ionic pairing" (SIP).

$$\hat{T}_{SIP} = \hat{T}_{IP} + \sum_{ij}^{pairs} \left\{ t_{i\bar{i}}^{i^*\bar{i}^*} a_{i^*}^{\dagger} a_{\bar{i}^*}^{\dagger} a_{\bar{j}} a_i + t_{i\bar{i}}^{i^*\bar{j}^*} a_{i^*}^{\dagger} a_{\bar{j}^*}^{\dagger} a_{\bar{i}} a_i \right\} \tag{2}$$

An alternative is to additionally include 2-electron transfers between electron pairs or doubly ionic pairing (DIP).

2-electron excitations also exist that couple together more than two electron pairs. In particular the most general excitations couple together 4 pairs. We shall not concern ourselves with 3-pair couplings in this work. We shall use results based on the full 2-electron substitution operator as a benchmark against which to test the simpler approximations later. This full theory is then valence optimized doubles (VOD) [8].

3 Full Space Models

We turn next to the formulation of local correlation models for the electron correlation operator when no active space is in use. The localized orbitals spanning the full virtual space are very effectively defined (that is with very small fractional redundancy) by projecting the atomic orbitals (AO's) into the virtual space with **1**-**P**, where **P** is the 1-particle density matrix. This, for a large molecule, automatically gives atom-centered functions spanning the virtual space, with the same locality properties as **P** (which of course is exponentially localized for insulators). Furthermore, these functions are also atom-labeled as they derive from the atom-labeled AO's via a direct mapping. So, while the virtual functions in the case of the valence active space partitioned naturally amongst the electron pairs (one per pair), here they partition naturally by atoms.

This suggests that we should build up a set of local correlation methods starting with an atomic partitioning of the orbitals and thus the correlation contributions. To make such a partitioning possible, we also need to partition the occupied orbitals amongst the atoms. Except for such uninteresting special cases as noble gas clusters, an atomic partitioning is not possible with orthogonal orbitals because bonding occupied orbitals are intrinsically shared between atoms. Therefore an over-complete (and thus non-orthogonal) set must be used to describe the occupied space. The least over-complete set that we could use is a minimal atomic orbital basis, which gives each atom a number of functions equal to the sum of its core and valence atomic orbitals. While these orbitals could be defined using some independent definition (such as "extracted polarized atomic orbitals") it is clearly best if they are defined to minimize the local correlation energy, just as the occupied and virtual functions defining the valence active space were optimized in the previous section.

The simplest full space local correlation model can now be defined as "atoms-in-molecules" (AIM). First, we express the correlation operator (for whatever method we are developing the local correlation model) in the combined minimal occupied and full virtual atomic representation discussed above. Then all correlation amplitudes that are localized entirely on a single atom are retained, while all correlation amplitudes coupling together two or more atoms are discarded. This model is manifestly exact for an isolated atom (or an ensemble of isolated atoms), and is the most drastic local correlation that is sensible.

Note that no parameters enter, and the AIM model meets all the criteria of a well-defined theoretical model chemistry. No results have been reported with the AIM model hitherto, and so, in the following section, one of our main objectives is to assess its performance on a judiciously chosen model problem. There are only a linear number of degrees of freedom in the AIM model, and so high-performance low-scaling implementations should be possible. This work is in progress for MP2 and orbital-optimized CCD (OD), for which the correlation operator then takes the form:

$$\hat{T}_{AIM} = \sum_{A}^{atoms} \left\{ \sum_{i\in A}^{occ} \sum_{j\in A}^{occ} \sum_{a\in A}^{virt} \sum_{b\in A}^{virt} t_{ij}^{ab} a_a^\dagger a_b^\dagger a_j a_i \right\} \tag{3}$$

We view AIM as the most reasonable analog of PP for the case when correlation is allowed in the full space.

The next logical level of local correlation modeling is to include those correlation amplitudes that couple together pairs of atoms, but without permitting net electron transfer between them. At the level of 2-electron excitations (correlations), this model, "diatomics-in-molecules" (DIM) fully recovers long-range correlation. It has already been explored [9] and quite efficiently implemented (M^3 scaling) for large-scale MP2 calculations. However the minimal atomic orbitals were not fully optimized, which is desirable in future work.

More complete 2-center local correlation models can also be defined, but has not been explored hitherto, primarily because of implementation challenges that have not yet been fully overcome. Disregarding practical challenges for the present, it is logical to next include the two classes of two-center terms responsible for single electron transfers between atoms. There were previously referred to as "charge coalescence and charge splitting in the previous section. This model will also be assessed in the following section via calculations on a toy system. Applied to double (2-electron) excitations, it includes all 2-center amplitudes apart from those that excite 2 electrons from one atom to another. They are omitted because they are clearly associated with basis set superposition effects: two electrons from one atom are correlating their motion by "borrowing" virtual functions from another atom.

Three-center models are also possible, and we have explored them with some success in other work [9]. However we shall not consider them further here.

4 Calculations on a Toy Model

To provide some assessment of how the different local correlation methods discussed in the previous sections perform, we shall consider some pilot calculations on a simple toy model system. It consists of 8 H_2 molecules, each arranged at the center of the sides of an octagon. Each intra-molecular H-H distance has its standard value of 0.74 Å, while the intermolecular spacing is variable. The 6-31G basis was used for the calculations, and the calculations were performed with test codes that were attached to the Q-Chem program package [10].

The results of local MP2 calculations using various local models discussed above are presented in Table 1, as a function of intermolecular spacing. The quantity reported is the percentage of the full MP2 correlation energy recovered by each local correlation method. We note that all local models are exact for isolated H_2 molecules, and thus the interesting regime is the transition from highly localized electronic structure (nearly non-interacting H_2 molecules) towards more and more de-localized electronic structure. At the same time, in terms of electron correlation we are changing from well-separated electron pairs completely dominated by intra-pair correlation towards less and less distinct sets of pairs, where inter-pair correlations are becoming stronger in both absolute and relative senses. So, while these are toy calculations, they can still pose a challenging test for local correlation methods.

Table 1. Local MP2 calculations using the AIM, DIM, and DIM+SI models. The results are expressed as percentage recovery of the full MP2/6-31G correlation energy for this system, which is a ring of hydrogen molecules, with various intermolecular spacings. To roughly characterize the electronic structure of the system at each separation, the bandwidth and bandgaps obtained with canonical Hartree-Fock orbitals are also reported. All calculations used a minimal basis of modified atomic orbitals to span the occupied space

R /Å	1.9	1.5	1.2	1.0	0.8
E_{HOMO}-E_1 /eV	2.72	5.58	9.63	13.88	20.11
E_{LUMO}-E_{HOMO} /eV	19.27	17.74	15.37	12.73	8.76
AIM /%	95.87	91.23	83.22	74.94	65.28
DIM /%	99.36	98.73	97.82	96.82	95.12
+ singly ionic /%	99.84	99.49	98.85	98.21	97.31

A quite rapid convergence of the results towards 100% is seen for this hierarchy of 3 models. The very strongly truncated AIM model still recovers clearly the leading contribution of electron correlation. It is quite possible that this will be a useful starting point for the development of local coupled cluster methods. The DIM model then recovers a substantial majority of the correlation neglected by the AIM model, substantiating its value as the next level of model. Singly ionic pair terms in turn cover a majority of the remaining discrepancy in this model system. Roughly we estimate that stable molecules might correspond to roughly between 1.2 and 1.5 in this toy system.

Table 2. Local valence coupled cluster doubles calculations using the PP, IP, SIP and DIP models. The results are expressed as percentage recovery of the full VOD/6-31G correlation energy for this system, which is the same as used for Table 1. All calculations used the PP orbitals

R /Å	1.9	1.5	1.2	1.0	0.8
PP /%	97.9	93.9	85.9	75.6	60.0
IP /%	99.1	96.4	90.6	82.4	75.4
SIP /%	99.9	99.3	97.1	92.6	82.8
DIP /%	100.0	99.5	98.0	94.8	87.4

In Table 2, we summarize a similar series of calculations on exactly the same systems, but this time using valence coupled cluster methods truncated with the local correlation models discussed in Sec. 2. These results are generally similar to those reported in Table 1. PP, containing just a linear number of degrees of freedom, nevertheless recovers the leading correlation contributions. IP is perhaps slightly less successful than DIM at correcting for the missing correlations, although it is still clearly a significant improvement. The singly ionic (SIP) and doubly ionic (DIP) pairing models add successively smaller improvements.

Overall, the results in Tables 1 and 2 provide a useful quantification of the performance of the local correlation models discussed here for a simple but illustrative

model system. We hope to present additional calculations and studies that further characterize these methods in due course, as well as algorithmic developments that permit efficient implementation.

Acknowledgments. This work was supported by a grant from the Department of Energy, Office of Basic Energy Sciences, SciDAC Computational Chemistry Program (Grant No. DE-FG02-01ER403301).

References

1. Saebo, S.; Pulay, P.: Local Treatment of Electron Correlation. *Annu. Rev. Phys. Chem.* 44 (1993) 213–236
2. Schutz, M.; Werner, H. J.: Low-order scaling local electron correlation methods. IV. Linear scaling local coupled-cluster (LCCSD). *J. Chem. Phys.* 114 (2001) 661–681
3. Goddard, W. A.; Harding, L. B.: Description of Chemical Bonding from Ab initio Calculations. *Annu. Rev. Phys. Chem.* 29 (1978) 363–396
4. Cullen, J.: Generalized valence bond solutions from a constrained coupled cluster method. *Chem. Phys.* 202 (1996) 217–229
5. Van Voorhis, T.; Head-Gordon, M.: A nonorthogonal approach to perfect pairing. *J. Chem. Phys.* 112 (2000) 5633–5638
6. Van Voorhis, T.; Head-Gordon, M.: The imperfect pairing approximation. *Chem. Phys. Lett.* 317 (2000) 575–580
7. Van Voorhis, T.; Head-Gordon, M.: Implementation of generalized valence bond-inspired coupled cluster theories. *J. Chem. Phys.* 117 (2002) 9190–9201
8. Sherrill, C. D.; Krylov, A. I.; Byrd, E. F. C.; Head-Gordon, M.: Energies and analytic gradients for a coupled-cluster doubles model using variational Brueckner orbitals: Application to symmetry breaking in O-4(+). *J. Chem. Phys.* 109 (1998) 4171–4181
9. Lee, M. S.; Maslen, P. E.; Head-Gordon, M.: Closely approximating second-order Moller-Plesset perturbation theory with a local triatomics in molecules model. *J. Chem. Phys.* 112 (2000) 3592–3601
10. Kong, J.; White, C. A.; Krylov, A. I.; Sherrill, D.; Adamson, R. D.; Furlani, T. R.; Lee, M. S.; Lee, A. M.; Gwaltney, S. R.; Adams, T. R.; Ochsenfeld, C.; Gilbert, A. T. B.; Kedziora, G. S.; Rassolov, V. A.; Maurice, D. R.; Nair, N.; Shao, Y. H.; Besley, N. A.; Maslen, P. E.; Dombroski, J. P.; Daschel, H.; Zhang, W. M.; Korambath, P. P.; Baker, J.; Byrd, E. F. C.; Van Voorhis, T.; Oumi, M.; Hirata, S.; Hsu, C. P.; Ishikawa, N.; Florian, J.; Warshel, A.; Johnson, B. G.; Gill, P. M. W.; Head-Gordon, M.; Pople, J. A.: Q-Chem 2.0: A high-performance ab initio electronic structure program package. *J. Comput. Chem.* 21 (2000) 1532–1548

Multiresolution Quantum Chemistry in Multiwavelet Bases

Robert J. Harrison[1], George I. Fann, Takeshi Yanai, and Gregory Beylkin[2]

[1] Oak Ridge National Laboratory,
P.O. Box 2008 MS6367,
Oak Ridge, TN 37831-6367,
harrisonrj@ornl.gov,
http://www.ornl.gov/~rj3
[2] Department of Applied Mathematics,
University of Colorado at Boulder,
UCB 526, Boulder, CO 80309-0526,
gregory.beylkin@colorado.edu,
http://amath.colorado.edu/faculty/beylkin/Home.html

Abstract. Multiresolution analysis in multiwavelet bases is being investigated as an alternative computational framework for molecular electronic structure calculations. The features that make it attractive include an orthonormal basis, fast algorithms with guaranteed precision and sparse representations of many operators (e.g., Green functions). In this paper, we discuss the multiresolution formulation of quantum chemistry including application to density functional theory and developments that make practical computation in three and higher dimensions.

1 Introduction

Chemists are familiar with the range of length-scales that are present in molecular electronic wave functions. There is a sharp cusp at the nucleus; atoms have core and valence regions; molecules also possess inter-atomic bonding regions; and both atoms and molecules have a long exponential tail. By capturing the essence of molecules as being composed of perturbed atoms, the linear combination of atomic orbitals (LCAO) approximation has proven tremendously successful. Part of this success stems from effective management of the length scales present in the problem. The atomic orbital basis functions naturally incorporate both the short- and long-range behavior. However, the non-local nature of atomic wave functions, especially in high-precision calculations, gives rise to inefficiencies, non-physical density matrices, and numerical problems due to severe linear dependence. Furthermore, the atomic orbital basis sets must be carefully designed and calibrated [1]. If used out of context (e.g., in charge-fitting [2] or resolution-of-the-identity methods [3,4]) uncontrolled errors can be introduced.

Multiresolution analysis in multiwavelet bases [5,6,7,8] is an alternative approach that is potentially very attractive for computational chemistry. Our approach closely follows that described by Alpert et al. [5]. Among the notable features are

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 103–110, 2003.
© Springer-Verlag Berlin Heidelberg 2003

- a fully orthonormal basis with high-order convergence even in the presence of singularities,
- an algorithm for adaptive refinement,
- sparse representations of many operators including the Hamiltonian and Green functions, and
- fast algorithms with guaranteed precision for many common operations.

The high-order, orthonormal basis with adaptive refinement enables robust high-precision computation. The separation of length scales implicit in the multiresolution formulation provides sparse representation of operators which enables fast computation. Although the orthonormal multiwavelet basis may be used in standard matrix (i.e., Hamiltonian) formulations of quantum problems, it is often much more effective to reformulate the problem as an integral equation.

In the following, we use the Poisson equation to introduce the essential details of multiresolution in multiwavelet bases and how they result in fast algorithms. Subsequently, we discuss an essential component for efficient computation in many dimensions, which is accurate separated representations of integral operators. This is done in the context of an integral formulation of density functional theory (DFT). Finally, we briefly describe our prototype implementation.

2 Sparse Representations and Fast Algorithms

Much of our discussion is, for simplicity, in one dimension, but the generalization to higher dimensions is straightforward, except where otherwise noted.

We map the problem to the unit interval. This is recursively divided by two, so that on each level $n = 0, 1, 2, \ldots$ there are 2^n boxes. In each box, we define basis functions that are the first k normalized Legendre polynomials inside the box and are zero outside. Smooth functions may be represented with an error $O(2^{-nk})$. By construction, the basis on a level is orthonormal and the functions associated with different boxes do not overlap each other. This last property of disjoint support is the most important advantage of multiwavelet bases. If boundaries or discontinuities are located at the nodes, high-order convergence may be maintained.

The basis at a given level of refinement is of dimension $k2^n$ and is labeled V_n^k. It satisfies

$$V_0^k \subset V_1^k \subset \cdots \subset V_n^k \subset \cdots. \tag{1}$$

The multiwavelet subspace (W_n^k) is defined as the orthogonal complement of V_n^k in V_{n+1}^k

$$W_n^k = V_{n+1}^k - V_n^k \tag{2}$$

which then leads to an alternative decomposition of our basis

$$V_n^k = V_0^k + W_0^k + W_1^k + \cdots + W_{n-1}^k. \tag{3}$$

Beylkin et al. give fast $(O(N))$ algorithms for converting between the two representations. Adaptive local refinement while maintaining a global norm-wise

error is accomplished by discarding small wavelet coefficients [5]. We presently use Alpert's definition of the multiwavelet basis [7].

Since V_n^k is the space of polynomials on level n, the first k moments vanish for functions in W_n^k (since they are by construction orthogonal to V_n^k). This property gives rise to sparse representations of smooth functions and operators. We use the Poisson equation in three dimensions

$$\nabla^2 u = -4\pi\rho \tag{4}$$

with free-space boundary conditions ($u(\infty) = u'(\infty) = 0$) as an example of how this enables us to replace iterative solution of differential equations with fast application of integral operators.

The matrix representation of the Laplacian in either the multiwavelet basis or standard discretizations is badly conditioned. Moreover, the largest eigenvalues are associated with the highest frequencies. This leads to the requirement of good preconditioners for the efficient iterative solution of differential equations. However, the Green's function for the Poisson equation is known and we may immediately write the solution as

$$u(r) = \int \frac{\rho(s)}{|r-s|}\, ds. \tag{5}$$

That the multiwavelet representation of this integral operator is sparse can be readily seen from the multipole expansion of $1/r$. Consider the interaction between two wavelets of order k in two boxes separated by r. Since the first k moments vanish, the interaction will decay as $r^{-(2k+1)}$. It is also necessary to consider the interaction between wavelets and polynomials, which will decay at worst as $r^{-(k+1)}$. We commonly employ wavelets of order 5–13, and, by increasing the order of the wavelets as we increase the required precision, it is never necessary to include interactions beyond the first 26 nearest neighbors of a box. This immediately results in an $O(N)$ algorithm with guaranteed precision. Also, the iterative solution of equations has been replaced with a single, fast, sparse matrix-vector product.

Some additional insight is provided by the telescoping series [5,6]

$$T_n = T_0 + (T_1 - T_0) + (T_2 - T_1) + \cdots + (T_n - T_{n-1}) \tag{6}$$

where T_n is the projection of an operator (or equivalently a multidimensional function) onto the polynomial basis at level n. This identity expresses an operator on the fine scale (level n) in terms of the operator on a coarse scale and successive differences between length scales. The representation of the operator at all levels may be dense, as is the case for $1/r$. However, if the operator is smooth at long range, then it will eventually be accurately represented at some scale, and thereafter the differences between levels will be small. With this awareness, it is often possible to compute efficiently and accurately without explicit introduction of the wavelet basis.

Alpert et al. [5] discusses a much wider range of differential and integral operators, including time dependent problems, for which efficient sparse representations may be constructed for arbitrary finite precision.

3 Integral Formulation of the Schrödinger Equation

The familiar Hamiltonian formulation of the Schrödinger equation

$$\left(-\frac{1}{2}\nabla^2 + V\right)\psi = E\psi \tag{7}$$

may be rewritten as an integral equation

$$\psi = -2\left(-\nabla^2 - 2E\right)^{-1} V\psi \tag{8}$$

where the inverse of the differential operator denotes application of the corresponding Green function. For bound states, the Green function is that of bound-state Helmholtz equation, which, in three dimensions, is given by

$$G(r, s, E) = \frac{e^{-\sqrt{-2E}|r-s|}}{4\pi|r-s|}. \tag{9}$$

This kernel is amenable to fast computation in the multiwavelet basis just as described for the Poisson kernel. Moreover, straightforward iteration of (8) will converge to the ground state, whereas an iterative, preconditioned eigensolver must be used to solve the differential form. For DFT models without Hartree-Fock exchange, the form of the integral operator includes the expected asymptotic exponential decay. To extract multiple roots from the integral equation we use deflation which is discussed elsewhere [8].

4 Practical Application of Integral Operators in Three Dimensions

Above, the application of integral operators has been informally demonstrated to be efficient, in the sense of scaling linearly with the number of boxes with significant coefficients. However, practical computation also requires a small prefactor. In three dimensions, the basis within each box on the locally finest level will be a tensor product of Legendre polynomials. The coefficients associated with the box will therefore carry three-indices. Similarly, a matrix representation of an operator will carry six indices; three for the source and three for the result. So it seems as if cost of both storing and applying the operator scales as $O(k^6)$, where k is again the order of the wavelet. This is prohibitively expensive since we commonly use wavelets of order 5–13.

Since the basis is orthonormal, the matrix representation of an integral operator with kernel (K) in the scaling function (Legendre polynomial) basis can be obtained by projection

$$r^{n,ll'}_{pp',qq',rr'} = \int \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z\,\mathrm{d}x'\,\mathrm{d}y'\,\mathrm{d}z'\; K(x-x', y-y', z-z') \\ \phi^{nl}_p(x)\phi^{nl'}_{p'}(x')\phi^{nl}_q(y)\phi^{nl'}_{q'}(y')\phi^{nl}_r(z)\phi^{nl'}_{r'}(z') \tag{10}$$

where l and l' label the result and source boxes, and $\phi_i^{nl}(x)$ is the i'th Legendre polynomial in box l on level n. If there existed a separated representation of the kernel that is accurate to our desired precision ϵ

$$K(x,y,z) = \sum_{i=1}^{M} \sigma_i X_i(x) Y_i(y) Z_i(z), \tag{11}$$

it is straightforward to see that each dimension may be separately applied. This reduces the cost of applying the operator to $O(Mk^4)$ and reduces the storage to $O(Mk^2)$. The computational cost may be further reduced by using singular value decomposition to exploit the low operator rank away from the singularity. The length of the expansion (M) is referred to as the *separation* rank. It is critical that this rank be as small as possible, and also that the coefficients in the expansion be small and of the same sign so as to avoid loss of precision.

Recently [9], it has been shown that many physically significant operators, including the Poisson kernel and even the entire many-electron Schrödinger Hamiltonian, have low separation rank that increases only logarithmically with both the precision and dimension. This hints at the possibility of alternative schemes for efficient computation in many dimensions. In the present context, it ensures the existence of efficient representations but it does not tell us how to construct them.

Beylkin et al. [10] have developed a powerful approach for constructing optimal separated representations of functions over finite ranges, but this is too complicated to discuss here. However, near-optimal representations may be formed from application of quadrature rules to integral representations of kernels as a sum of Gaussians

$$K(r) = \sum_{i=1}^{M} c_i e^{-t_i r^2}. \tag{12}$$

For instance, we have the standard identity

$$\frac{e^{-\mu r}}{r} = \frac{2}{\sqrt{\pi}} \int_0^\infty e^{-r^2 t^2 - \frac{\mu^2}{4t^2}} dt \tag{13}$$

which with the substitution $t = e^s$ can be transformed to

$$\frac{e^{-\mu r}}{r} = \frac{2}{\sqrt{\pi}} \int_{-\infty}^\infty e^{-r^2 e^{2s} - \frac{\mu^2}{4} e^{-2s} + s} ds. \tag{14}$$

The advantages of this form are that the integrand dies very rapidly at both ends of the quadrature range so the trapezoidal rule converges geometrically, and that the required resolution (bandlimit) is almost independent of r. The resulting Gaussian exponents form a geometric series ensuring a uniform error at all length scales. The form of the integrand may be analyzed to establish a suitable range and empirical tests suffice to find the required resolution for given precision. The results are

$$s_{hi} = \log(T/r_{lo}^2)/2 \tag{15}$$

$$s_{lo} = -\log(4T/\mu^2)/2 \tag{16}$$

$$1/h = .2 + 0.47 \log_{10} \epsilon \tag{17}$$

where the range of the quadrature is $[s_{lo}, s_{hi}]$, the expansion has a relative precision of ϵ over the range $[r_{lo}, 1]$, and h is the spacing of the quadrature points. The parameter T is empirically determined to be 5, 10, 14, 18, and 24, respectively for accuracies 1e-2, 1e-4, 1e-6, 1e-8, and 1e-10. However, because of the super-exponential decay, the number of quadrature points is only weakly dependent upon T, so using the optimal value of T saves at most only 1 or 2 points. The quadrature may be modified to provide compact representations of the Poisson kernel ($\mu = 0$). To illustrate the automatic procedure, fits were generated for $\exp(-30r)/r$ over [1-e8,1] for accuracies 1e-10, 1e-8, 1e-6, 1e-4 and 1e-2. The errors are plotted in figure 1.

5 Prototype Implementation

In the current implementation, multiwavelets of any order are supported. The two-scale coefficients are generated using Alpert's algorithm [7] in extended precision floating point numbers in Python (e.g., 156-bit arithmetic is used to generate the two-scale coefficients for order 10). The extended precision is necessary only to generate the coefficients which are stored for subsequent use. Standard double precision arithmetic is used for all other operations.

Our initial implementation uses Python for high-level control and C/Fortran for computationally intensive operations including matrix transformations, quadratures and the innermost loops. At the highest level, we have defined a `Function` class that includes methods for evaluation, compression, reconstruction, addition, multiplication by a function or scalar, differentiation, application of the Laplacian, and other operations. The operator overloading capabilities of Python provide great expressivity and enable very compact programs. For instance, if `psi` is an instance of the `Function` class representing an orbital, and similarly `V` represents the potential, then the following statement applies the Fock operator to the orbital[1]

```
Hpsi = -0.5 * (Delsq * psi) + V*psi
```

`Delsq` is an empty class that is never instantiated. If an instance of `Function` is multiplied on the right by `Delsq`, the function's Laplacian method is invoked. Evaluation of a function at a point with the natural semantics `psi(x,y,z)` is accomplished by overloading the function call operator.

6 Conclusions

Multiresolution in multiwavelet bases is an attractive alternative framework for computation in chemistry. It provides fast algorithms with guaranteed precision.

[1] Note that matrix elements of the Laplacian, including evaluation of the kinetic energy, should employ the variational form $< f|\nabla^2|g >= - < \nabla f.\nabla g >$.

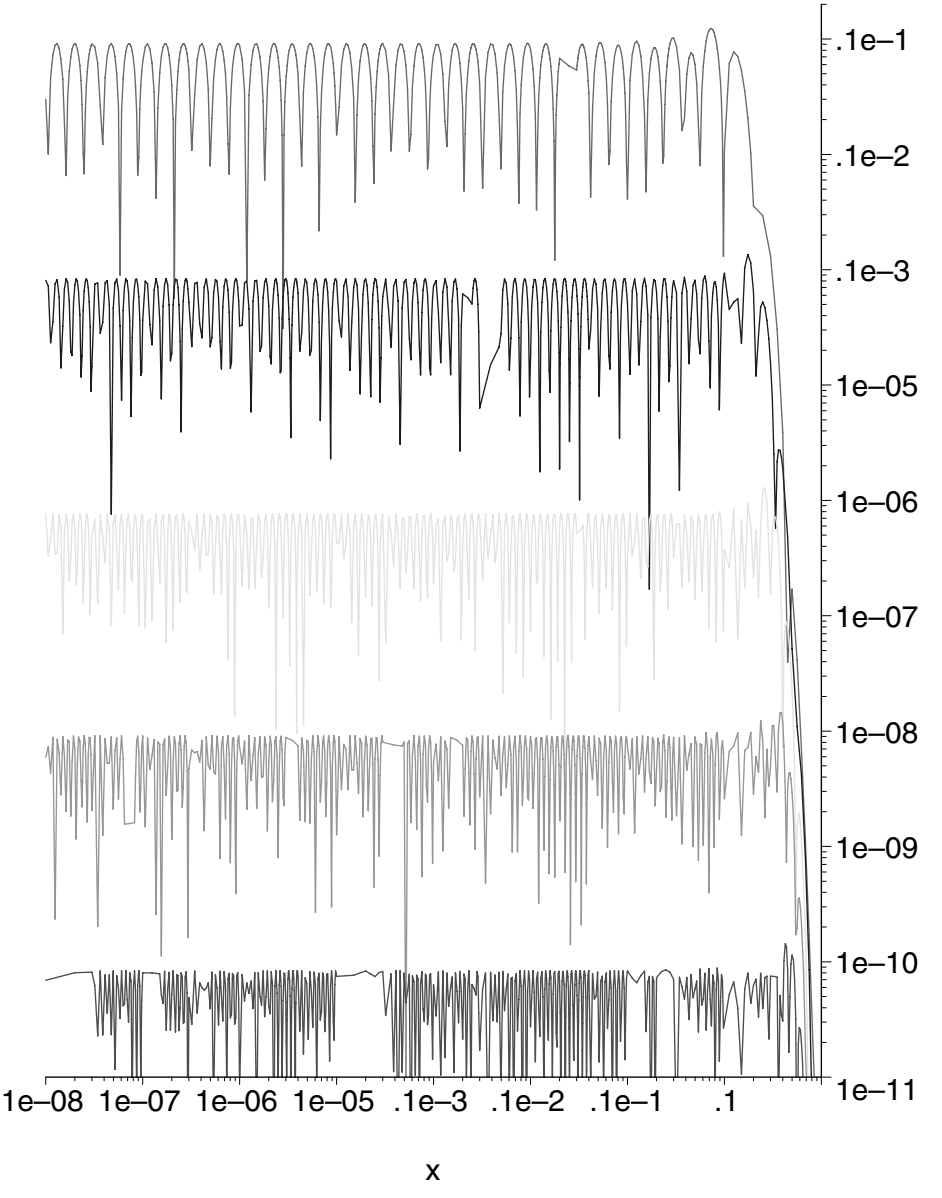


Fig. 1. Automatically generated representations of $\exp(-30r)/r$ accurate to 1e-10, 1e-8, 1e-6, 1e-4, and 1e-2 (measured by the weighted error $r(\exp(-30r)/r - \mathrm{fit}(r))$) for r in [1e-8,1] were formed with 92, 74, 57, 39 and 21 terms, respectively

The orthonormal basis enables all of the familiar results of quantum theory to be used without change, though the greatest benefit is obtained by reformulating the entire solution scheme to take advantage of the properties of the basis. In particular, the sparse representation of many integral operators, including many physically significant Green functions, enables new approaches. Additional techniques are necessary for scattering states for which the corresponding Green functions do not become smooth at long range. Compact separated representations are important for efficient computation in higher dimensions, and may also be of utility in conventional Gaussian-based methods. Finally, although our current implementation uses orbitals, so cannot attain full linear scaling, it is well established [11,12,13] that a density-matrix approach in a wavelet basis will achieve linear scaling while maintaining a guarantee of arbitrary, finite precision.

References

1. Dunning, Jr., T.H., Peterson, K.A., and Woon, D. E.: Gaussian Basis Sets for Use in Correlated Calculations, in Encyclopedia of Computational Chemistry, Ed., P. v. R. Schleyer, (John Wiley & Sons Ltd., 1997)
2. Dunlap, B.I., Connolly, J.W.D., and Sabin, J.R.: J. Chem. Phys. **71** (1979) 3396 and 4993
3. Vahtras, O., Almlöf, J., and Feyereisen, M. W.: Integral approximations for LCAO-SCF calculations. Chem. Phys. Lett **213** (1993) 514; Feyereisen, M., Fitzgerald, G., and Komornicki, A.: Chem. Phys. Lett. **208** (1993) 359
4. Challacombe, M., Schwegler E., and Almlöf, J.: pp 53–107 in Review of Current Trends, ed. J. Leczszynski (World Scientific, 1996)
5. Alpert, B., Beylkin, G., Gines, D., and Vozovoi, L.: Adaptive Solution of Partial Differential Equations in Multiwavelet Bases, J. Comp. Phys. **182** (2002) 149–190
6. Beylkin, G., Coifman, R., and Rohklin, V.: Fast wavelet transforms and numerical algorithms, I, Comm. Pure. Appl. Math. 44 **141** (1991) 183
7. Alpert, B.K.: A Class of Bases in L^2 for the Sparse Representation of Integral Operators, SIAM J. Math. Anal. **24** (1993) 246–262
8. Harrison, R.J., Fann, G.I., Yanai, T., and Beylkin, G.: Multiresolution Quantum Chemistry, in preparation, 2003.
9. Beylkin, G. and Mohlenkamp, M.J.: Numerical operator calculus in higher dimensions, Proc. Nat. Acad. Sci. **99** (2002) 10246–10251
10. Beylkin, G. and Monzon, L., work in progress.
11. Li, X.-P.; Nunes, R.W. and Vanderbilt, D.: Density-matrix electronic-structure method with linear system-size scaling, Phy. Rev. B **47** (1993) 10891
12. Beylkin, G., Coult, N. and Mohlenkamp, M.J.: Fast Spectral Projection Algorithms for Density-Matrix Computations, J. Comp.Phys. **152** (1999) 32–54
13. Goedecker, S.: Linear Scaling Electronic Structure Methods, Rev. Mod. Phys. **71** (1999) 1085

Management of Web and Associated Grid Technologies for Quantum Chemistry Computation

Kim K. Baldridge [1] and Jerry P. Greenberg[1]

[1] San Diego Supercomputer Center
9500 Gilman Drive
La Jolla, CA 92037-0505
{kimb,jpg}@sdsc.edu

Abstract. We present a description of our ongoing efforts to utilize the services provided over the World Wide Web to advance the field of computational quantum chemistry utilized for detailed study of the electronic structure and properties of molecular systems, as well as the dissemination of results both for research and education. There are several facets of these efforts: (1) development of a computational chemistry web portal (2) development of an XML schema based on output data of electronic structure software, and (3) development of database and associated query tools that will serve as a basis for storage, retrieval, and manipulation of QM data in uniquely new ways. These efforts are distinctly coupled together through our creation of a "workflow" of web services, using grid resources.

1 Introduction

Advances in computational science technologies continually present researchers with new opportunities for solving problems. However, at the same time, such technologies often present a bewildering set of choices for computing paradigms, and additionally can result in overwhelming amounts of data to process. While computational resources have grown in power and size, they have associated complexity issues to negotiate. Thus, while one can now obtain unprecedented speed, memory and storage capacities, there is an associated plethora of platforms, with different operating systems, queuing mechanisms, directory structures, and access methods. We have recently described the use of web service technology to facilitate quantum chemistry (QC) calculations, shielding the user from some of these complexities[1]. Here, we expand on those efforts with an emphasis on the design of XML schema.[2]

A parallel effort in the development of QM database resources is intended to store, in a systematic manner, data obtained from QC calculations. Such a database can serve several purposes, including, a) providing a source for querying chemical properties over a range of chemical compounds, b) comparing new results with previous calculations, or levels of theory, or c) accessing accurate data for chemical analysis, or for use in one of a variety of empirical models. Such inquiries are somewhat difficult now since results of QC calculations are often stored as "flat" files that are human readable only, or at best, parsed by shell scripts. Comparisons become difficult not only because of different storage formats, but also because of

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 111–121, 2003.
© Springer-Verlag Berlin Heidelberg 2003

accessibility: researchers may not know about previous calculations, and it may not be clear how to compare data.

The process of building a QM-DB and working with data mining and query experts, enables one to not just build a database of chemical quantities, but provides a tool that accommodates extensive manipulation of the quantum data. Specifically, access to of query capabilities on top of a rigorous QM database offers the ability to query data in a ways that were unavailable, or, even more so, not thought of due to the inaccessibility – one can now imagine cross coupling data types that can answer deeper chemical questions than previously enabled, or construction of creative theoretical methods that span molecular scale.

2 The GAMESS Portal

Our first effort to utilize web services involved the creation of a GAMESS (General Atomic and Molecular Structure System)[3] portal (gridport.npaci.edu/gamess). GAMESS is an *ab initio* electronic structure code in which the standard release reads and writes conventional input and output files. The most basic input file (.inp) consists of atomic coordinate data and associated symmetry information, basis set specification, and keywords describing the type of calculation(s) to perform. The output consists of files containing virtually all user information pertaining to the actual computational process and associated results (.out), as well as a data summary file (.dat) that may be used for data analysis, visualization and/or for starting new runs based on the data from the old. The initial motivation for the GAMESS portal was to isolate users from the complexities of the computational grid by providing a standard user interface for accessing input and output, running jobs on one of a variety of platforms without logging onto those platforms, and the ability to transfer data among various platforms.

The portal has facilities for processing the output of a particular run via visualization with our computational chemistry visualization and analysis tool, QMView. However, the major focus to date has been facilitation of the processes of computing and storing data. The existing portal enables facile job submission to remote platforms, and retrieval of results via any commodity browser. More recently, storage for input and output data has been enabled through connections to the SDSC Storage Research Broker (SRB). [4] The SRB additionally provides the abstraction mechanisms needed to implement data grids, digital libraries, and persistent archives for data sharing, data publication, and data preservation. Thus, things like the quantum mechanical database project as well as other planned digital libraries of QM data and images, can easily involve more sites (institutions) as the library grows. The SRB will register digital entities of any format into a logical name space to create global, persistent identifiers in the digital library. The SRB is currently used in production or being evaluated across multiple projects at NSF, NASA, DOE, DOD, NIH, NLM, NARA, and the Library of Congress. Our existing portal invokes the persistent archiving capability of the SRB to store its digital entities from the GAMESS portal.

The underlying basis for the chemistry portals is the Gridport Tool Kit [5] which in turn provides an interface to the GLOBUS toolkit [6] for secure transfer of files and execution of remote procedure calls to any high performance computing (HPC) resource. The GAMESS portal submission page is illustrated in Figure 1.

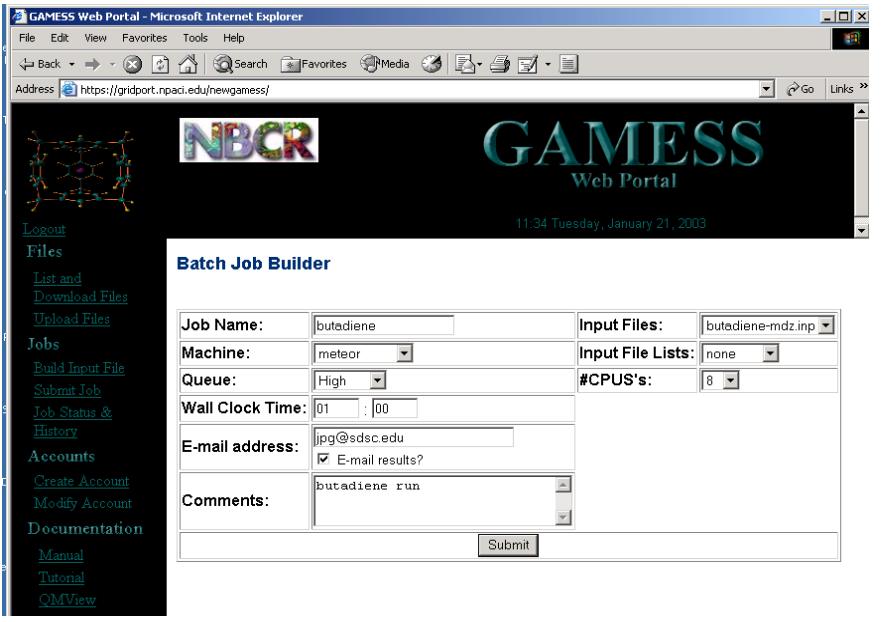


Fig 1. The submit page from the GAMESS PORTAL

One can choose an input file from a list of previously uploaded files, select the remote resource to run on, and choose from such options as the queue name, number of processors, and maximum time. The portal informs the user of job status and enables results to be stored in the SRB in a specific user data collection space.

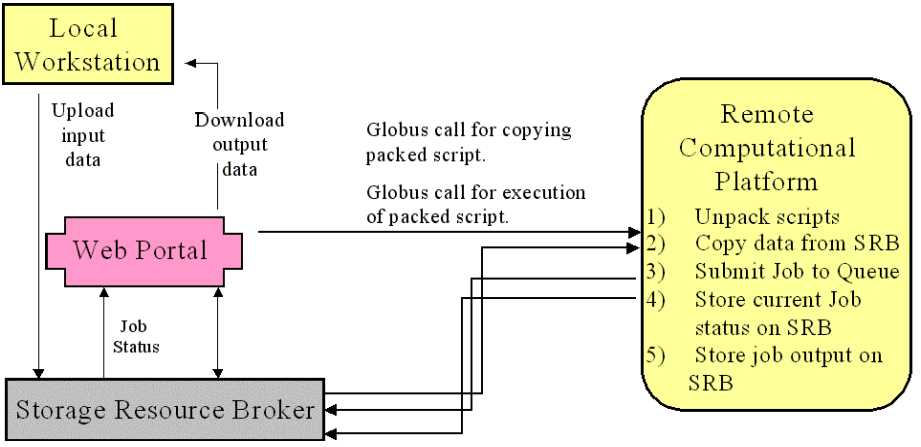


Fig. 2. Flow chart illustrating the operation of the GAMESS portal.

Because of the general need for secure communication (authenticated and perhaps confidential communication) between the elements of a computational Grid, we have incorporated Grid Security Infrastructure (GSI) [7]. Other major incentives for including such security within the portal include the need to support security across organizational boundaries (thereby eliminating the need for a centrally-managed security system), and the need to support "single sign-on" for users of the Grid, including designation of authorization for computations that involve multiple resources, Thus, for example, authentication for retrieving and copying files from a remote platform to the SRB is accomplished using this technology, as illustrated in Figure 2.

The design of the SDSC science portals has undergone several renovations as we leverage what we have learned in the construction of the GAMESS portal in order to "portalize" other computational chemistry and biochemistry software. One of the most notable extensions came with collaborative work in conjunction with Pevzner and his group at the Bioinfomatics Lab at UCSD. Their work involved the development of the "EULER" [8] software, a new technique for genetics sequencing. The construction of a portal for their application enabled important enhancements to our existing portal in general, as a result of the specialized needs of extending this application as a service to a general community on a few nodes of a commodity cluster. Portal enhancements included a) the optimization of high throughput for job submissions, b) improved formatting and presentation of the output, and c) enhanced facilities for downloading results.

Portal optimization for high throughput job submissions was accomplished primarily by reducing the number of remote procedure calls to the remote resource. Such calls were reduced to one call to copy a shell script and one call to execute the script. The shell script "unpacks" several additional scripts, places them in the appropriate directory, and then executes them appropriately. The unpacked scripts retrieve input data from the SRB and submit the job to the queue. In order to eliminate calls to the remote machine that inquire about job status (e.g., queued, running, or finished status), the running batch script sends a file containing the current job status directly to the SRB. The web portal then copies and reads this file in order to report job status. The resulting procedure is much less costly in terms of time.

An example of the portal output data page is given in Figure 3. QMView (see below) may be registered as a recognized MIME type in order to view output files by clicking on them in the browser. In addition, there are facilities for running other programs with the selected files as input. Files may also be viewed within the browser or, one or more files may be selected for downloading in one of several archiving formats. In addition, an error reporting facility has been added to send examples of bugs to the portal developers. Users may submit reports of bugs along with all relevant files.

In order to facilitate access to clusters by users who may not have an account on those systems, we have taken advantage of the sub-account system within the gridport toolkit in order to allow users who have a portal account to run jobs on a cluster under a master account. The "owners" of the portal may add sub-accounts from the portal master account. The mapping of the portal accounts to the master account on the remote cluster is handled by Globus. This has been implemented within the Euler portal, where users may now run jobs on a small dedicated cluster. We intend to extend these service-oriented capabilities to other applications, including GAMESS.

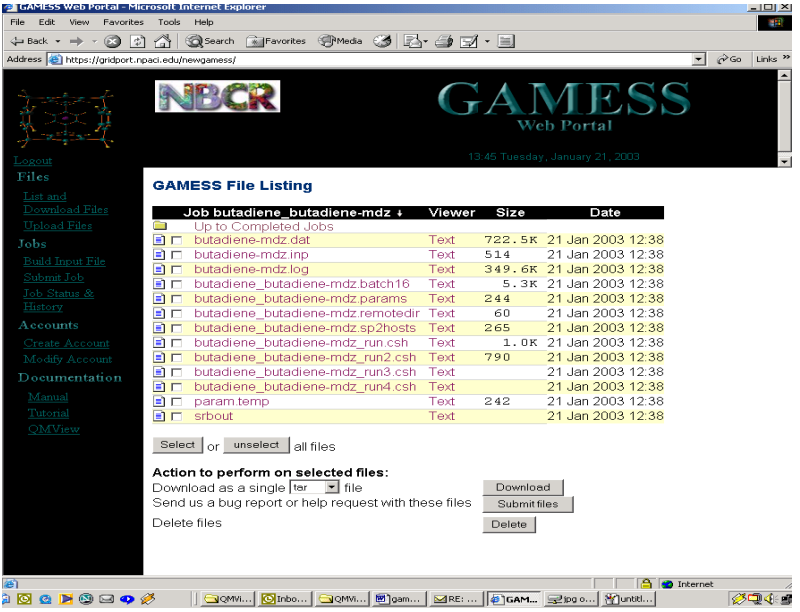


Fig. 3. The GAMESS Portal Ouput Page

2.1 Visualization and Analysis Extensions to the Portal

The discussion so far has centered on the mechanism of running a job and has not dealt at all with handling and processing of the data. For these latter capabilities, we have made initial efforts to produce structured output documents directly from the *ab initio* software, GAMESS. As previously mentioned, there are many advantages of structured data over "flat files" of data. The most commonly used format for accessing such data is via Extensible Markup Language, XML. XML is rapidly becoming the industry standard for transferring data between web services [9] and actually facilitates not only data analysis, but also transfer, storage and retrieval.

There has been previous work in creating schemas specifically for chemical data such enveloped in Chemical Markup Language, CML [10]. We have created our own schema whose focus is on quantum chemistry computational data but could in principle be modified and incorporated into the CML schema using the CML data elements. For now, the data element names and structure resemble the output of a GAMESS job. That is, at least at the initial stages, the elements and attributes are specifically related to *ab initio* quantum mechanical calculations, and to GAMESS in particular. This schema is being implemented directly within the GAMESS software, and additionally within our visualization and analysis program, QMView, which is described next. The goal is to have a workflow of XML documents for a variety of functional purposes, including the generation of input and output and the deposit of output into a database that will subsequently be available for querying.

Currently, the graphical interfaces that are available for *ab initio* electronic structure software are limited in terms of computational flexibility and in graphics output capabilities. We have already written packages including QMView, (Quantum

Mechanical Viewing tool) [11],[12] and MDMovie (Molecular Dynamics Movie [13], that have addressed these needs, in the former case for quantum chemistry, and the latter, for molecular dynamics. The software is available under the OpenGL™ graphics library, which allows our interactive 3-D software to run on many types of workstations and personnel computers.

It is the nature of QM calculations that a relatively small amount of data is required for input to the software, but large volumes of data (e.g., 3-dimensional molecular orbital, vibrational, electrostatic, and dynamical data) are output. Our initial efforts in the development of QMView were devoted to the visualization of calculated quantum mechanical data. The continued development of visualization now enables one to generate images for purposes that range from publication, to creation of illustrative animations of, say, reaction pathways, dynamics and vibrational phenomenon. Analysis capabilities encompass a broad range of manipulation of structure, property and differencing techniques. Modifications and revisions are made continually based on our own scientific needs, as well as those of the community of users (Figure 4).

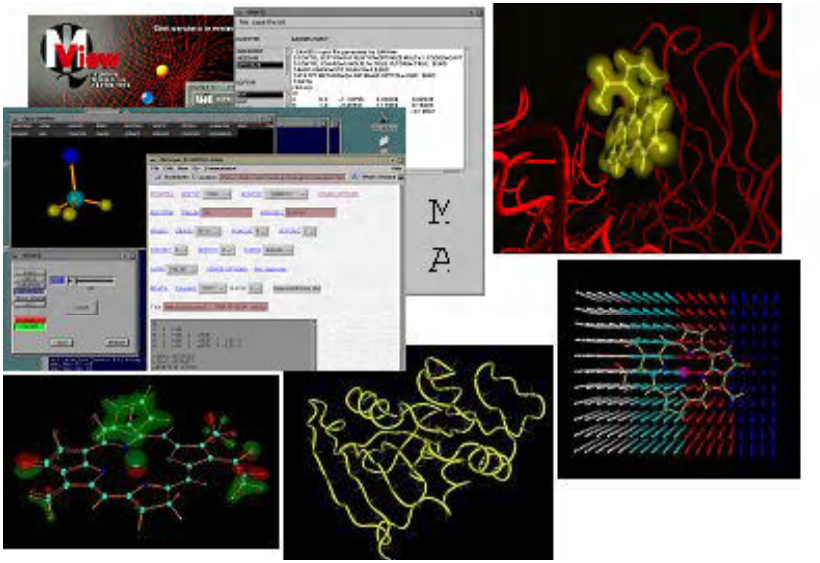


Fig. 4. A Sample of QMView interface and output type.

More recently, QMView is evolving into a distributed computational chemistry environment for building, simulating, and analyzing chemical structures, as well as a central controller for access to other resources (e.g., databases, auxiliary programs etc). In the context of the GAMESS portal, and as a part of our overall project of incorporating grid technologies into quantum chemistry applications, QMView is developing into a front-end for accessing several services including, for example:

a) Protein DataBank (PDB): Extracting the coordinate data of biopolymers and ligands from the PDB, (e.g, GAMESS input, display of volumetric data . [14]

b) Compute Engines (e.g., GAMESS, G98, MOPAC): Calculation of coordinate and volumetric data, and subsequent visualization and analysis.
c) QM-Database (QM-DB): Extraction of data from QM-DB, followed by analysis, correlation, or computation of new fragments and deposit back into the QM-DB.
d) WebViewing: Exportation of XML files of chemical data from QMView for DB storage, manipulation, and/or viewing within a browser.
e) Differencing Tools: Analysis of volumetric data, including a variety of differencing schemes and capabilities.
f) Grid-Workflow Environment: Serves as a full working discovery environment for computational chemistry research, with access to a full set of resources.

2.2 Portal Technology Transfer

GAMESS was the first computational science web portal built using the SDSC GridPort Toolkit (http://gridport.sdsc.edu/). The software development package consists of standard, portable technologies with which developers can quickly create Web interfaces to scientific applications. Such application portals can then be used to securely access high-end computing resources from any Web-connected device, including wireless handheld devices. Any application portal built on this toolkit automatically inherits these "HPC anywhere" capabilities. Therefore, we have used what we have learned in developing the GAMESS portal to facilitate other applications. Typically we have found that enhancements or needs of any particular application are quite easily propagated to a new portal environment from the GAMESS portal base.

We now have a variety of useable computational (bio)chemistry and bioinformatics portals that we are offering to the community through SDSC and the National Computational Biomedical Resource. In addition to GAMESS and EULER, these include the APBS (Adaptive Poisson-Boltzmann Solver) [15] portal for calculation of electrostatic potential surfaces for molecular constructs, CE (Combinatorial Extension) [16]portal for performing structural comparison of proteins, Cardiac Modeling [17]portal for electrocardiogram modeling of the heart, and the LAPK portal for Population Pharmacokinetics[18].

While these types of portals have lifted considerable complexity from the attention of the user, it is clear that once this basic portal technology is enabled for an application, one begins to envision more functionality that would further enhance science capabilities. In particular, interoperability within an application, between one application and another, or even between one or more applications and other resources would be the next goal. Thus, in a more general instantiation of these technologies, we are in the process of designing “workflow environments” that will enable more advanced working environments for scientific discovery. These designs, as was true for the portal designs, will be suitable for a wide variety of scientific applications. These efforts are outlined elsewhere [1], and so here we will concentrate on the general relevance and applicability of web services, and the use of the computational grid for the GAMESS software.

3 Advancements towards Work-Flow and Data Sharing

The most common way to extract data from output files is by processing them with command language scripts. Extraction of data from output files using scripts eliminates the need to incorporate the data extraction program into the simulation program. However there are significant disadvantages to the use of common language scripts. A small change in output format can result in the script to fail or a misinterpretation of data. Significant changes in output format may require rewriting the entire script. By placing structured data output facilities within the program, the effort and code expended on data parsing is removed, and the software becomes independent of the format of the "human readable" output file.

As a step in this direction, we initially embedded calls within the FORTRAN routines of the GAMESS program, which prints out an XML document according to a Document Type Definition, DTD. More recently, we have devised an a schema for this purpose.

An effort has been made to put the minimal amount of code into the GAMESS FORTRAN modules for this purpose. The code that outputs the XML document is embedded in a C library that is easily linked to the GAMESS executable. For example, the code fragment below shows how the atomic coordinates are written to the XML document:

```
call output_coords_init(UNITS,4,AORIG,8,IXCOUNT)
call energyxml(ETOT,EUNITS,ILEN,IXCOUNT)
DO 310 IAT=1,NAT
      IF(SKIP(IAT)) GO TO 310
      ZNUC = ZAN(IAT) + IZCORE(IAT)
      X = UNIT * C(1,IAT)
      Y = UNIT * C(2,IAT)
      Z = UNIT * C(3,IAT)
C       IF (MASWRK) WRITE(LU,9030) ANAM(IAT),BNAM(IAT),ZNUC,X,Y,Z
      IF (MASWRK) call output_coord(ANAM(IAT),BNAM(IAT),ILEN,ZNUC
   1                          ,X,Y,Z)
 310 CONTINUE
```

The call to "output coords_init" sets up the "SYSTEM_STATE" element, which we refer to again below, while the call to "energyxml" prints out the energy. The loop is over all atoms in the molecule and prints the "ATOM_POSITION" elements. The coordinates and energy are stored in common blocks that already exist in the GAMESS program. For run types that generate multiple coordinate sets (such as an Internal Reaction Coordinate run, or a molecular optimization run), the address of the starting point of the molecular orbital data is passed to a C routine. In the case of a run with no coordinate updates, the molecular orbital coefficients are stored in dynamic memory allocated by the C XML library, and then printed out when the coordinates are printed.

3.1 Schema

The schema in its initial implementation defines a document for storing only basic data, such as atomic coordinates, atom types, energies and molecular orbitals. In principle, this is enough data to evaluate further properties of a molecular system. However, in practice and for convenience, we plan to add other system properties such as gradient and hessian information, solvation information, and other such data, that could as well be evaluated from the essential data.

The data that characterizes a particular run is initially specified by including the job type (RUNTYP), wavefunction type (SCFTYP), the basis set (BASIS), and symmetry information. All choices for run specification type (e.g., energy, optimization, hessian, etc), wavefunction type (e.g., closed shell, RHF, open shell, UHF, etc), basis sets (e.g., 6-31G(d,p), cc-pVDZ, etc), and symmetry (CNV, DND, etc.) are enumerated. Where possible, error checking is being incorporated into the schema. For example, limits specifying the correct choices for parameters such as NGAUSS (number of Gaussians per atomic orbital), correct numeric values for NDFUNC (number of 'd' type polarization functions for heavy atoms), and/or job type combinations can be monitored.

The output data follows with the coordinate data along with all properties associated with that data (e.g., energy, etc.), grouped into an "system_state" set. These properties include even the molecular orbital coefficients for that coordinate set, and in the future will include the vibrational mode vectors. In the case of several groups of coordinate data sets, such as in the case of reaction coordinate runs, each set is specified in the appropriate order. This schema is obviously still under development to add more data types. A full illustration of the schema is provided at http://www.sdsc.edu/~jpg/nmi/.

An example XML document for even a small system can be quite lengthy. A full illustration of such a document for a very basic GAMESS input is illustrated at http://www.sdsc.edu/~jpg/nmi. We plan to use the XML documents generated by GAMESS for several purposes. One, which we have briefly mentioned, is to provide a single structured input for our visual interface QMView. Such a document will provide a single access point for all data produced by GAMESS computations. In turn, QMView will be able to generate XML documents as well as "flat" input files for GAMESS computations. We plan to utilize the JAVA based "Castor" [19] project to "unmarshall" the XML document data from XML documents to JAVA objects. Subsequently, one could imagine using such document data in QMView for processing, or perhaps to prepare new input files for generating several GAMESS runs in sequence (e.g. minimization followed by vibrational analysis). The glue that we plan to use to bind data extracted by the XML documents by the Castor JAVA classes to QMView is provided by the JAVA Native Interface (JNI) [20]. Such an interface will allow us to load the JAVA classes directly into QMView (written in C) .

For purposes of utilizing the computational grid, the GAMESS XML documents will provide the basis for transferring data from remote resources to a user from one or more jobs scattered across more then a single resource. This format of data will be ideal for depositing results into remote databases, and/or for creating new instances of GAMESS jobs that will be launched across the grid. The data used for subsequent runs may also be obtained by querying databases, in which case the XML document is the best format for the retrieved data.

Our concurrent project involving the construction of a rigorous federated QM database will serve as a focal point for creating computational chemistry jobs. The database will contain the data addressed by our XML schema along with other calculated properties. As with the GAMESS output, we will bind QMView to the database with JNI. One could imagine exploiting such a database to extract common properties within a single molecular structure or across a group of compounds. For example, given the presence of a 3-D grid of electrostatic data, one could query for certain features such as zero flux surfaces. Interactive queries could also be done on a portion of space occupied by a structure, such as a reactive ligand, and surface volumes in that region could be displayed. Across several compounds, trends for certain properties (structural feature, dipole moment, or vibrational frequency) could be searched and correlated.

4 Conclusions

Computational modeling in the era of grid technology is still very challenging and often confined to specialized research groups that are set up with all the key components and tools. Much of the success in these studies have been despite the difficulties involved in integrating all of the technologies, software, and other tools necessary to answer the scientific questions of interest. Additionally, the current technical software available for general utility is typically limited in computational flexibility, graphics and database capabilities, and access, particularly for the remote user. The effort involved in keeping abreast of latest advances in computer technology, the explosive growth in the size and types of chemical databases, and new visualization and analysis tools, all of which offer new computational solutions for increasingly complex problems, requires an inordinate commitment of time on the part of a researcher who may not be well versed in one or more of the resource areas, nor may they feel obliged to take the time away from research to learn.

Our present goals involve the creation of tools at the interface of molecular electronic structure and associated informatics tools, cheminformatics, and grid technologies, to develop a “one-stop” set of web services to offer the research and education communities. The work described here is ongoing and all the component tools are by no means at their final stages. The extensibility of XML makes it relatively easy to add new data types, making it possible to query and display a wide range of data. Coupled with the Grid Workflow environment, which is also currently under development in this group, efforts expended to the collection of data, running of jobs, analysis, visualization, and dissemination of results will be greatly reduced. The resulting set of tools and resources should enable new thought processes in scientific discovery, as well as be quite important for education and training in the classroom.

Acknowledgements. The authors would like to acknowledge support from the NSF through ANI-0223043, and the NIH through NBCR-RR08605 (NIH).

References

[1] Baldridge, K. K., Greenberg, J. P., Elbert, S. T., Mock, S. ,Papadopoulos, P., "QMView and GAMESS: Integration into the World Wide Computational Grid," presented at Proceedings of Supercomputing 2002, 2002.

[2] Fallside, D. C., XML Schema Part 0: Primer, W3C Recommendation,

[3] Schmidt, M., Baldridge, K. K., Boatz, J. A., Elbert, S., Gordon, M., Jenson, J. H., Koeski, S., Matsunaga, N., Nguyen, K. A., Su, S. J., Windus, T. L., Dupuis, M., Montgomery, J. A.:The General Atomic and Molecular Electronic Structure System, J. Comp. Chem., vol. 14, 1993 pp. 1347–1363.

[4] Rajasekar, A. K.,Wan, M., "SRB and SRBRack- Components if a Virtual Data Grid Architecture," presented at Advanced Simulation Technologies Conference, SD, CA, 2002. .

[5] Thomas, M., Mock, S., Dahan, M., Mueller, K., Sutton, D. ,Boisseau, J. R., "The Gridport Toolkit: a System for Building Grid Portals," presented at 10th IEEE International Symp. on High Perf. Comp, 2001.

[6] Foster, I.,Kesselman C.:Globus: A Metacomputing Infrastructure Toolkit, Intl. J. Supercomputing Applications, vol. 11, 1997 pp. 115–128

[7] Perlman, L. W. V., Foster I.,Kesselman C., Tuecke S.:A Community Authorization Service for Group Collaboration, Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002

[8] Pezvner, P. A., Tang H., Waterman M.S.:An Eulerian path approach to DNA fragment assembly, Proc. Natl. Acad. Sci., vol. 98, 2001 pp. 9748–9753

[9] Ewald, T., Understanding XML Web Services The Web Services Idea, http://msdn.microsoft.com/webservices/understanding/readme/default.aspx

[10] Murray-Rust, P., Rzepa, H. S. ,Wright, M.:Development of chemical markup language (CML) as a system for handling complex chemical content, New Journal of Chemistry, 2001 pp. 618–634

[11] Baldridge, K. K.,Greenberg, J. P.:QMView: A Computational 3D Visualization Tool at the Interface Between Molecules and Man, J. Mol. Graphics, vol. 13, 1995 pp. 63-666

[12] Baldridge, K. K.,Greenberg, J. P.:QMView as a SupramolecularVisualization Tool, in Supramolecular Chemistry, J. Siegel, Ed.: Kluwer Academic Publishers, 1995, pp. 169–177.

[13] Greenberg, J. P.:MDMovie: a Molecular Dynamics Viewing Tool, J. Mol. Graph., vol. 14, 1996 pp. 287–290

[14] Jimenez, J., Salazar, G., Baldridge, K. K. ,Romesberg, F. E.:Flexibility and molecular recognition in the immune system, Proc. Natl. Acad. Sci., vol. 100, 2003 pp. 92–97

[15] Baker, N. A., Sept, D., Joseph S., Hoist J. M.,McCammon, J. A. J.:Electrostatics of nanosystems: application to microtubules and the ribosome, Proc. Natl. Acad. Sci., vol. 98, 2001 pp. 10037–10041

[16] Shindylov, I.,Bourne, P.:Protein structure alignment by incremental combinatorial extension (CE) of the optimal path, Protein Eng., vol. 11, 1998 pp. 739–747

[17] McCulloch, A., Continuity 6.0

[18] Laboratory for Applied Pharmokinetics, http://www.usc.edu/hsc/lab_apk

[19] Exolab Group, Castor 0.9.4.2

[20] Liang, S., The JavaTM Native Interface: Programmer's Guide and Specification, 1 ed: Addison Wesley Longman, Inc, 1999.

The Extensible Computational Chemistry Environment: A Problem Solving Environment for High Performance Theoretical Chemistry

Gary Black, Karen Schuchardt, Debbie Gracio, and Bruce Palmer

Pacific Northwest National Laboratory
Richland, Washington 99352
{gary.black,Karen.Schuchardt,debbie.gracio,
Bruce.Palmer}@pnl.gov

Abstract. The Extensible Computational Chemistry Environment (Ecce) is a suite of distributed applications that are integrated as a comprehensive problem solving environment for computational chemistry. Ecce provides scientists with an easily used graphical user interface to the tasks of setting up complex molecular modeling calculations, distributed use of high performance computers, and scientific visualization and analysis. Ecce's flexible, standards-based architecture is an extensible framework that represents a significant milestone in production systems, both in the field of computational chemistry and problem solving environment research. Its base problem solving architecture components and concepts are applicable to problem solving environments beyond the computational chemistry domain.

1 Introduction

Addressing the complex environmental issues facing the nation today requires new scientific understanding of the fundamental chemical, physical, and biological processes underlying these issues. We must understand these phenomena at the basic molecular level. Modeling and simulation techniques have advanced to an accuracy that is quickly approaching that of experimental processes. An integration of experimentation with modeling and simulation will support scientists' understanding of the behavior associated with these systems to solve grand challenge problems.

The Extensible Computational Chemistry Environment (Ecce), developed at the William R. Wiley Environmental Molecular Sciences Laboratory (EMSL) at Pacific Northwest National Laboratory (PNNL), is the first comprehensive, integrated, problem solving environment developed to support computational chemists. Ecce is a seamlessly integrated suite of distributed client/server applications. Ecce enables scientists to easily use computational chemistry software to perform complex molecular modeling and analysis tasks by accessing distributed, high-performance computers from their desktop workstations. Ecce includes a sophisticated graphical user interface, scientific visualization tools, and the underlying data management framework, providing scientists with the tools to efficiently set up calculations, store

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 122–131, 2003.
© Springer-Verlag Berlin Heidelberg 2003

and retrieve results, and analyze the rapidly growing volumes of data produced by computational chemistry studies.

Ecce includes support for building molecular models; a graphical user interface to a broad range of electronic structure theory types; remote submission of calculations to UNIX and Linux workstations, clusters, and supercomputers; three-dimensional visualization and graphical display of molecular data properties; and extensive web-based help. Ecce applications can be divided into the following categories based on usage:

Organizational: The Ecce Gateway provides central access to the core tools within the Ecce application suite as well as access to online help and support options. The Calculation Manager allows researchers to organize and manipulate computational chemistry studies in Ecce. This application provides an at-a-glance overview of the status of every calculation, and easy access to key setup parameters and run statistics.

Setup: The Molecule Builder enables researchers to build, visualize, modify, and manipulate 3D images of chemical systems to be used in Ecce calculations. Chemical systems may be imported/exported in standard file formats such as XYZ and PDB. The Basis Set Tool supports over 230 predefined Gaussian basis sets for the user to select from or create new basis sets for use in *ab initio* electronic structure calculations. For a given chemical structure, a list of available basis sets is displayed for selection. The Calculation Editor provides the user with application-specific options needed to perform a calculation. As selections are made and parameters set, the editor automatically enforces valid inputs. The editor provides the final translation to the required input format for each computational code supported.

Job Launching: The Launcher submits the calculation to a compute resource after performing validation checks against user input and generating a job submission script. The Machine Browser displays reference information about compute resources known to Ecce and performs queries for overall machine, job, and disk status to aid selection of an appropriate resource to run a job.

Monitoring/Analysis: The Calculation Viewer provides access to current information for a single calculation during and after execution. The viewer supports computation of molecular orbitals, animation of normal modes, display of geometry optimization, Mulliken charges, and much more. The jobstore and jobmonitor applications are non-graphical interface processes that link the workstation running Ecce applications and the compute resource for monitoring job status and parsing output properties.

This paper provides an overview of the Ecce architecture and highlights many of the key technologies used to enable flexibility within an ever-expanding suite of software tools. Collaborations with outside projects and future directions are also summarized.

2 Ecce Architecture

The Ecce architecture seeks to achieve a number of often competing goals. These include creating a loosely coupled, component-based design; providing secure extensible access to any computational resource; leveraging existing networking services (rather than requiring the installation and administration of new ones); supporting the registration of additional computational codes with minimal

programming and in a way that takes advantage of the rest of the architecture; and designing for the flexibility to incorporate new technologies and techniques. In addition, since the initial development of Ecce there has been a movement toward non-proprietary and standards-based software bolstered by many freely available and open source projects. Ecce's architecture is evolving to assimilate these standards and projects. A major new goal of Ecce is to remove remaining dependencies on proprietary products so that Ecce can be extended by a much wider developer base.

Fig. 1 shows a high-level operational view of the Ecce architecture. A suite of desktop applications assist the researcher with calculation setup, job execution, and results analysis. These components, acting independently, each augment a web-enabled data store with data and metadata, communicating indirectly via a central messaging system. Standard network services are used to submit jobs to a wide variety of queuing systems as well as workstations. The chemistry codes currently supported are NWChem, Gaussian 98, and Amica. Ecce is written primarily in C++ but also makes appropriate use of Perl, Java, and Python, as will be discussed.

The following sections describe the core components of the Ecce architecture, illustrating why Ecce is a significant milestone in production systems, both in the field of computational chemistry and problem solving environment research.

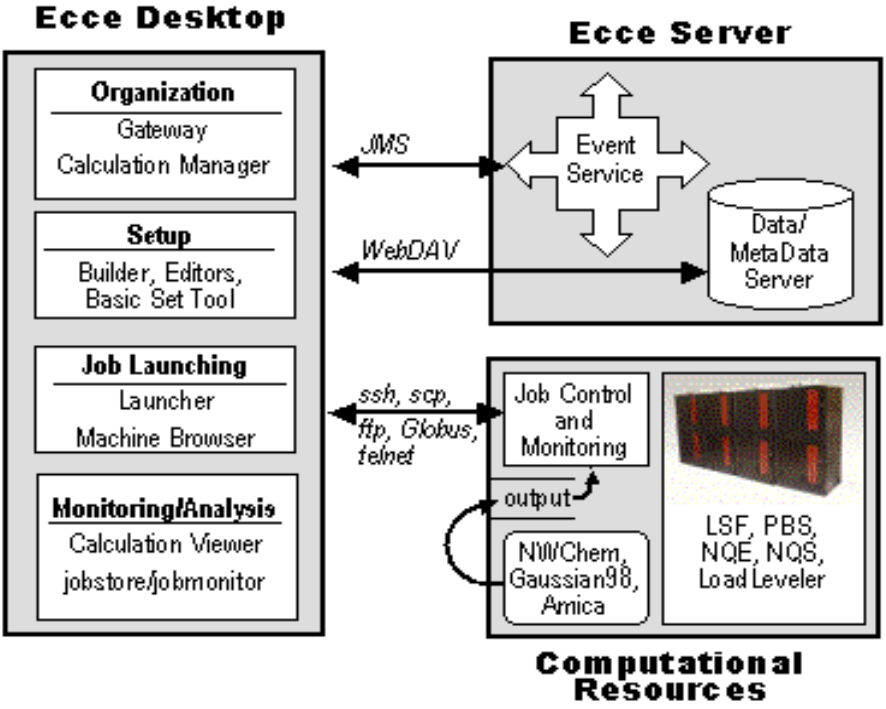


Fig. 1. Operational view of Ecce architecture

2.1 User Interface and Visualization

Two commercial tools for developing Motif applications are licensed by the Ecce project. They are TeleUSE from Aonix Corporation for laying out interfaces and generating source code to create them, and XRT PDS from Sitraka for advanced user interface widgets. Ecce has developed a base set of reusable interface objects (within TeleUSE these are referred to as templates) that can be used to more quickly create applications conforming to the standard Ecce look and feel. These templates are instantiated as C++ objects, hiding much of the underlying complexity of traditional Motif GUI development. The XRT PDS widget set provides objects such as spreadsheet-like tabular data entry and display, hierarchical folder/document tree views, and two-dimensional XY graphs.

Visualization is based on the Open Inventor C++ class library developed by SGI now available as an open source project. Open Inventor is implemented on top of OpenGL, the industry standard for high-performance visualization, and uses a scene graph paradigm to considerably reduce the complexity of visualization development. Molecular Inventor, an API that sits on top of Open Inventor, supports display and manipulation of atoms, bonds, and iso-surfaces.

2.2 Data Management

The data management component of Ecce is a ground-breaking achievement that addresses some of the inherent problems of managing scientific data with traditional database approaches such as relational database management systems or custom file-based implementations. Ecce uses the Web Distributed Authoring and Versioning protocol (WebDAV) [1], a set of extensions to the HTTP protocol that supports document retrieval, storage, and editing, as well as the annotation of the data with arbitrary metadata. Documents are used to store input setup information (the molecule, basis set, input files) and results, while metadata is used to annotate data to make it more meaningful to query engines or other non-Ecce applications. Documents can be in any format. Ecce uses a combination of formats native to the computational codes and XML for its flexibility in describing scientific data and the wide availability of tools. Because of the WebDAV-based architecture, Ecce data sets are directly available to users through their web browsers. Since WebDAV is a standard, many alternative implementations are available, allowing scalability decisions to be made at deployment time. As currently distributed, Ecce uses an Apache 2.0 server with the mod_dav module. A more complete discussion of the Ecce data management architecture can be found in [2].

2.3 Messaging

Ecce applications are able to work together while achieving the architectural goal of loose coupling, through the messaging or event layer. The event architecture allows individual applications to notify subscribers of its actions without direct knowledge of other applications. It also provides basic capability for collaborative research and supports the addition of agents without impacting existing functionality. Ecce has adopted the Java Messaging System (JMS) standard for its basic messaging engine. JMS is a subscription-based, publish-subscribe service where applications declare an interest in receiving notifications matching certain patterns, and the JMS server filters and routes these messages. JMS is a standard with multiple implementations. It runs

on any platform and non-Java bindings are available. Java Open Reliable Asynchronous Messaging (JORAM) was selected as our JMS server.

2.4 Computational Code Registration

Ecce provides a framework for registering new computational codes through structured script writing rather than reworking core C++ applications. Registering a code with Ecce consists of developing three components:

1. user interface detail dialogs with Python
2. application input file generation with Perl
3. output log file parsing with Perl to extract properties for analysis.

The design of code registration allows it to be done by a scientist with expertise in the computational code.

Ecce uses a set of Python classes that extend the freely available Qt user interface toolkit, PyQt. The Ecce Python classes define the base interface objects such as numeric input with range validation, option menus, and toggles used to create "detail dialogs." Detail dialogs are code-specific dialogs to set input options such as convergence criteria or algorithms to apply.

To accomplish input file generation, a Perl script (or other program) is written that is responsible for turning user inputs (molecule, basis set, options specified in detail dialogs) into a valid input file.

Output file parsing is the final aspect of code registration. Both server- and client-side parsing is done. On the computational server, chunks of raw data are extracted with keyword matches as the output log file is being generated, providing a filtering capability. This data is sent back to the client machine where the Ecce application software is running. A parser on the client side takes this raw property data and translates it into a standard XML file format defined by Ecce based on the dimensionality of the data. The client-side parsing is handled by writing separate Perl scripts, typically short, for each property. The Ecce Calculation Viewer provides default displays and visualizations based on data dimensionality.

2.5 Remote Communication

All communication with distributed compute resources is handled through an Ecce C++ class. This class is a high-level robust implementation for managing remote command shells, single commands, and file transfer on UNIX/Linux platforms. Specifically, the RCommand class is used for copying files to and from computational servers, submitting jobs, collecting job status and ongoing output data, and handling various other file and job management tasks. RCommand wrappers the industry standard remote shell and remote copy commands, presently including secure shell and secure copy (ssh/scp), telnet and ftp, rsh and rcp, and the Globus Toolkit 2.2 [3]. The RCommand class treats these remote shells as "black boxes"—it is not intrinsically linked or otherwise dependent on libraries, services, daemon processes, or socket-level protocols supporting remote access. Use of either secure shell and secure copy or the Globus Toolkit adds all the authentication and encryption security normally afforded these commands to any software using the RCommand class. RCommand is unique as a fast, lightweight, standards-based mechanism for building

remote access into UNIX software and allows Ecce to deploy to virtually any infrastructure without imposing new administrative tasks.

2.6 Security

The security model within Ecce is vital to maintaining the confidentiality of calculation data, and even more importantly, preventing access to remote systems that are configured to run jobs. Ecce uses a single-point login, chosen by the user and referred to as a passphrase. This passphrase is used as the decryption key for all other passwords that are stored as encrypted values using the Blowfish encryption algorithm [4], thus achieving the appearance of single signon from the user's perspective. The passphrase is maintained as an encrypted value unique to every Ecce session, stored only in the memory of Ecce applications. To maintain this level of security, it is strongly recommended that some form of secure remote communications shell be used—such as ssh, Globus, or Kerberized rsh—though Ecce maintains support for traditional protocols such as telnet and rsh.

2.7 Compute Resource Registration

Ecce supports remote submission of jobs to UNIX and Linux workstations, clusters, and supercomputers with built-in support for LoadLeveler, Maui Scheduler, NQE, NQS, PBS, and LSF, as well as workstations. Ecce also provides a mechanism to integrate new queue management systems through script files. Experience shows that each site uses queue management systems in unique ways to support local features and requirements. Through configuration files, Ecce allows complete customization of job script generation.

A GUI application, which can be used by an Ecce administrator for shared resources or by individuals for their own resources, is the primary means of making compute resources known to Ecce. The machine registration application collects reference information about the type of machine, number of processors, paths to computational codes, and supported remote communications shells. Queued machines additionally require resource limits for each queue such as time, memory, and processors.

2.8 Job Launching and Monitoring

Launching a job to a compute resource is a straightforward process, made so by its reliance on the Ecce architecture components for remote communication and compute resource registration. Using the Job Launcher application, a user selects from available machines, overrides default request values such as number of processors, and presses the "Launch" button. Ecce creates a remote connection to the resource, performs various checks, transfers files, invokes the job submit script, and initiates the applications that provide job monitoring functionality.

The term job monitoring refers both to monitoring the status of a job in the queue management system (idle, running, complete) or process table for workstations and to providing near real-time analysis of results while the code is running. Ecce extracts, parses, stores, and visualizes any output data immediately as it is generated through passive file monitoring techniques, allowing the user to monitor progress and take corrective action. The client-side monitoring process, jobstore, is a C++ application that is responsible for dynamically installing the compute server monitoring script,

jobmonitor, re-establishing connections when faults occur, and posting all output data and files to the data server.

3 Ecce Collaborations

The Ecce project participates in number of collaborations. The most extensive collaboration is with the NWChem project with current emphasis on adding support for molecular dynamics (MD) calculations. Efforts to date have focused on construction and validation of large proteins. The Ecce Molecule Builder has been optimized to support systems with hundreds of thousands of atoms. The Builder's MD Toolkit supports manipulation of residues and validates structures against known residue datasets, assigning atom types and partial charges. Tools for editing force field parameters and constructing the topology are nearing an initial release. Future plans call for an MD Calculation Editor for input file setup, job execution, and archiving trajectory files. Support for viewing trajectory files and creating movies exist today.

A recent successful collaboration with the University of Utah resulted in the integration of Amica, a code for very high accuracy electronic structure energy calculations, into the Ecce framework. The effort entailed developing custom detail dialogs, input file generation, job submit script generation, and the extraction of selected properties for viewing in the Calculation Viewer. In addition, capabilities added to the Basis Set Tool give researchers significant control over construction of custom basis sets using the extensive EMSL reference basis set library.

An ongoing collaboration exists with the Collaboratory for Multi-scale Chemical Sciences (CMCS) project. The objective of CMCS is to provide tools that enhance chemical science research among multidisciplinary teams of researchers. For example, combustion researchers require expertise and data from thermochemists, kineticists, and molecular modelers. CMCS, using data middleware provided by the Scientific Annotation Middleware (SAM) project, is developing an adaptive informatics infrastructure and a web portal environment that integrates chemistry applications, data resources, and collaboration tools to achieve this objective. The initial integration of Ecce with CMCS focuses on integration at the data level. Consider a scenario in which a researcher developing chemical mechanisms has trouble matching experimental data. A sensitivity analysis indicates that the result is sensitive to the thermochemistry of a particular chemical species. The researcher uses CMCS to seek out data with lower levels of uncertainty. If the data does not exist, CMCS connects the researcher with a molecular chemist who provides the desired results by using Ecce.

CMCS is defining standards for annotating data with pedigree, allowing researchers to search, categorize, and trace data across scientific disciplines and back to its original source. Fig. 2 illustrates a capability to export from Ecce to CMCS with CMCS pedigree information attached. In this example, a molecular chemist computed a heat of formation for the scenario described above. Calculating this one value required several sub-calculations. When the data is exported to CMCS, the complete sequence with all data is provided and full pedigree information is added to allow other researchers to examine the steps used to create the value. This prototype scenario was demonstrated at the IEEE/ACM Supercomputing 2002 conference.

Finally, since CMCS also uses the WebDAV protocol for its data layer, Ecce can be run directly against a CMCS server. Users have immediate web browser access to their data via translators and viewers integrated within the CMCS portal environment.

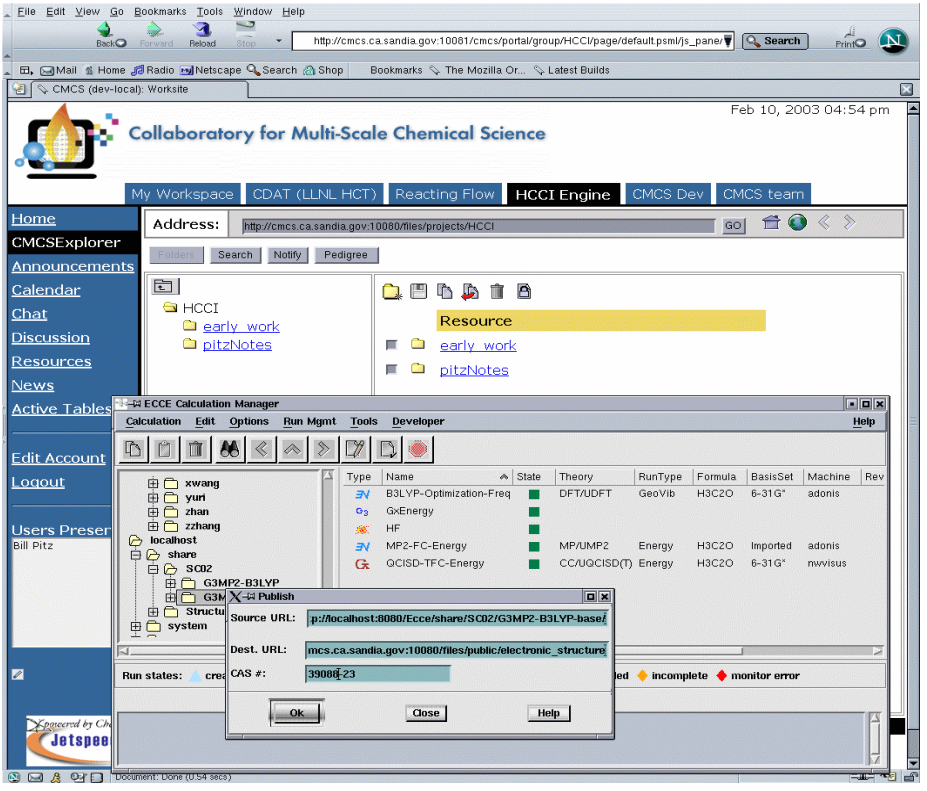


Fig. 2. Exporting results from Ecce to the CMCS Community Portal

4 Future Directions

Future development in Ecce will focus on both extending the chemistry domain functionality and improving and enhancing the underlying architecture. New domain functionality will continue to focus on molecular dynamics support and the development of a workflow-oriented interaction model that allows researchers to express and execute higher order operations that inherently require many computational steps. Examples include evaluations of the Basis Set Superposition Error (BSSE) and fragment guess calculations.

Extensions to the architecture include the replacement of the Motif development environment with a multiplatform, Python-extended toolkit such as Qt or GTK,

integration with an archive for larger data requirements of molecular dynamics, integration with new Open Grid Services Architecture (OGSA) [5], and continued adoption of CMCS and SAM technologies to support collaboration and web-based access to Ecce services.

Several Ecce applications are useful independent of the Ecce suite and will be distributed individually. These include the Builder, Basis Set Tool, and Calculation Viewer. The Builder will be available as a standalone distribution in spring 2003.

5 Summary

Ecce has matured from a test bed of prototype, standalone computational chemistry software in the mid 1990's to an integrated suite of tools heavily reliant on proprietary technology in the late 1990's to a robust, flexible, standards-based, globally available framework for ever-expanding computational chemistry capability development today. As Ecce follows this evolution, projects outside the domain of computational chemistry will be able to utilize the base problem-solving architecture components and concepts. With early collaborations already demonstrating results and with Ecce continuing to push new boundaries with support for user-controlled workflow, complete script-driven application development, Grid interoperability, and web integration, the promise for this pioneering problem solving environment continues to grow.

Ecce currently runs on Linux, Sun, and SGI workstations. The software suite is available to researchers through our website at http://ecce.emsl.pnl.gov. To download the software, an EMSL software user agreement must be signed by the requesting institution.

Acknowledgements. Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830. The William R. Wiley Environmental Molecular Sciences Laboratory, located at PNNL in Richland, Washington, is a national scientific user facility sponsored by the U.S. Department of Energy's Office of Biological and Environmental Research.

We would like to acknowledge the valuable discussions and interactions with the development team of the computational chemistry code NWChem.

We would like to acknowledge our ongoing collaboration with the Collaboratory for Multi-scale Chemical Science project and the Scientific Annotation Middleware project, both funded by the DOE Scientific Discovery through Advanced Computing Initiative. The CMCS website is located at http://cmcs.ca.sandia.gov. The SAM website is located at http://www.emsl.pnl.gov:2080/docs/collab/sam.

We would like to thank Dr. Robert Gdanitz formerly at University of Utah, now at North Carolina Agricultural and Technical State University, for his help in the integration of Amica as a computational code that Ecce supports. The Amica website is located at http://gdanitz.hec.utah.edu/amica.

References

1. Web Distributed Authoring and Versioning Protocol, http://www.ietf.org/rfc/rfc2518.txt
2. Schuchardt, K.L., Myers, J.D., Stephan, E.G.: A Web-Based Data Architecture for Problem-Solving Environments: Application of Distributed Authoring and Versioning to the Extensible Computational Chemistry Environment. Cluster Computing 5(3) (2002) 287–296
3. The Globus Project, http://www.globus.org/
4. Schneier, B.: Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993). Springer-Verlag (1994) 191–204
5. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C.: Grid Service Specification. Open Grid Service Infrastructure Working Group, Global Grid Forum, Draft 2, 7/17/2002

Computational Quantum Chemistry Experiments via the Web

Brian F. Yates[1,*] and Brian J. Duke[2]

[1] School of Chemistry, University of Tasmania, Private Bag 75,
Hobart TAS 7001, Australia
Brian.Yates@utas.edu.au
[2] School of Biological, Environmental and Chemical Sciences,
Northern Territory University, Darwin, NT 0909, Australia
b_duke@octa4.net.au

Abstract. The use of web technology for the provision of efficient and user-friendly interfaces to a variety of computational chemistry programs used in undergraduate experiments is described. Together with pedagogical descriptions of the theory, these interfaces are used to enhance the accessibility of experiments in computational quantum chemistry for senior students at the undergraduate level. The advantages and disadvantages of this approach are discussed and recommendations for future incorporation into the curriculum are made.

1 Introduction

For some time we have been involved in teaching computational quantum chemistry courses to students at our respective institutions. For many students this has been their first experience of computers beyond simple word processing and, prior to the work described here, they had to contend with a bewildering variety of personal computers, operating systems, communication programs, and analysis tools. In 1995 we were successful in obtaining funding from the Committee for Australian Undergraduate Teaching for a project entitled "Learning Modules for Computational Chemistry Over AARNET". This enabled us to develop the skeleton of a set of web-based modules for teaching quantum chemistry. Other modules in molecular modelling were developed by team members from Monash University and Swinburne University of Technology.[1] In 1997 one of us was successful in obtaining a Teaching Development Grant from the University of Tasmania which enabled us to develop the quantum chemistry modules further to the point where they could be used as part of our fourth-year undergraduate course.[2] Although direct funding for this work has now stopped, the team members continue to collaborate and we have continued to improve and develop the material under the banner of the Australian Computational Chemistry Via the Internet Project (ACCVIP).[3]

The quantum chemistry modules consist of some 200 web pages of material that comprehensively covers a large amount of quantum chemistry, several on-line com

* To whom correspondence should be addressed.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 132–141, 2003.
© Springer-Verlag Berlin Heidelberg 2003

prehension tests which give students immediate feedback and guidance when the answer is incorrect, a large number of on-line tutorial problems and exercises, and web-based interfaces to several computational chemistry programs. Thus the modules consist of a large amount of pedagogical material plus the interfaces.

The advantage of having so much of the pedagogical information on the web is that it provides more flexible access to the material. It is a resource which is available to the students 24 hours a day and it can be used in a distance education environment. However in our experience there is still a need for the students to print out the material and for the instructor to lead them through it in a classroom situation in order to gain the best understanding. Clearly different things work for different people. Part of the skill of being a good teacher is to assess where people are up to and present the material accordingly. For all of its use in flexible delivery, a computer cannot respond in a genuinely personal way to an individual's situation. Thus the real strength of this project and the one that has been of most benefit to the students is the development of the web-based program interfaces.

We have used material from the quantum chemistry modules in teaching a variety of courses, both remotely to a small number of MSc students based overseas and locally to students at our institutions in third and fourth year. This article describes some of our recent work in adapting the material to provide a set of computational quantum chemistry experiments via the web.

2 Web Interfaces

We have written the web interfaces as simple html documents which obtain information from the user via the use of forms and then process the information using various CGI scripts. Students are not required to log in to a specific computer or to learn any Unix commands. It is of course possible to restrict access to the web pages (either by use of passwords or by IP address) and this is used in some situations. We have developed web-based interfaces for a number of programs including the James-Coolidge, Hückel, Extended Hückel, MOPAC,[4] Gaussian,[5] GAMESS(US)[6] and DeFT[7] computer programs. Our CGI scripts incorporate a variety of features which are listed below:

a) The ability to run the job interactively or via a queue. We use the interactive method for the Hückel and Extended Hückel programs and the queuing method for the quantum chemistry programs. One of the advantages of the use of a queue is that because we have designed the experiments to involve fairly small calculations we can set a fairly short time limit on the length of each job. This prevents a student's calculation from going on endlessly when a mistake has been made.

b) The ability to check the status of the queue. This is a simple web interface to the utility in whatever queuing system is being used that displays all the entries in the queue.

c) The ability to execute jobs remotely. The web server machine does not need to be the same computer that the actual calculations are run on. For example, this allows an instructor at the Northern Territory University in Darwin to design and implement their own web interface as an html file served from an NTU computer, but for the actual calculations to be carried out on a computer at the University of Tasmania in Hobart. We have tested this in the case of the MOPAC program. Subject to licensing

agreements and permissions from the appropriate software companies, this allows computing resources to be shared for educational purposes. It also gives the instructor flexibility while reducing duplication of system administration.

d) The ability to produce a Chime[8] or Rasmol[9] picture of the input structure. This is achieved by calling Babel[10] from within the CGI script to convert the input data for the quantum chemistry programs into pdb format. This is then displayed for the user.

e) The ability to view the output as it is running. This is an advantage for short calculations in that it gives the student immediate feedback and allows them to see the results of the calculations as they are produced. For interactive jobs (for the Hückel and Extended Hückel programs) this is the only way the output is available.

f) The ability to email the output back to the user. Students are asked to specify their email address in the web interface and then the output from the calculation is emailed back to them. The success of this feature relies on the student typing their address correctly, so to help them the instructor can set the default for that part of their address relating to the particular institution (for example at the University of Tasmania it is set to "@postoffice.utas.edu.au").

g) The ability to log all usage. When difficulties occur due to jobs not running correctly or when students request extensions for work it is useful to be able to look at the logs to determine what went wrong. Our CGI scripts record the date, the calling page, the email address of the user, the address of the remote host, the input data file, and any error result upon submission of the job.

We have used three different styles of web-based interfaces for the computational chemistry programs:

a) Forms with drop down menus. These interfaces have spaces where the user is required to fill in certain information (for example, the title and Cartesian coordinates), and other areas where options for the calculation can be easily selected through the use of drop down menus (for example, the level of theory, the basis set, the type of calculation (single point, optimisation or frequency), and the charge and multiplicity). This style of interface is well suited for the beginning student where most of the information can be hidden away or presented as the default menu item so as to simplify the input. It may be that all the student has to do is fill in their email address and press 'submit'. This style of interface is also well suited to fairly straight forward experiments where only a few theoretical methods are to be used. This has the advantage that only the options specified by the instructor for that experiment need be presented in the menu. Finally this style of interface is also well suited for enhancing with extra commands the basic input data provided by the student. For example, the choice of 'single point' in the type-of-calculation menu may be translated to mean 'scf=tight' in the Gaussian program input data.

In practice we have developed several different general interfaces of this type to the quantum chemistry programs, each with an increasing number of options.

b) Plain form. With this style of interface the user is presented with a single box on the page into which they may enter a complete data file, including pre- and post-processing commands (for example, the '%' commands in Gaussian).

c) Upload a file. With this style of interface the user selects an input file on their local computer and it is then uploaded to the web server and submitted to the queue.

Some examples of the interfaces we use in our teaching are shown at http://www.chem.utas.edu.au/staff/yatesb/honours/modules/faces.html. An example of the html source code for a simple form with drop down menus is shown below:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<p>Select the appropriate menu or fill in the appropri-
ate box. The output will be emailed to you.
<FORM METHOD="post" ACTION=
"http://www.chem.utas.edu.au/cgi-bin/master.cgi">
Your e-mail address:<BR>
<INPUT TYPE="text" NAME="emailaccount" SIZE=12
Value=""> @ 
<INPUT TYPE="text" NAME="emailserver" SIZE=40
VALUE="postoffice.utas.edu.au"><P>
<INPUT TYPE="hidden" NAME="qu_time" VALUE="">
<INPUT TYPE="hidden" NAME="callpg" VALUE=
"http://www.chem.utas.edu.au/teaching/example.html">
Title: <INPUT TYPE="text" NAME="title" SIZE=40
VALUE=""><P>
Select the method in the menu below.<BR>
Method:
<SELECT NAME="method">
          <OPTION>RHF
          <OPTION>B3LYP
</SELECT><P>
Select the basis set in the menu below.<BR>
Basis set:
<SELECT NAME="basis">
          <OPTION>STO-3G
          <OPTION>3-21G
          <OPTION>6-31G
          <OPTION>6-31G(d,p)
</SELECT><P>
Charge      
<INPUT TYPE="text" NAME="charge" SIZE=1 VALUE="0">
     
Multiplicity      
<INPUT TYPE="text" NAME="multiplicity" SIZE=1
VALUE="1">
<P>Select geometry optimisation or single point:
<SELECT NAME="cmd">
          <OPTION>Optimisation
          <OPTION>Single Point
</SELECT><P>
```

```
Z-matrix: Make sure you hit "Enter" after the last
line, and only hit it once.<BR>
<TEXTAREA NAME="matrix" ROWS=4 COLS=40></TEXTAREA><P>
<INPUT TYPE="submit" VALUE="Run Gaussian data">
<INPUT TYPE="reset" VALUE="Reset"><P>
</FORM>
</BODY>
</HTML>
```

In this case we have restricted the user to the choice of two levels of theory, four basis sets, and two types of calculation. This produces the interface shown in Fig. 1.

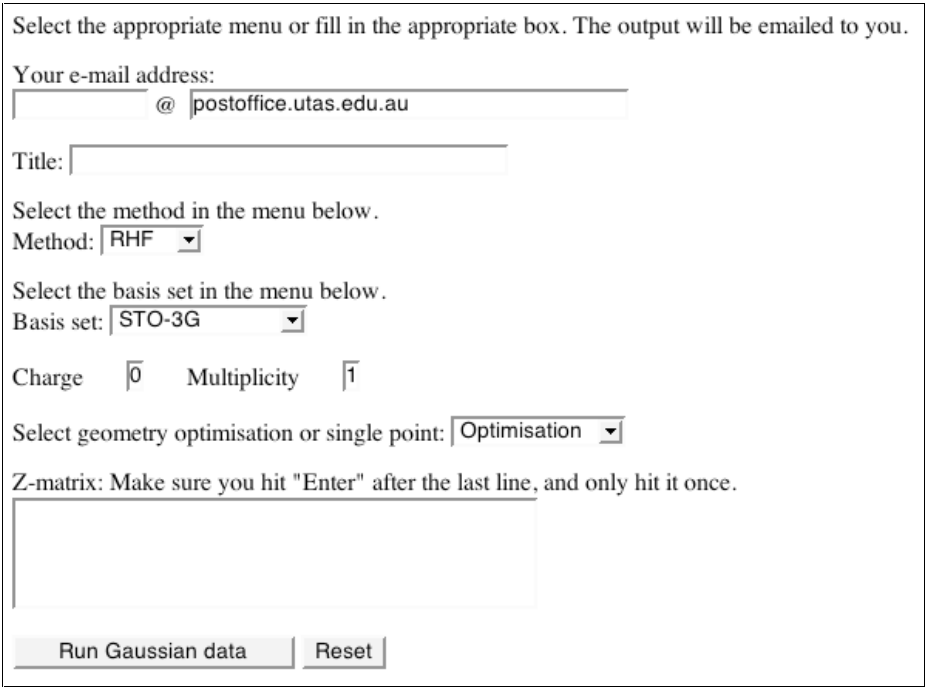


Fig. 1. Example interface to the Gaussian program

While the html code is fairly straight forward, the crown jewels of this project are of course the CGI scripts that make it all work.

3 Experiments

The interfaces described above have been used extensively in a series of exercises and problems incorporated into our modules on semi-empirical theory and basic *ab initio*

theory which were developed as part of the ACCVIP collaboration. More recently we have written a series of third-year undergraduate experiments in computational quantum chemistry which have been used at the Northern Territory University (NTU) over a number of years and will be used for the first time at the University of Tasmania in late 2003.

These experiments include one entitled "Calculating simple force constants with computational quantum chemistry", developed as part of the Australian Physical Chemistry Enhanced Laboratory Learning (APCELL) initiative.[11] The experiment was first used in the current form with second-year undergraduate students at NTU in 2002. It grew out of a similar experiment on the bending force constant of CO_2 which had been used for many years at NTU. That experiment in turn owed its origins to a description by Mathews.[12] Full details of the current web-based experiment will be published elsewhere.[13]

This experiment investigates the calculation of the force constants of some simple diatomic molecules: N_2, O_2 and F_2. For each molecule, students are asked to calculate the minimum energy and the single point energies at slightly distorted geometries. Then by assuming a quadratic parabola the force constant is determined. Students are asked to investigate the results using different levels of semi-empirical, *ab initio*, and density functional theory, and compare their findings to their expectations and to experiment.

In this experiment we have used web interfaces with drop down menus to limit the number of choices that the students have. For example, in the semi-empirical calculations with MOPAC the methods are restricted to just the choice of AM1 or PM3. We have also used these web pages to provide a combination of the explanation and the interface. This allows us to provide the information (about the program or about the experiment) at the time it is needed to perform the calculation. The MOPAC interface used in this experiment is shown in Figures 2 and 3. This shows how the MOPAC keywords are described where necessary, and how extra information is provided about the O_2 triplet molecule. This interface is available at http://www.chem.utas.edu.au/teaching/KRA301/exp8/mopac.html.

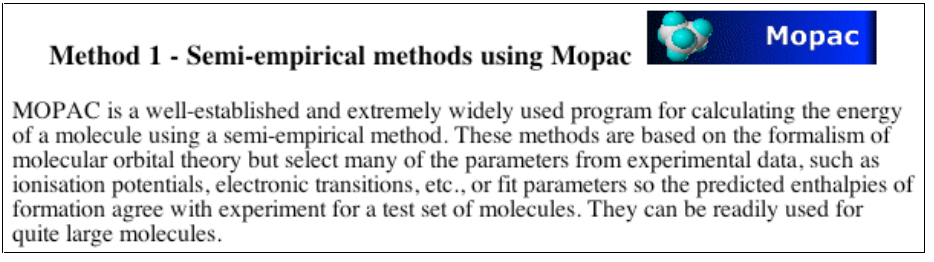


Fig. 2. Interface to MOPAC for the force constant experiment

There are several different methods in the MOPAC code. We will use just two - AM1 and PM3. The method is selected from a drop down menu. First, determine the optimised bond distance for your diatomic molecule. Then calculate the energy difference when the bond length is increased by 1.0 pm and then decreased by 1.0 pm. Note that you actually have to give the bond length in Ångstrom units. Select just one of the two methods, but if you are in a group, organise different members of the group to select different methods and later pool your results.

Since this method uses significant resources on the server computer, although only for molecules much larger than the ones you are using, the calculations are queued and the results are returned to you by e-mail. Make sure you get your e-mail address correct below.

Here is a simple interface to **MOPAC**. All you need to do to run this calculation yourself is fill in your email address (change the default that the page gives you), select one of the methods, PM3 or AM1 from the menu, modify the title window, type in a Z-matrix, check you have the keywords correct and then press "Run MOPAC data".

Your e-mail address:

[ ] @ [postoffice.utas.edu.au]

Title:

[ ]

Method: [PM3]

Other keywords:

[precise]

The default action in MOPAC is to optimise the geometry - the 'precise' keyword tightens up the convergence criteria to ensure that sufficient accuracy is obtained in the final geometry. Do this calculation first. Subsequently, to do what is called a single point calculation, i.e. just at the geometry you have given it, for the shortened and lengthened bonds, you have to replace 'precise' with the **'1scf'** keyword.

No other keywords need be added for F_2 and N_2. However, since O_2 is a triplet ground state, you have to add, after either 'precise' or '1scf', **'uhf triplet'**.

Z-matrix or cartesian co-ordinates:

[ ]

[Run MOPAC data] [Clear form]

Extract the energy from either the enthalpy of formation value in kJ (actually kJ/mol) or the total energy in eV [1 electron volt (eV) = 96.485 kJ/mol]. Why do the enthalpy and energy values in kJ/mol give the same value for the energy difference caused by expanding or contracting the bond?

Fig. 3. Continuation of interface to MOPAC for the force constant experiment

4 Discussion

The major advantage of the web-based approach described here is that it provides a consistent cost-effective interface to a variety of computational chemistry programs. Rather than confronting students with a disconcerting array of login procedures, Unix commands, and job submission processes and perhaps restricting them to certain computers in the undergraduate lab where the software is available, this approach enables the student to use just one program, a web browser, and to perform the calculations from wherever internet access is available (their lab, dorm or internet café). It does not matter which type of personal computer or operating system the students are using; each student will see the same thing. This uniformity encourages a more ready acceptance by the students. In addition the material is available 24 hours a day and the students are not restricted to just certain times when the computer lab is open. These reasons all make it easier for the student to focus on the underlying computational chemistry which is being taught. It also makes it much easier for the instructor since they do not need to worry about setting up communication programs (telnet, x-windows, etc.) and login accounts, and teaching the students about the operating system. Instead they can focus on teaching students about computational chemistry.

There are some disadvantages however. In our implementation we have deliberately removed nearly all reference to and use of files. In the basic web interface the student does not deal with any input files, output files, scratch files, or checkpoint files. Our experiments are designed to use small molecules (so there is no need to type a lot of information) and there is no need to use a checkpoint file. This simplifies things for the student, but it would certainly constrain the serious user. We do print the input data at the top of the output so that the student begins to get the idea of what a complete input file looks like. Our more advanced experiments also make use of the plain form and the file upload interfaces in which a complete set of input data is required. These move the student towards an understanding of how to run real research jobs. We have also developed some advanced interfaces which have ways to keep and handle the checkpoint file in Gaussian, although this has not yet been trialled with students.

Another possible disadvantage is that although this method of presentation provides a good mode of instruction for undergraduate students, are we really teaching them how to use the computational quantum chemistry programs in real life? After completing the experiments could they say they know how to use the MOPAC program in a research situation for example? The answer is probably ‘no’.

It is worth comparing our approach to that of others. In recent years a number of other workers have developed web interfaces to computational quantum chemistry programs (see for example references [14]-[17]). The most sophisticated of these is WebMO developed in the group of Professor William Polik.[14,18] This not only provides some of the capabilities described in this article but also includes a tool for visualising the input data and output results on the web via a Java applet. This is a very nice package and is highly recommended. We have continued to use our system because of the flexibility it offers us as instructors, and because we are able to readily integrate the delivery of information within the interfaces. As described above, this

allows us to provide the instructions for an experiment on the same page as the interface and to very easily tailor the interface for the particular set of options required for that experiment.

The ubiquitous use of the web means that without doubt it is the most used and most accessible interface in the world today. Nearly all the traditional computer information systems in a university (library catalogues, financial management, student records, timetables, booking systems, email, lecture notes, and so on) are now available via the web. Students have embraced this and become used to the convenience that it offers. It is only natural that they expect the same ease of use from our computational chemistry programs and the interfaces described here go some way to meet their needs. We expect this demand from students to increase and our computational quantum chemistry experiments will be adapted and expanded to meet this demand.

Acknowledgements. It is a pleasure to acknowledge our collaborators who contributed so much to this project: Dr Graeme Cross, Dr Margaret Wong, Dr Ted Lloyd, Dr Kai Ling Ng, Dr Yos Ginting, Trent Wale and Vinutha Ramakrishna.

References

1. Salter-Duke, B.J.; Wong, M.G.; Lloyd, E.J.; Yates, B.F.; Cross, G.J.: Learning Computational Chemistry via the Internet. CAL-laborate (October 1998) 31–33
2. Yates, B.F.: KRA400 Chemistry Honours – Topics in computational chemistry. http://www.chem.utas.edu.au/staff/yatesb/honours, accessed on 21 February 2003
3. Australian Computational Chemistry Via the Internet Project (ACCVIP). http://www.chem.swin.edu.au/CAUT.html, accessed on 21 February 2003
4. Stewart, J.J.P.: MOPAC93. Fujitsu Limited, Tokyo, Japan (1993)
5. Frisch, M.J., Trucks, G.W., Schlegel, H.B., Gill, P.M.W., Johnson, B.G., Robb, M.A., Cheeseman, J.R., Keith, T., Petersson, G.A., Montgomery, J.A., Raghavachari, K., Al-Laham, M.A., Zakrzewski, V.G., Ortiz, J.V., Foresman, J.B., Peng, C.Y., Ayala, P.Y., Chen, W., Wong, M.W., Andres, J.L., Replogle, E.S., Gomperts, R., Martin, R.L., Fox, D.J., Binkley, J.S., Defrees, D.J., Baker, J., Stewart, J.P., Head-Gordon, M., Gonzalez, C., Pople, J.A.: Gaussian 94, Revision B.3. Gaussian, Inc., Pittsburgh PA (1995)
6. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M., Montgomery, J.A.: GAMESS(US). J. Comput. Chem. **14** (1993) 1347–1363
7. St-Amant, A.: DeFT. University of Ottawa, Ottawa (1998)
8. Chime. http://www.mdli.com/chime/, accessed on 21 February 2003
9. Rasmol. http://www.umass.edu/microbio/rasmol/, accessed on 21 February 2003
10. Walters, P., Stahl, M.: Babel. (1992–1996)
11. Kable, S., Buntine, M., Jamie, I.: APCELL. http://www.apcell.org, accessed on 21 February 2003
12. Mathews, G.P.: Experimental Physical Chemistry. Clarendon Press, Oxford (1985) section 5.13, pages 280–287
13. Duke, B.J., Ramakrishna, V.: Calculating Simple Force Constants With Computational Quantum Chemistry. Aust. J. Educ. Chem. (2003) to be submitted
14. Polik, W.: WebMO. http://www.webmo.net, accessed on 21 February 2003

15. Rösch, N.: Web interface to Gaussian. http://theo1.theochem.tu-muenchen.de/qcl /help/interface_e.html, accessed on 21 February 2003
16. Portal interface to Gaussian. http://www.osc.edu/research/bioinformatics/portal /gaussian.shtml, accessed on 21 February 2003
17. WebSubmit. http://math.nist.gov/mcsd/savg/websubmit/, accessed on 21 February 2003
18. A review of WebMO. http://www.gaussian.com/nl001_up.htm, accessed on 21 February 2003

Tonto: A Fortran Based Object-Oriented System for Quantum Chemistry and Crystallography

Dylan Jayatilaka and Daniel J. Grimwood

Chemistry, School of Biomedical and Chemical Sciences,
University of Western Australia, Crawley 6009, Australia

Abstract. Tonto is an object oriented system for computational chemistry. This paper focuses mainly on the `Foo`, the object oriented language used to implement Tonto. `Foo` currently translates into Fortran 95. It offers almost all the features of the coming Fortran 2000 except for dynamic types. It goes beyond the Fortran standard in that parameterised types and template-like inheritance mechanisms are provided. Since the method is based on textual inclusion, it generates code which is easy for the compiler and human to understand. Example code is given, and possible future work on the language is discussed.

1 Introduction

Careful studies have demonstrated that over the useful lifetime of a software product, testing, documentation, minor extension, and "bug-fixing" account for the majority of the programming effort [1]. At the same time, the ability to join together different software technologies in computational chemistry is currently extremely poor: at our disposal are monolithic program packages, for example Mopac [2] and Gaussian [3]. These packages are often poorly documented at the code level; many were developed over decades in an ad-hoc manner. Consequently, these packages require experts in order to extend and modify them. Further, the extensions may not be available to all researchers for further development because the source code may be subject to restricted distribution policies.

Over the last five years we have developed a system called Tonto, which attempts to address these problems. Tonto is mainly intended for computational chemists with new ideas, but who are discouraged by the time it might take to understand and modify old, monolithic programs in order to implement those ideas. The three main goals of Tonto are, in order of importance:

1. To provide useful open-source tools for computational chemistry
2. To be simple to use, understand, maintain, and modify
3. To provide code that executes quickly.

Fortran was chosen as the language to implement the Tonto system because of large existing base of Fortran software in computational chemistry, and also because there is a tradition of producing highly optimising Fortran compilers for

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 142–151, 2003.
© Springer-Verlag Berlin Heidelberg 2003

numerical applications. At the same time, however, it was difficult to ignore the general consensus in the software engineering community that object oriented programming techniques form the best basis for developing well structured and modular code—the kind of structured code that can be easily modified and joined, as required to solve modeling problems in chemistry. Although Fortran 95 is an object oriented language, there are significant deficiencies. That is why it was decided to develop a new dialect of Fortran 95 which more fully supports object orientation. We call the resulting language `Foo`, which stands for object oriented Fortran in reverse (since true polymorphic object orientation is not really possible in Fortran just yet). The `Foo` language forms the central part of the Tonto system.

We are not able to discuss all aspects of the Tonto system in this article. Instead, we focus on a description of the `Foo` language and its features, especially its object oriented features. Some of the object-oriented aspects of `Foo` have been separately implemented in Fortran 90 [4,5,6,7,8,9,10,11,12,13,14], for example, the use of the `self` type [11], template inheritance using textual replacement [13, 14], and the passing of data and functions together [5]. We give some examples of the code written in `Foo`, together with its translation into Fortran 95, and output. Our conclusions and outlook for the system are given at the end.

2 The `Foo` Preprocessor

The `Foo` language is currently implemented as a `perl` script which translates into standard Fortran 95. Since the `Foo` language is almost identical to Fortran at the expression level, it can be viewed as a dialect of Fortran.

The use of preprocessors to implement object oriented languages is not new. For example, the first implementation of C++ was as a preprocessor translating into C [15]. For object oriented styles, preprocessing is particularly useful for incorporating parameterised types into languages that only support the static specification of type parameters [16].

The syntax of the `Foo` language is similar to that of Python, Eiffel and Sather [1]. The most important syntactic element of `Foo` is the use of "dot notation". In this notation, a dot "`.`" expands into the Fortran 95 "`%`" type separator symbol if the symbol after the dot refers to a component of the derived type of the object, otherwise into a procedure call. By using this notation, it is possible for the programmer to "hide" the detailed implementation of a type (and hence a module) from a casual user in the sense that a user need not worry about whether a particular element is a procedure or a derived type component.

There are also clear practical advantages to using the `Foo` language. A simple character count on the existing Tonto library shows that coding in `Foo` eliminates 40% of characters compared to the corresponding Fortran code. We believe that this is a real figure, rather than a reflection of a poor translation into Fortran. It is easy to verify that the emitted Fortran code is clean, efficient and understandable; a programmer could equally work with the Fortran code emitted from the `Foo` preprocessor, as with the `Foo` code itself.

The following is a brief description and discussion of the features of the `Foo` preprocessor.

- **Dot notation**. Dot notation, as used in most object oriented programming languages, mostly eliminates the need for `call` statements.
- **`HTML` and other code documentation facilities**. A translator automatically generates `HTML` documentation from the source code, which is fully cross-referenced. Syntax highlighting and routine cross-referencing are also provided for the `vim` editor.
- **Automatic inclusion of `use`'d modules**. The `Foo` language automatically generates a list of the modules required by a particular program or module, and the routines needed from those modules. This greatly alleviates the maintenance burden from the programmer, and can reduce the compilation resources for complex modules.
- **Automatic `self` variable inclusion, and rudimentary closures**. The `Foo` preprocessor automatically inserts variable `self` as the first dummy argument for every procedure in a module. The type of `self` is of the same kind as that described by the module. This automatic mechanism can be suppressed if desired. Such "`selfless`" procedures are typically passed to other procedures (they may be, for instance, procedures used for numerical integration). Even though the `self` variable does not appear explicitly as a procedure argument in a `selfless` procedure, it may be stored temporarily as a module variable (typically called `saved_self`) at the point of the procedure call, and then passed to a local `self` variable as the first executable statement within the `selfless` routine. By this mechanism, the state of the function at the point of call can be stored i.e. this becomes a way to define functions with partially evaluated arguments (such notion is called a "closure", or "lambda expression" in the Lisp or Scheme languages).
- **Overloading and interface files**. Procedures with exactly the same name can be written within a particular module, provided that they can be distinguished by their call signature. The interface file required for this overloading is automatically generated.
- **Multiple template inheritance**. `Foo` allows a textual inheritance mechanism for modules based on derived types which are built as compositions of other, simpler derived types; and also for modules which are related to one another by a dependence on a type parameter (for example, matrices of real and complex numbers). This mechanism is not restricted to inheriting code from a single module.
- **Automated build procedure**. The build procedure for automatically takes account of the dependencies generated through the inheritance mechanism, and solves the problem of cascading module compilation [17], greatly reducing compilation times.
- **Preconditions, postconditions, and error management facilities**. Preconditions and postconditions can be placed at the beginning and end of procedures to ensure that the minimum requirements for a routine to perform its task correctly are met. This "design-by-contract" style greatly helps

in reducing run-time errors. If an error is detected during program execution, a traceback of the call stack may elucidate where it happened. These features can be disabled or enabled when the program is compiled.
- **Memory management facilities**. Fortran 95 offers dynamic memory allocation method and pointers, but this can lead to memory leaks. `Foo` can automatically check if a procedure is memory conserving, unless specifically marked `leaky`. A running total of allocated memory can be printed out during program execution to trace routines where unintentional memory leaks and sinks occur. This facility can be enabled or disabled when the program is compiled.
- **Well defined input syntax**. The Tonto system defines a (BNF) syntax for input file format, which can be used both for control of program execution and for data interchange. Code tools are provided for this through the `Foo` inheritance mechanism. An automatic mechanism is provided whereby unrecognised input file syntax generates an error message listing the keywords allowed at that point. This greatly facilitates debugging input files when used with the manual.
- **Conditional compilation of `pure` and `elemental` routines**. Fortran 95 disallows write statements within procedures which are `pure` or `elemental`, a great annoyance for testing purposes. `Foo` code can be conditionally compiled with or without these keywords.
- **Automatic `end` keyword completion**. The `end` keyword can be used to stand for any of the usual block termination words: `end program`, `end module`, `end subroutine`, `end function`, `end if`, `end do`, `end interface`, `end select`, `end type`, `end forall`. This encourages the use of indentation. Programmers may choose not to use this feature, and can insert the keywords for themselves.
- **Automatic `subroutine` and `function` detection**. The `Foo` language automatically inserts `subroutine` and `function` keywords, to eliminate typing.
- **Protection against changes in the language elements**. The `Foo` preprocessor represents a minimal approach to object oriented programming, and it is conceivable that it may be translated into a language other than Fortran 95 at a future date. There is also the possibility of translation into more than one language. For example, automatic translation into the Python language may be considered, for "high level" objects.

3 Examples of the Tonto System

3.1 A Short Program Using Some Tonto Modules

Figure 1 presents a sample program written in `Foo`, which makes use of the `TEXTFILE`, `REALMAT`, and `GAUSSIAN2` modules. These modules are concerned with, respectively, text file manipulation, real matrix operations, and pairs of gaussian functions (especially quantum mechanical integrals). The corresponding Fortran 95 source emitted by the `Foo` preprocessor is shown in Figure 2. The output for the program is shown in Figure 3.

```
program run_example

   implicit none

   g :: GAUSSIAN2       ! A GAUSSIAN2 object
   S :: REALMAT*        ! A pointer to a real matrix

   tonto.initialize     ! Tonto system accounting (memory, call stack)
   stdout.create_stdout ! Create the "stdout" file
   stdout.open          ! Open it.
   tonto.set_error_output_file(stdout)

   stdout.flush
   stdout.text("First, define a gaussian pair ...")
   g.set(l_a=0,pos_a=[0.0d0,0.0d0,0.0d0],ex_a=0.3d0, &
         l_b=1,pos_b=[0.0d0,0.0d0,0.1d0],ex_b=0.5d0)
   g.put                    ! Output to "stdout"
   stdout.flush             ! Observe dot notation access to "g"
   stdout.show("The position of the first gaussian is ",g.a.pos)
   stdout.show("The exponent of the first gaussian is ",g.a.ex)

   stdout.flush
   stdout.text("Change the precision of the output to 3 decimal")
   stdout.text("places, and a field width of 8 characters")
   stdout.set_real_precision(3)
   stdout.set_real_width(8)

   stdout.flush
   stdout.text("Analytically evaluate the nuclear attraction")
   stdout.text("integrals for gaussian pair assuming the nucleus")
   stdout.text("of unit charge is at (0,1,0) :- ")
   stdout.flush
   g.make_nuclear_attraction_ints(S,c=[0.0d0,1.0d0,0.0d0])
   stdout.put(S)

   tonto.report
end
```

Fig. 1. An example program written in `Foo` source code and making use of a number of Tonto modules

The first observation is that the `Foo` source code is much shorter than the corresponding Fortran 95 code. The main reason for this is the insertion of the `use` statements. There is also a minor code reduction coming from the fact that `call` statements are not used. Although this is only a minor issue, the `call` syntax of Fortran 95 often causes great annoyance for those accustomed to more modern languages. Unfortunately, such stylistic trivialities greatly affect the popularity of a language.

The declaration of variables is also elegant: the rather verbose Fortran 95 `type` keyword is removed and replaced by a convention of using uppercase macros for the type. The use of macros also hides the distinction between intrinsic and derived types. This is important for in the case that `Foo` would be translated into a language other than Fortran, where such distinctions do not exist. Pointer attributes are represented by a star character after the type, a feature that should be familiar to C programmers. In `Foo`, the declaration order for variables and their attributes is reversed compared to Fortran 95. This style follows the philosophy of an assignment operation — except we are assigning a type rather

```
program run_example

   use TYPES_MODULE
   use SYSTEM_MODULE
   use TEXTFILE_MODULE, only: stdin
   use TEXTFILE_MODULE, only: stdout
   use TEXTFILE_MODULE, only: set_real_width_
   use TEXTFILE_MODULE, only: put_
   use TEXTFILE_MODULE, only: text_
   use TEXTFILE_MODULE, only: open_
   use TEXTFILE_MODULE, only: create_stdout_
   use TEXTFILE_MODULE, only: flush_
   use TEXTFILE_MODULE, only: set_real_precision_
   use TEXTFILE_MODULE, only: show_
   use GAUSSIAN2_MODULE, only: put_
   use GAUSSIAN2_MODULE, only: make_nuclear_attraction_ints_
   use GAUSSIAN2_MODULE, only: set_
   use GAUSSIAN_MODULE, only: n_comp_
   use REALMAT_MODULE, only: create_

   implicit none

   type(gaussian2_type) :: g ! A type(gaussian2_type) object
   real(8), dimension(:,:), pointer :: S ! A pointer to a real matrix

   call initialize_(tonto)      ! Tonto system accounting (memory, call stack)
   call create_stdout_(stdout)  ! Create the "stdout" file
   call open_(stdout)           ! Open it.
   call set_error_output_file_(tonto,stdout)

   call flush_(stdout)
   call text_(stdout,"First, define a gaussian pair ...")
   call set_(g,l_a=0,pos_a=(/0.0d0,0.0d0,0.0d0/),ex_a=0.3d0, &
         l_b=1,pos_b=(/0.0d0,0.0d0,0.1d0/),ex_b=0.5d0)
   call put_(g)                      ! Output to "stdout"
   call flush_(stdout)               ! Observe dot notation access to "g"
   call show_(stdout,"The position of the first gaussian is ",g%a%pos)
   call show_(stdout,"The exponent of the first gaussian is ",g%a%ex)

   call flush_(stdout)
   call text_(stdout,"Change the precision of the output to 3 decimal")
   call text_(stdout,"places, and a field width of 8 characters")
   call set_real_precision_(stdout,3)
   call set_real_width_(stdout,8)

   call flush_(stdout)
   call text_(stdout,"Analytically evaluate the nuclear attraction")
   call text_(stdout,"integrals for gaussian pair assuming the nucleus")
   call text_(stdout,"of unit charge is at (0,1,0) :- ")
   call flush_(stdout)
   call make_nuclear_attraction_ints_(g,S,c=(/0.0d0,1.0d0,0.0d0/))
   call put_(stdout,S)

   call report_(tonto)
end program
```

Fig. 2. The Fortran 95 source generated by the `Foo` preprocessor, corresponding to the source code in Figure 1

than a value. Procedures in `Foo` have a similar attribute declaration style (see Figure 4).

Note that in the Fortran 95 source, every routine has an underscore character appended relative to the corresponding `Foo` routine. These underscored rou-

```
First, define a gaussian pair ...

GAUSSIAN2:

l_a   =          0
l_b   =          1
pos_a =   0.000000   0.000000   0.000000
pos_b =   0.000000   0.000000   0.100000
ex_a  =   0.300000
ex_b  =   0.500000

The position of the first gaussian is    0.000000   0.000000   0.000000
The exponent of the first gaussian is    0.300000

Change the precision of the output to 3 decimal
places, and a field width of 8 characters

Analytically evaluate the nuclear attraction
integrals for gaussian pair assuming the nucleus
of unit charge is at (0,1,0) :-

               1       2       3

        1   0.000   1.651  -0.334

SYSTEM: Memory usage report:

Memory used                =       110 Words
Maximum memory used        =    390718 Words
Memory blocks used         =         3
Maximum memory blocks used =        14
Call stack level           =         0
Maximum call stack depth   =        10
```

Fig. 3. Output corresponding to the programs in Figures 1 and 2

tines have been overloaded. Generic Fortran 95 function interfaces (not shown) have been generated automatically by `Foo` when compiling the modules. Unfortunately, this method generates a very large number of generic names, and compilers have great difficulties to deal with the resulting name space. `Foo` can use another method to generate the generic names: it can prepend each routine name with the name of the module it belongs to. This greatly reduces the number of generic names which are the same, and consequently reduces compilation requirements. Unfortunately, the names generated by this method are often long and exceed the standard length of 31 characters. This restriction will be alleviated in the Fortran 2000 standard when 63 characters. Some compilers already allow long character names.

Note how the dot notation operates: if the `Foo` code refers to a routine, then the corresponding code expands into a procedure call, but if the dot refers to a type component, the corresponding Fortran code uses a % symbol. From the `Foo` programmer's perspective, there is no difference between the two. Thus, the internal type structure of the module can be changed without affecting the code which uses it: functions can replace type components and vice-versa. (Of course, in high performance applications, one must be careful about unnecessary use of

function calls, so although the calling code will work, it will probably need to be rewritten for fast execution).

Memory management is dealt with largely automatically by the system. The `allocate` and `deallocate` routines are replaced in the Tonto system by `create` and `destroy` routines. The amount of memory, and which routines are called by the system, are stored in the `tonto` system variable. A printout of the call stack is available at any point in the program (provided that support for this feature was compiled into it, as an option). In principle, profiling features could also be incorporated, but we have not yet done so. At the end of the program, we print out details of the statistics of memory use.

The style of the code is also very important – perhaps as important as the use of object based techniques. Long descriptive names are used (up to 31 characters) and underscores are used liberally to make reading easier. Although typing these names can be annoying, there is a significant payoff in terms of clarity. The whole Tonto system is written this way.

```
append(value) ::: leaky
! Expands self and appends the single scalar "value" onto the end.
  self :: PTR
  value :: ELEMENT_TYPE, IN
  dim :: INT
  if (.destroyed) then; dim = 0
  else;                 dim = .dim
  end
  .expand(dim+1)
  self(dim+1) = value
end
```

Fig. 4. `Foo` source for appending a scalar value to an array, appearing in the `INTRINSICVEC` module. Notice the type parameter `ELEMENT_TYPE`

```
append(value) ::: leaky, get_from(INTRINSICVEC)
! Expands self and appends the single scalar "value" onto the end.
  self :: PTR
  value :: ELEMENT_TYPE, IN
end
```

Fig. 5. Inherited `Foo` source for appending a string value to a string array, appearing in the `STRVEC` module. `ELEMENT_TYPE` has the value `STR` in this module

3.2 A Short Example of Template Inheritance

Figure 4 shows a code fragment for appending an element to a vector, and makes use of the predefined type parameter `ELEMENT_TYPE`. This code appears in the `INTRINSICVEC` "virtual" module – a module which is not compiled, but comprises

routines that are "inherited" by other modules. Such modules are called "virtual classes", "abstract types" , or "template classes" in other languages. Figure 5 shows how the inclusion of this code is effected in the `STRVEC` module, a module concerned with one dimensional arrays of strings. In this module, `ELEMENT_TYPE` is replaced by the type `STR`, which is the type corresponding to an element of the array of type `STRVEC`. Notice the use of the `get_from` procedure attribute in `Foo`. The `leaky` attribute indicates that a memory leak occurs in the routine (a warning message will not be generated in this case). Code is inherited only if the routine signatures are identical. Notice also that the body of the routine in the `STRVEC` module is absent. The source code from `INTRINSICVEC` is inserted verbatim.

4 Concluding Remarks

There are, of course, a number of ways to improve the `Foo` language. The syntax of the parameterised type and template inheritance mechanism could be refined. We have also noted that with very large modules, compile times can be very long. Clearly, some syntax for *submodules* needs to be defined, such as has been recommended by the Fortran standards body [18]. The preprocessor code itself is a rather ad-hoc `perl` script: a proper parser-generator should probably be written, perhaps using the `RecDescent` package [19]. Due to the similarity of `Foo` and Fortran, we have not attempted a full BNF syntax specification yet, but this should be done.

Despite these problems, we believe the `Foo` language offers a seamless way to take almost full advantage of the latest Fortran 2000 standard, and features beyond it, without waiting for vendors to produce compilers that adequately support the new standard — a development that may be 5 years or more away.

References

1. B. Meyer. *Object-Oriented Software Construction*. Prentice Hall, Inc., second edition, 1997.
2. J. J. P. Stewart. *MOPAC 2002*. Fujitsu Limited, Tokyo, Japan, 1999.
3. Æ. Frisch and M. J. Frisch. *Gaussian 98 User's Reference*. Gaussian, Inc., Pittsburgh PA, second edition, 1999.
4. B. J. Dupee. Object oriented methods using fortran 90. *ACM Fortran Forum*, 13:21–30, 1994.
5. R. Lutowski. Object-oriented software development with traditional languages. *ACM Fortran Forum*, 14:13–18, 1995.
6. C. D. Norton, V. K. Decyk, and B. K. Szymanski. On parallel object oriented programming in fortran 90. *ACM SIGAPP Applied Computing Review*, 4:27–31, 1996.
7. C. D. Norton, V. K. Decyk, and B. K. Szymanski. High performance object-oriented programming in fortran 90. In *Proc. Eighth SIAM Conference on Parallel Processsing for Scientific Computing*, March 1997.

8. J. R. Cary, S. G. Shasharina, C. Cummings, J. V. W. Reynders, and P. J. Hinker. Comparison of c++ and fortran 90 for object-oriented scientific programming. *Computer Phys. Comm.*, 105:20–36, 1997.
9. V. K. Decyk, C. D. Norton, and B. K Szymanski. Expressing object-oriented concepts in fortran 90. *ACM Fortran Forum*, 16:13–18, 1997.
10. L. Machiels and M. O. Deville. Fortran 90: An entry to object-oriented programming for the solution of partial differential equations. *ACM Transactions on Mathematical Software*, 23:32–49, 1997.
11. M. G. Gray and R. M. Roberts. Object-based programming in fortran 90. *Computers in physics*, 11:355–361, 1997.
12. V. K. Decyk, C. D. Norton, and B. K. Szymanski. How to support inheritance and run-time polymorphism in fortran 90. *Computer Phys. Comm.*, 115:9–17, 1998.
13. V. Snyder. Constructive uses for include. *ACM Fortran Forum*, 20:2–4, 2001.
14. A. Markus. Generic programming in fortran 90. *ACM Fortran Forum*, 20:20–23, 2001.
15. B. Stroustrup. *The Design and Evolution of C++*. Addison-Wesley, 1994.
16. S. Khoshafian and R. Abnous. *Object Orientation*. John Wiley & Sons, Inc, second edition, 1995.
17. T. Stern and D. Grimwood. Cascade compilation revisited. *ACM Fortran Forum*, 21:12–24, 2002.
18. JTC1/SC22/WG5. Enhanced module facilities in fortran. Technical Report 19767, ISO/IEC, 2002.
19. D. M. Conway. The man(1) of descent. *The Perl J.*, 3:46–58, 1998.

Explicitly Correlated Second Order Perturbation Theory with Frozen Gaussian-Type Geminals

Seiichiro Ten-no

Graduate School of Human Informatics, Nagoya University,
Chikusa-ku Nagoya 464-8601 Japan
tenno@info.human.nagoya-u.ac.jp

Abstract. We report on the performance of the explicitly correlated second order perturbation theory with frozen Gaussian-type geminals. Components in the Hylleraas energy functional are analyzed using the ansatz based on the *s*- and *p*-wave cusp conditions for singlet and triplet pairs. It is shown that calculated correlation energies are improved substantially by the introduction of the frozen geminals.

1 Introduction

In calculations with correlated methods, the basis set convergence is crucial for the predictions of reliable energetics and properties. It has been shown that enormous basis functions are required to achieve the chemical accuracy. This is the direct consequence of the inability of describing the Coulomb cusp[1] with one-electronic basis. To ameliorate this feature, various alternatives incorporating explicitly correlated functions have been proposed. The Gaussian-type geminals (GTGs)[2,3,4] have been used in many places. Especially, Szalewicz and coworkers introduced a novel alternative in the Hylleraas functional with the requirement of at most 3-electron integrals[5]. Although Gaussian-type functions never become exact for the cusp condition, the methods have shown the efficiency predicting correlation energy to an accuracy of $1\mu Eh$, as demonstrated by Cencek and Rychlewski[6] This situation is comprehensible by evaluating the motion of pair-electrons in the three-dimension space, *i.e.* the gain from the cusp around a fixed electron scales as $4\pi r_{12}^2$, vanishing at $r_{12} = 0$.

The R12 methods [7,8,9,10] developed by Kutzelnigg and coworkers utilize the linear r_{12} behavior with systematic approximations using the resolution of identity (RI) for bypassing the explicit treatment of 3- and 4-electron integrals. A large basis set is, however, required to guarantee the accuracy of RI with the non-vanishing ansatz. The long-range nature of the linear r_{12} behavior also makes a scalable treatment less straightforward. As a compromise between GTG and R12, Persson and Taylor proposed a method which fits the linear r_{12} behavior with GTG to avoid the nonlinear optimization[11]. The method aims at calculations of modest accuracy in comparison with the previous GTG theories, i.e. sub-*mEh* accuracy for valence

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 152–158, 2003.
© Springer-Verlag Berlin Heidelberg 2003

correlations within the framework of the second order many-body perturbation theory (MBPT2).

All of the above methods use explicitly correlated functions with orthogonal projectors and the Hylleraas energy functional. The transcorrelated method [12,13] of Boys and Handy is on the basis of a correlation factor independent of pairs. The similarity transformed Hamiltonian includes at most three-body effective interactions. Recently, we developed a new transcorrelated method with frozen GTG especially for pair-wise short-range collisions [14,15]. The method does not require any optimization of the correlation factor making use of the cusp condition. Instead, the long-range correlations are dealt with in terms of the usual configuration interaction (CI)-type expansion. In this paper, we study the second order Hylleraas energy functional of frozen GTG. When the geminals are universal for correlated pairs, the component of the kinetic energy operator is closely related to the transcorrelated Hamiltonian through the virial theorem. The effect of the *p*-wave cusp condition is also examined for triplet pairs. Numerical results are given in Sec. 3. and the conclusion is depicted in Sec. 4.

2 MBPT2 with Frozen GTG

Henceforward, we denote orthonormalized occupied and general orbitals in a given basis set as $ij\ldots$ and $pq\ldots$ respectively. In the GTG method, the second order energy is expressed as a sum of the usual MBPT2 energy and a correction in the Hylleraas energy functional outside the Hilbert space spanned by the given basis set,

$$E_V^{(2)} = E^{(2)} + \Delta E_V^{(2)} = E^{(2)} - \Delta E_N^{(2)} + 2\Delta E_Z^{(2)}, \tag{1}$$

$$\Delta E_N^{(2)} = \sum_{SM} \sum_{i \geq j} (c_{ij}^{(S,M)})^2 \left\langle \{ij\}^{(S,M)} \middle| K_{12}^{(L)} (Q_1^* Q_2^* - Q_1 Q_2) f_{12} \middle| \{ij\}^{(S,M)} \right\rangle, \tag{2}$$

$$\Delta E_Z^{(2)} = \sum_{SM} \sum_{i \geq j} c_{ij}^{(S,M)} \left\langle \{ij\}^{(S,M)} \middle| r_{12}^{-1} (Q_1^* Q_2^* - Q_1 Q_2) f_{12} \middle| \{ij\}^{(S,M)} \right\rangle, \tag{3}$$

where Q_n^* and Q_n are one-electron projectors in the virtual spaces for the complete and given basis sets, respectively, $\{ij\}^{(S,M)}$ denotes the spin-adapted antisymmetrized pair functions, $c_{ij}^{(S,M)}$ are variational parameters, and the operator, $K_{12}^{(L)}$, is an anti-hermite single commutator between the explicitly correlated geminal, f_{12}, and the kinetic energy operator,

$$K_{12}^{(L)} = [T_1 + T_2, f_{12}] = -(\nabla_1^2 f_{12}) - (\nabla_1 f_{12}) \cdot (\nabla_1 - \nabla_2), \tag{4}$$

f_{12} is expanded as a linear combination of spherically symmetric GTGs. The quantum numbers of the pair functions take the values, $(S,M) = (0,0)$, $(1,0)$, $(1,\pm 1)$ for $i \neq j$ and $(S,M) = (0,0)$ for $i = j$. The commutator involving the exchange operator is neglected in this particular work as in the MP2-R12/A ap-

proximation[9]. All three energy corrections become identical, $\Delta E_V^{(2)} = \Delta E_N^{(2)} = \Delta E_Z^{(2)}$, when the geminal is exact outside the orbital space,

$$(Q_1^* Q_2^* - Q_1 Q_2)(c_{ij}^{(S,M)} K_{12}^{(L)} + r_{12}^{-1}) \left| \{ij\}^{(S,M)} \right\rangle = 0, \quad \forall ijSM \ . \tag{5}$$

Thus the ratio,

$$\chi = \Delta E_Z^{(2)} / \Delta E_N^{(2)} , \tag{6}$$

which becomes unity in the above condition, is a good measure to indicate the appropriateness of geminals. It is possible to think of the connection between $\Delta E_V^{(2)}$ and $\Delta E_N^{(2)}$ as the virial theorem for the cusp condition. When $c_{ij}^{(S,M)}$ are coincident for all pairs, $\Delta E_N^{(2)}$ is directly obtained from the order-by-order expansion of the Schrödinger equation for the transcorrelated Hamiltonian [14]. In the transcorrelated method, the first order cusp condition in f_{12} is renormalized to infinite order in the similarity transformed Hamiltonian.

According to the analytical resolution,

$$Q_1^* Q_2^* = 1 - P_1 - P_2 + P_1 P_2 , \tag{7}$$

the functional is divided as

$$\Delta E_N^{(2)} = \Delta E_N^{(2)}[1] - \Delta E_N^{(2)}[P_1 + P_2] + \Delta E_N^{(2)}[P_1 P_2] - \Delta E_N^{(2)}[Q_1 Q_2] , \tag{8}$$

$$\Delta E_N^{(2)}[O] = \sum_{SM} \sum_{i \geq j} (c_{ij}^{(S,M)})^2 \left\langle \{ij\}^{(S,M)} \left| K_{12}^{(L)} O f_{12} \right| \{ij\}^{(S,M)} \right\rangle , \tag{9}$$

and a similar expression for $\Delta E_Z^{(2)}$. The component, $\Delta E_N^{(2)}[1]$, can be rewritten in the commutator form leading to the operator, $K_{12}^{(Q)}$, which appears in the transcorrelated method,

$$\Delta E_N^{(2)}[1] = \sum_{SM} \sum_{i \geq j} (c_{ij}^{(S,M)})^2 \left\langle \{ij\}^{(S,M)} \left| K_{12}^{(Q)} \right| \{ij\}^{(S,M)} \right\rangle , \tag{10}$$

$$K_{12}^{(Q)} = \frac{1}{2} [K_{12}^{(L)}, f_{12}] = -(\nabla_1 f_{12}) \cdot (\nabla_1 f_{12}) . \tag{11}$$

The term with single projectors reduces to 3-electron integrals and is approximated using RI of Kutzelnigg and Klopper[8], in the spin orbital basis as,

$$\left\langle ij \left| K_{12}^{(L)} P_1 f_{12} \right| kl \right\rangle \cong \sum_{mp} \left\langle ij \left| K_{12}^{(L)} \right| mp \right\rangle \left\langle mp \left| f_{12} \right| kl \right\rangle , \tag{12}$$

Since the 2-electron integrals in the RI expressions involve three occupied orbitals, the maximum angular momentum required in the RI basis becomes $3L_{occ}$ for the highest occupied angular momentum, L_{occ}, in an atomic calculation. Thus *s*-, *f*-, and *i*-functions contribute for systems with *s*-, *p*-, and *d*-occupied shells, respectively. Very

recently, making use of density fitting, we developed a novel decomposition scheme, which reduces the requirement to $2L_{occ}$ [16]. We do not employ the scheme in this work, focusing on molecules without *d*-occupied orbitals. The performance of the approximation is investigated using the index of RI,

$$\tau = \Delta E_N^{(2)}[P_1 + P_2] - 2\Delta E_Z^{(2)}[P_1 + P_2] + 4\Delta E_Z^{(2)}[P_1 P_2] \tag{13}$$

The term, $\Delta E_Z^{(2)}[P_1 P_2]$, dependent on the description of occupied orbitals has relatively large amplitude whereas $\Delta E_N^{(2)}[P_1 P_2]$ is zero because of the symmetry of the operator, $K^{(L)}$. The exact energy for a given basis and f_{12},

$$\bar{E}_V^{(2)} = E_V^{(2)} - \tau + \bar{\tau} \tag{14}$$

is estimated with a reliable $\bar{\tau}$ of an augmented basis set.

We bypass the optimizations of the parameters, $c_{ij}^{(S,M)}$, the coefficients and exponents of GTGs, using a template geminal, f_{12}^t, expanded as a linear combination of 10 Gaussians. It is determined in such a way that the Coulomb repulsion multiplied by a short-range weight Gaussian is suppressed in the similarity transformed Hamiltonian [14]. For the exponents, we use an even-tempered sequence in the range between 10^6 and 0.5 along with the weight Gaussian with the exponents, 5. f_{12}^t is further transformed with a scaling parameter, c_Z, as

$$f(r_{12}; c_Z) = c_Z^{-1} f^t(c_Z r_{12}). \tag{15}$$

Fig. 1 shows some profiles of $f(r_{12}; c_Z)$ with different scaling parameters. The effective radius of an explicitly correlated function is of c_Z with a template function whereas the approximate slope, $1/2$, is maintained around $r_{12} = 0$ by the scaling. As for the parameter, $c_{ij}^{(S,M)}$, Klopper developed a unitary invariant formulation to pair functionals in the MP2-R12 theory [17]. Such a convention is not employed in this work to keep the short-range behavior of f_{12}. Hence we use $c_{ij}^{(0,0)} = 1$, for all singlet pairs. The triplet pairs $(S = 1)$ are antisymmetric in the spatial part to follow the *p*-wave cusp condition[18] and the parameters,

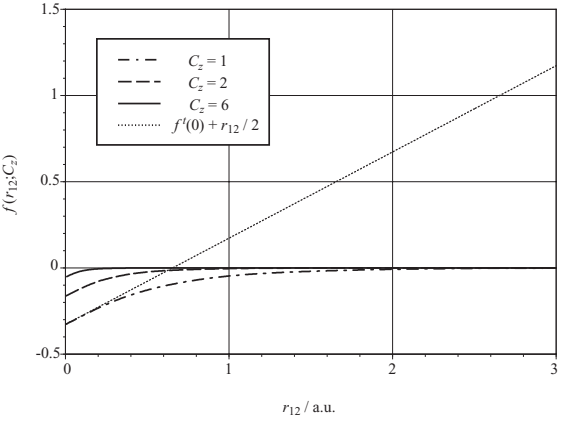


Fig. 1. Frozen geminals as functions of c_Z

$c_{ij}^{(1,M)} = 1/2$, are used for the pairs. The latter condition is not crucial especially for the use of a short-range geminal because of the Fermi hole at $r_{12} = 0$.

Table 1. Correlation energies of the Ne atom in mEh. All electrons are correlated

Basis set[b]	c_z	$\tau - \overline{\tau}$ [a]	χ	$E_V^{(2)}$	$\overline{E}_V^{(2)}$	$E^{(2)}$
VDZ	1.0	-41.05	0.82	-381.69	-340.6	-255.48
(9*s*4*p*1*d*)	2.0	-16.54	1.08	-351.25	-334.7	
	3.0	-7.89	1.24	-322.06	-314.1	
VTZ	1.0	-6.98	0.76	-377.39	-370.4	-321.93
(10*s*5*p*2*d*1*f*)	2.0	-3.58	0.98	-375.26	-371.7	
	3.0	-2.14	1.16	-364.33	-362.2	
VQZ	1.0	-1.52	0.75	-381.32	-379.8	-351.22
(12*s*6*p*3*d*2*f*1*g*)	2.0	-0.87	0.89	-382.86	-382.0	
	3.0	-0.63	1.06	-379.48	-378.9	
CVTZ	1.0	-5.40	0.86	-382.03	-376.6	-333.49
(12*s*7*p*3*d*1*f*)	2.0	-1.79	1.05	-374.46	-372.7	
	3.0	-0.91	1.19	-364.59	-363.7	
AVDZ	1.0	-27.45	0.80	-395.07	-367.6	-278.21
(10*s*5*p*2*d*)	2.0	-15.32	1.07	-371.26	-355.9	
	3.0	-7.66	1.23	-343.51	-335.9	
AVTZ	1.0	-3.20	0.75	-381.83	-378.6	-330.17
(11*s*6*p*3*d*2*f*)	2.0	-3.24	1.12	-382.35	-379.1	
	3.0	-2.06	1.14	-371.64	-369.6	
ACVQZ[c]	1.0	-13.43	0.97	-399.84	-386.4	-320.08
(16*s*10*p*6*d*)	2.0	-5.60	1.32	-373.57	-367.9	
	3.0	-2.44	1.53	-354.50	-352.1	
ACVQZ[c]	1.0	0.00	0.89	-387.55	-387.5	-354.94
(16*s*10*p*6*d*4*f*)	2.0	0.00	1.12	-382.35	-382.4	
	3.0	0.00	1.29	-375.56	-375.6	
ACVQZ[c]	1.0	0.00	0.85	-387.14	-387.1	-365.91
(16*s*10*p*6*d*4*f*2*g*)	2.0	0.00	1.00	-386.08	-386.1	
	3.0	0.00	1.17	-382.69	-382.7	
Limit						-388.1[d]

[a] The reference values of the index, $\overline{\tau}$, are based on the calculations with the aug-cc-pCVQZ (uncontracted) set. It takes the values, 1354.27, 447.03, and 200.98 *mEh* for c_z =1.0, 2.0, and 3.0, respectively.
[b] The uncontracted set of the primitives in cc-p(C)VXZ and aug-cc-pCVXZ sets.
[c] Angular subcomponents of the reference primitive set, ACVQZ (uncontracted).
[d] A estimate value in the MBPT2 limit taken from Ref.[1].

3 Results and Discussions

Table 1 shows the results of the Ne atom with the uncontracted functions in the correlation consistent basis sets [20,21]. We took the reference value of the RI index, $\overline{\tau}$, from the calculations with aug-cc-pCVQZ (uncontracted) set, which is sufficiently accurate to ca. 0.2mEh [16]. The errors in RI, $\tau - \overline{\tau}$, are significant especially for the

use of a small basis set and a large c_z. The positive increment of $E_V^{(2)}$ with $c_z = 1$ from VDZ to VTZ is mostly an artifact of the insufficiency of the DZV set, in which the *f*-shells are absent. The energy, –399.84 *mEh*, in the calculation of (16*s*10*p*6*d*) and $c_z = 1$ is also lower than the MBPT2 limit by ca. 12 *mEh* artificially. It is preferable to use the decomposition scheme with density fitting to bypass the requirement of the *f*-shells[16]. The slightly positive increment from (16*s*10*p*6*d*4*f*) to (16*s*10*p*6*d*4*f*2*g*) is due to the neglect of the commutator between the exchange and geminal operators since the augmentation of the *g*-shell does not affect the accuracy of RI.

The electron correlation is a short-range phenomenon. Thus the calculation with VXZ sets shows that the results become less sensitive to the choice of c_z with the increase of the cardinal number, X. The series of ACVQZ shows quite different behavior; the estimated energy, $\bar{E}_V^{(2)}$, is almost saturated at the *d*-shell and is less sensitive to the augmentation of angular functions at $c_z = 1$. In the difference between VTZ and CVTZ, the results with $c_z = 2$ and 3 are hardly altered whereas there is an improvement, ca. 5mEh for $c_z = 1$. This implies that the geminal with a large radius is inappropriate for correlations concerning core electrons and the situation is improved significantly by the augmentation of the core correlation functions.

Table 2. MBPT2 energies (in *mEh*) of some typical molecules computed with the cc-pVTZ basis set (uncontracted) and $c_z = 2$

Molecule	χ	$E^{(2)}$	%	$E_V^{(2)}$	%	Limit[a)]
CH_2	1.00	-183.79	87.6	-202.15	96.3	-209.9
CH_4	1.05	-243.89	89.1	-265.34	97.0	-273.6
NH_3	1.05	-283.32	87.7	-310.81	96.3	-322.9
H_2O	1.06	-318.46	87.9	-347.45	96.0	-362.1
HF	1.01	-326.07	84.8	-370.27	96.3	-384.6
Ne	0.98	-321.93	83.0	-375.26	96.7	-388.1
CO	1.02	-450.42	86.7	-501.71	96.5	-519.7
N_2	1.02	-469.31	87.4	-518.20	96.5	-536.9
F_2	1.00	-630.61	85.1	-714.74	96.5	-740.6

[a)] The values in the Refs. [19] and [22]

We show the calculated MBPT2 energies of some molecules with cc-pVTZ basis set (uncontracted) and $c_z = 2$ in Table 2. The conventional MBPT2 covers from 83% (Ne) to 89% (CH_4) of the correlation energies in the complete basis limits. In explicitly correlated calculations, the ratio, χ, is dependent on the choice of c_z and basis set as seen in the Ne result. It however ranges in a small area between 0.98 (Ne) and 1.06 (H_2O) irrespective of the system. Generally speaking, the error of RI reduces in the calculation of a molecule compared to atoms. The effects are thus estimated less than 1% (for Ne) in the correlation energy. The use of the frozen GTG reproduces ca. 96% of the MBPT2 energies reducing the correlation error by 2/3–3/4 compared to the conventional expansion.

4 Conclusion

We have illustrated the MBPT2 method with frozen GTG. Instead of minimizing the pair energy functional, the *s*- and *p*-wave cusp conditions are utilized in adapting the geminal. It is shown that the present method improves the calculation of MBPT2 energy substantially.

Acknowledgement. The author acknowledges the support of the Grant-in-Aids for Scientific Research (B) (No. 13440177) from the Japan Society for the Promotion of Science (JSPS).

References

1. T. Kato Commun. Pure Appl. Math. 10 (1957) 151
2. S. F. Boys, Proc. Roy. Soc. London Ser. A258 (1960) 402
3. K. Singer, Proc. Roy. Soc. London Ser. A258 (1960) 412
4. K.-C. Pan and H. F. King, J. Chem. Phys. 56 (1972) 4667
5. K. Szalewicz, B. Jeziorski, H. J. Monkhorst and J. G. Zabolitzky, J. Chem. Phys. 78 (1983) 1420
6. W. Cencek and J. Rychelewski, Chem. Phys. Lett. 320 (2000) 549
7. W. Kutzelnigg, Theor. Chim. Acta. 68 (1985) 445
8. W. Kutzelnigg, W. Klopper, J. Chem. Phys. 94 (1991) 1985
9. V. Termath, W. Klopper and W. Kutzelnigg, J. Chem. Phys. 94 (1991) 2002
10. W. Klopper and W. Kutzelnigg, J. Chem. Phys. 94 (1991) 2020
11. B. J. Persson, P.R. Taylor, J. Chem. Phys. 105 (1996) 5915
12. S. F. Boys and N. C. Handy, Proc. R. Soc. A 310 (1969) 43
13. N. C. Handy, Mol. Phys. 26 (1973) 169
14. S. Ten-no, Chem. Phys. Lett. 330 (2000) 169; *ibid*. 330 (2000) 175
15. O. Hino, Y. Tanimura and S. Ten-no, J. Chem. Phys. 115 (2001) 7865; O. Hino, Y. Tanimura and S. Ten-no, *Chem. Phys. Lett.* 353 (2002) 317
16. S. Ten-no and F. R. Manby, J. Chem. Phys., to appear. (2003)
17. W. Klopper, Chem. Phys. Lett. 186 (1991) 583
18. V. A. Rassolov and D. M. Chipman, J. Chem. Phys. 104 (1996) 9908
19. W. Klopper and C.C. M. Samson, J. Chem. Phys. 116 (2002) 6397
20. T. H. Dunning, Jr., J. Chem. Phys. 90 (1988) 1007
21. D. E. Woon and T. H. Dunning, Jr., J. Chem. Phys. 103 (1995) 4572
22. "CC-R12: An Explicitly Correlated Coupled-Cluster Theory", J. Noga, W. Klopper and W. Kutzelnigg, in R. J. Bartlett (ed) "Recent Advances in Coupled-Cluster Methods" World Scientific (1997)

Molecular Potential Energy Surfaces by Interpolation

Michael A. Collins

Research School of Chemistry, Australian National University, Canberra. ACT 0200 Australia. collins@rsc.anu.edu.au

Abstract. The molecular potential energy surface governs the motion of the atomic nuclei for a molecule in an isolated electronic state. For a molecule of N atoms, this surface is a function of 3N-6 internal coordinates which determine the shape of the molecule. For molecules undergoing chemical reaction, the surface is a relatively complicated function of these many coordinates. Methods have now been developed which allow us to construct this surface as an interpolation of Taylor expansions of the surface around molecular configurations scattered throughout the accessible space.

1 Introduction

Many important chemical reactions occur for molecules in a single electronic state; that is where the wavefunction for the electrons is given by a single eigenfunction of the time-independent Schrödinger equation with the positions of the nuclei fixed in space. The total electronic energy is the eigenvalue. This energy depends on the position of the nuclei. The motion of the atomic nuclei is determined by a Hamiltonian which is just the sum of the nuclear kinetic energy and this total electronic energy – the potential energy of the nuclei. The motion of the nuclei can be evaluated by solving the associated Schrödinger Equation, or approximately, by solving the classical Hamilton's equations. In either case, we must evaluate the electronic energy as a function of the nuclear positions.

Why do we want to evaluate the motion of the nuclei? Because this motion tells us everything there is to know about the mechanism of the chemical reaction, about what new molecules are formed, and about how fast the reaction occurs. Reaction rates are important facts to possess, because the outcome of any practical chemical system is the result of competing processes, and far from equilibrium, speed is more important than stability. Chemists strive to understand the mechanism of reactions in the expectation that this knowledge will provide the means of designing new reaction sequences for novel chemical syntheses.

Unfortunately, the molecular potential energy surface (PES) has proved to be very difficult to evaluate. The value of the PES at any molecular configuration can now be evaluated with adequate accuracy for many small to medium-sized molecules using the methods of *ab initio* quantum chemistry. However, the cost of such calculations can be high and the number of configurations at which the energy must be evaluated

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 159–167, 2003.
© Springer-Verlag Berlin Heidelberg 2003

is very large for a molecule undergoing chemical reaction. Pessimistically, this number is of the order of $d^{(3N-6)}$, where d is the number of configurations needed for each configurational degree of freedom.

So-called direct dynamics methods attempt to evaluate the classical dynamics of chemical reactions using Monte Carlo sampling to minimise the number of electronic structure calculations. However, up to now, only very low levels of *ab initio* calculations have been possible with inadequately small samples. The traditional route around this $d^{(3N-6)}$ problem has been to represent the PES as an explicit function of the internal molecular coordinates. This function contains many adjustable parameters. *Ab initio* calculations are performed at a small number of significant molecular configurations, including any known stationary points, and the parameters in the PES function are adjusted to fit this data. Some success has been achieved with this method for reactions involving three atoms, but little progress has been made for larger systems.

Recently, substantial progress has been made in evaluating PES by interpolation of *ab initio* data. For triatomic molecules, very accurate PES have been constructed by Ho, Rabitz and coworkers using a "reproduciing kernel Hilbert space" method.[1-4] Our group has pursued a modified form of Shepard interpolation and applied this successfully to reactions involving several atoms. In order to construct a PES in this way, we need both an accurate interpolation method and an efficient method for deciding where in configuration space the (inevitably) limited number of *ab initio* calculations are performed.

This paper presents a brief description of this method, illustrated by some recent applications to polyatomic reactions. More details can be found elsewhere.[5-11]

2 The Method

The PES is a function of the internal coordinates of the molecule. Any complete set of coordinates would suffice, but for convenience and accuracy we employ the atom-atom distances, $\{R_i\}$. In practice, we actually use the reciprocal distances, $Z_i = 1/R_i$. There are N(N-1)/2 such Z_i and only 3N-6 of them can be independent. We will discuss this "redundant coordinate" problem below. For the moment, let's assume that we can use *ab initio* quantum chemistry to evaluate the energy E(Z) at any molecular configuration we like. Moreover, if we have evaluated E at a set of N_{data} configurations {Z(n)}, we can write the energy at any configuration as a weighted average of these energies:

$$E(Z) \quad = \quad \sum_{n=1}^{N_{data}} w_n\big[Z - Z(n)\big]E(n) \ . \tag{1}$$

If the weights are chosen so that E(Z) = E(n) when Z = Z(n), then Eq.(1) is an interpolation formula, called a Shepard interpolation. A suitable choice of weight function would be

$$w_n = \frac{v_n}{\sum_{j=1}^{N_{data}} v_j} ; \quad (2)$$

$$v_n[Z - Z(n)] = \|Z - Z(n)\|^{-p} . \quad (3)$$

The so-called primitive weight v_n diverges with some positive power p as $Z \rightarrow Z(n)$ to ensure that the relative weight $w_n \rightarrow 1$ as $Z \rightarrow Z(n)$, and that Eq.(1) is an interpolation of the *ab initio* energies. Equation (1) suffers from the drawback that its gradient is zero at all the data point configurations (when p is as large as it needs to be). To avoid this, we use a modified form of Shepard interpolation where the energies E(n) are replaced by Taylor expansions of the energy:[12, 13]

$$E(Z) = \sum_{n=1}^{N_{data}} w_n[Z - Z(n)]T_n[Z - Z(n)]. \quad (4)$$

We have shown that Eq.(4) can provide an accurate description of the PES and its gradient if the Taylor expansion is at least second order in the deviation from the data point, Z - Z(n). This requires that we evaluate *ab initio* the energy gradient and second derivative matrix at Z(n) in addition to the energy. Higher order Taylor expansions are even more accurate, but at present this does not justify the additional *ab initio* computational cost.[7]

We cannot implement Eq.(4) if N > 4: Since the Z_i are not independent, the partial derivatives of the PES (which appear in the Taylor expansions) cannot be evaluated. The reciprocal bondlengths include redundant coordinates. This problem has been discussed in detail elsewhere.[8, 11, 14] The relevant solution here is to choose an independent set of 3N-6 linear combinations of the Z_i as internal coordinates at each data point Z(n). This can be achieved as follows. The deviation of the N(N-1)/2 reciprocal bondlengths from their values at a data point, $\delta Z = Z - Z(n)$, is related to the corresponding displacement of the 3N Cartesian coordinates, δX, by

$$\delta Z = B\delta X , \quad (5)$$

where the matrix B (a variant of the Wilson B matrix [15]) is given by

$$B_{ij} = \frac{\partial Z_i}{\partial X_j}, \; i = 1,..,N(N-1)/2; \; j = 1,..,3N. \quad (6)$$

A singular value decomposition of B reveals that there are only 3N-6 non-vanishing singular values, which reflects the fact that the internal structure of a molecule can only be distorted in 3N-6 independent ways. The singular vectors, u(m), associated with these non-zero singular values, λ(m), provide the means to define 3N-6 independent coordinates:

$$\xi_m = \sum_{i=1}^{N(N-1)/2} u_i(m) Z_i / \lambda(m), \quad m = 1,..3N-6. \tag{7}$$

The singular value decomposition is carried out at each data point Z(n) to define a new set of independent coordinates for use in each Taylor expansion. Eq.(4) is then replaced by a form which can be evaluated for any number of atoms:

$$E(Z) = \sum_{n=1}^{N_{data}} w_n[Z - Z(n)] T_n[\xi - \xi(n)] . \tag{8}$$

One very attractive feature of Eq.(8) is the ease with which symmetry is implemented. Once we have calculated the energy, energy gradient and second derivatives at some molecular geometry, these quantities are also known at all other geometries which are related by the permutation of indistinguishable particles. So, we can easily add all these "indistinguishable geometries" to the data set. This ensures that Eq.(8) gives the same energy at all permuted versions of a molecular configuration, as symmetry requires.

An important refinement of Eq.(8) has been achieved by implementing a more sophisticated weighting function than that in Eq.(3). The primitive weight in Eq.(3) is based on the reasonable premise that the closer a data point Z(n) is to the point where the PES is evaluated, the more accurate should be the estimate T_n. So the primitive weight increases as an inverse power of the distance ||Z - Z(n)||. However, an analysis of the convergence of Eq.(8) with increasing data set size, shows that the simple form of Eq.(3) ensures convergence of the PES but exaggerates the divergence of the weight function as $Z \rightarrow Z(n)$. A more accurate primitive weight function is given by

$$v_n = \left\{ D[Z, Z(n)]^q + D[Z, Z(n)]^p \right\}^{-1}, \text{ where} \tag{9}$$

$$D[Z, Z(n)] = \left\{ \sum_{i=1}^{N(N-1)/2} \left[\frac{Z_i - Z_i(n)}{d_i(n)} \right]^2 \right\}^{\frac{1}{2}} .$$

The power q > 2, but q << p which must be greater than 3N-3. The $d_i(n)$ are "confidence lengths". If the Taylor series, T_n, is acceptably accurate for $Z_i - Z_i(n) < d_i(n)$, then the weight function varies only slowly for such cases. The confidence lengths can be estimated from a Bayesian analysis of the error that T_n makes in estimating the energy gradients at nearby data points.[10] Eqs (2), (9), and (8) provide us with an accurate interpolation formula if the data point locations are well chosen.

Traditionally, the locations of stationary points on the PES (equilibrium reactants and products and saddle points) have been seen as particularly significant for the reaction dynamics. The simplest picture of the reactive dynamics is that of motion along a "reaction path" or "minimum energy path" linking reactants, the saddle point (barrier), and products. So, in this tradition, we choose a set of molecular configurations on such a reaction path as an initial data set to describe the PES via Eq.(8). Already, we have just about exhausted our intuition as to where data points should be located. Henceforth, the important decisions are automated. The automation

is based on the following reasoning. Ultimately the PES is used to simulate the chemical reaction to calculate observables such as the reaction rate or the distribution of energy in the reaction products, or qualitative and quantitative information about the reaction mechanism. The relationship between the PES and such observables is rather obscure. However, it is certain that the value of the PES at configurations which the molecule passes through during the dynamics must play some role in determining the observed dynamics. It is only at these configurations where the PES must be known, *not* throughout the whole 3N-6 dimensional configuration space.

Given an initial set of data points on the minimum energy path for the reaction, the PES of Eq. (11) is defined, albeit inaccurately at configurations far from this path. We simulate the chemical reaction in a standard way by solving the classical (Newtonian or Hamiltonian) equations of motion with initial conditions corresponding to the collision of the reacting molecules. The choice of initial conditions should reflect the observable properties we wish to simulate; for example, the initial vibration-rotation-translational energies or temperature should be at least as high as the values appropriate to any relevant experiments. Typically, a small number (say 10) of trajectories (collisions) are calculated, and the molecular configuration is periodically written to a file during these trajectories. This set of configurations represents a small (typically of order 10^3) sample of the dynamically important portion of configuration space. We choose one of these configurations to be a new data point. Two criteria for this choice are used. One is purely geometrical in that it chooses the sampled point which is optimally near many other sampled points but far from existing data points. The second criterion chooses the sampled point for which the uncertainty, $\sigma(Z)$, in E(Z) is highest:

$$\sigma(Z)^2 \;=\; \sum_{n=1}^{N_{data}} w_n\left[T_n(Z) - E(Z)\right]^2. \tag{10}$$

We assume that if the uncertainty in the PES is large at some Z(k), then it is more likely that the estimated value of E[Z(k)] is inaccurate, and by this criterion we hope to reduce the largest errors in the PES.

Having chosen a new data point, we have a new version of the PES. We repeat the cycle of simulating the reaction and choosing a new data point from a sample obtained by exploring the new PES. The process is iterated until convergence. The PES is deemed to be "converged" when the values of the dynamical observables of interest do not change significantly with increasing data set size. These values are measured by performing large scale classical simulations of the reaction from time to time - say after every 100 new data points are added to the data set.

There are two important aspects of "automated learning" in this process of iteratively constructing a PES (we say we "grow" PES). As each new data point is added the shape of the PES changes. The computer program "learns" more about the shape of the PES at each cycle. In the small set of classical trajectories, the molecule may move to new configurations not previously allowed by the PES, so the program "learns" about new regions of configuration space. In practice, previously unknown reactions and reaction paths can be discovered. The second type of automated learning is achieved by the iterative improvement in the confidence lengths used in the weight function. As the data set grows, the estimation of the confidence lengths becomes more accurate. The program "learns" more accurate values of the PES not

only by accumulating more *ab initio* data, but by learning how to use all its accumulated data more accurately.

The complete process for constructing a PES is automated in a program package called *Grow* which is available from the author at no cost to noncommercial researchers. At present, the package is compatible with the GAUSSIAN program. The interface with GAMESS will be available shortly.

3 Examples

PES have now been constructed for many reactions involving four or more atoms. In this section, results from a few of these systems are used to illustrate the method.

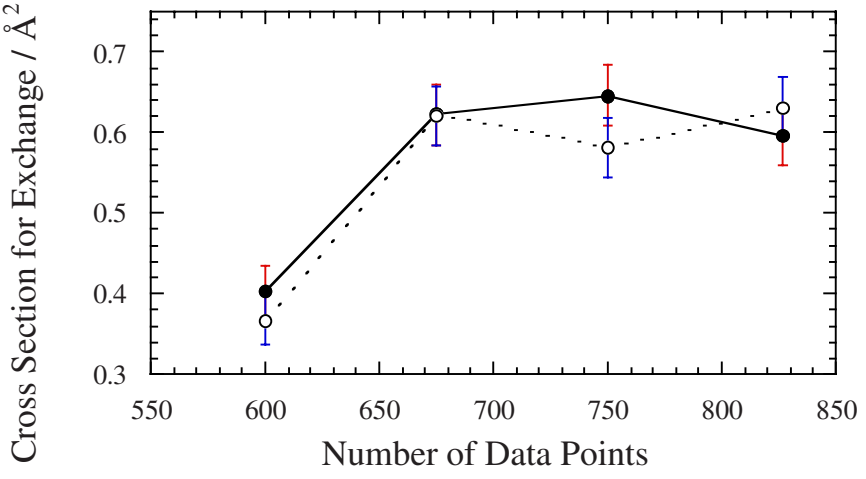


Fig. 1. The classical cross section for the exchange reaction is shown for PESa (o) and PESb (•) as a function of the size of the data set. The error bars correspond to two standard deviations in the trajectory simulations.

The abstraction reaction, H + $H_2O \rightarrow H_2$ + OH, is an important rate determining step in the overall process of hydrogen combustion. This reaction, and its reverse, have also become benchmark systems in the development of quantum reactive scattering calculations. Highly accurate *ab initio* PESs have been constructed for this system and employed to test quantum scattering calculations against experiment.[16-19] This system, and its deuterated analogues, also support the exchange process: H + $H_2O \rightarrow$ HOH + H. The PES for this system was initially grown at the QCISD(T) level of theory using a basis set correction approach: E = E[QCISD(T)/6-311G(d,p)] + E[MP2/6-311++G(3df,2pd)] - E[MP2/6-311G(d,p)]. We denote this surface as PESa. Later the energy (only) of each data point was replaced by the QCISD(T)/6-311++G(3df,2pd) value. We denote this surface as PESb. Figure 1 shows that the

classical cross section for the exchange reaction is converged with respect to the data set size for both surfaces.

This apparent convergence in a dynamical observable is consistent with our general observation, for over ten systems, that the average interpolation error decreases as a function of data set size; usually in a fashion similar to that shown in Figure 2 for the $ArCOH^+$ system.[20] Here 900 molecular configurations were sampled from classical trajectories simulating the collision of Ar and COH^+ and the energy at each geometry was evaluated at the same MP2/6-311G(d,p) level of theory used to construct the PES. The interpolated energy for each of the 900 configurations was evaluated for data sets of increasing size and compared with the directly calculated value. The mean absolute error is shown in Figure 2. The mean error of about 0.8 kJ mol^{-1}, for the largest data set, represents less than 0.3 % of the energy range of these 900 configurations.

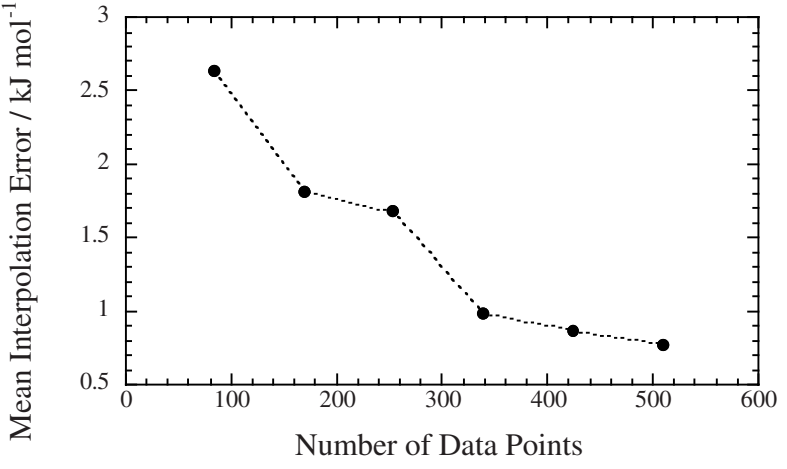


Fig. 2. The mean absolute error in the PES for $ArCOH^+$, estimated from 900 sampled configurations, is shown as a function of the number of data points.

Note that the PES for H_3O was initially constructed with no knowledge of the exchange reaction, and no data points on a pathway for exchange were included in the initial data set. The automated construction process "discovered" the exchange reaction and included data points in that region as required. Figure 3 depicts the data point locations which feature both abstraction and exchange reaction paths in the final data set.

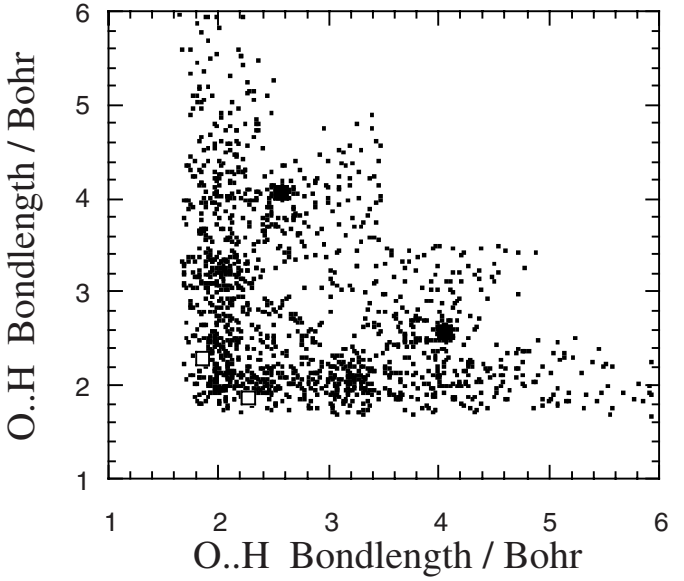


Fig. 3. Part of the data set for H + H_2O is represented as points in a projection onto the two longest OH bondlengths for each data point geometry. The large symbols indicate saddle points for abstraction and exchange.

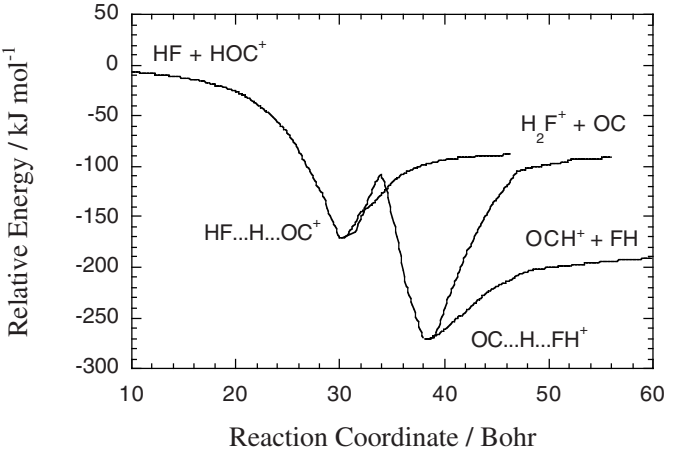


Fig. 4. The energy profile for minimum energy paths on the PES for FH + HOC^+ is shown as a function of a reaction coordinate.[21]

Multiple pathways and competing reactions are common features in the PES for molecules of even as few as four atoms. It seems reasonable to suggest that even greater complexity will be discovered in the PES for larger molecules. Figure 4 depicts the energy profiles for reactions paths on a PES for the five-atom system FH +

COH^+, constructed at the MP2/6-311G(d,p) level of theory. Using a deuterated analogue for clarity, it is clear that four different processes are possible in this system; abstraction, FH + $DOC^+ \rightarrow FHD^+$ + OC; isomerisation, FH + $DOC^+ \rightarrow$ FH + OCD^+, exchange, FH + $DOC^+ \rightarrow$ FD + HOC^+; and isomeration with exchange, FH + $DOC^+ \rightarrow$ FD + OCH^+. Construction of the interpolated PES for this system has provided the means to calculate, classically, the reaction cross sections for all four processes.[21]

References

1. T. Ho, and H. Rabitz, J. Chem. Phys. 104, 2584 (1996).
2. T. Hollebeek, T. Ho, and H. Rabitz, J. Chem. Phys. 106, 7223 (1997).
3. G. C. Schatz, A. Papaioannou, L. A. Pederson, L. B. Harding, T. Hollebeek, T.-S. Ho, and H. Rabitz, J. Chem. Phys. 107, 2340 (1997).
4. T. Hollebeek, T.-S. Ho, H. Rabitz, and L. B. Harding, J. Chem. Phys. 114, 3945 (2001).
5. J. Ischtwan and M. A. Collins, J. Chem. Phys. 100, 8080 (1994).
6. M. J. T. Jordan, K. C. Thompson, and M. A. Collins, J. Chem. Phys. 102, 5647 (1995).
7. M. J. T. Jordan, K. C. Thompson, and M. A. Collins, J. Chem. Phys. 103, 9669 (1995).
8. K. C. Thompson, M. J. T. Jordan, and M. A. Collins, J. Chem. Phys 108, 8302 (1998).
9. K. C. Thompson and M. A. Collins, J. Chem. Soc., Faraday Trans. 93, 871 (1997).
10. R. P. A. Bettens and M. A. Collins, J. Chem. Phys. 111, 816 (1999).
11. M. A. Collins, Theor. Chem. Acc. 108, 313 (2002).
12. R. Farwig, Mathematics of Computation 46, 577 (1986).
13. R. Farwig, in *Algorithms For Approximation,* edited by J.C. Mason and M.G. Cox (Clarendon Press, Oxford, 1987)p. 194.
14. M. A. Collins and D. F. Parsons, J. Chem. Phys. 99, 6756 (1993).
15. E. B. Wilson, J. C. Decius, and P. C. Cross, *Molecular Vibrations* (Dover, New York, 1955)p. 140.
16. M. Yang, D. H. Zhang, M. A. Collins, and S.-Y. Lee, J. Chem. Phys. 115, 174 (2001).
17. D. H. Zhang, M. A. Collins, and S.-Y. Lee, Science 290, 961 (2000).
18. M. Brouard, I. Burak, S. Marinakis, D. Minayev, P. O'Keeffe, C. Vallance, F. J. Aoiz, L. Banares, J. F. Castillo, D. H. Zhang, D. Xie, M. Yang, S.-Y. Lee, and M. A. Collins, Phys. Rev. Lett., in press (2003).
19. M. Brouard, I. Burak, D. Minayev, P. O'Keeffe, C. Vallance, F. J. Aoiz, L. Banares, J. F. Castillo, D. H. Zhang, and M. A. Collins, J. Chem. Phys. 118, 1162 (2003).
20. A. J. Chalk, S. Petrie, L. Radom, and M. A. Collins, J. Chem. Phys. 112, 6625 (2000).
21. M. A. Collins and L. Radom, J. Chem. Phys., in press (2003).

NWChem: New Functionality

Theresa L. Windus, Eric J. Bylaska, Michel Dupuis, So Hirata, Lisa Pollack, Dayle M. Smith, T.P. Straatsma, and Edoardo Aprà

Pacific Northwest National Laboratory,
906 Battelle Blvd.,
Richland, WA 99352

Abstract. NWChem is a computational chemistry code designed for efficient execution on massively parallel computers. Through effective use of the aggregate resources of such computers, NWChem enables much larger and more accurate production simulations than previously feasible. In this paper we briefly describe the NWChem architecture and highlight some of the new capabilities in NWChem.

1 Introduction

Parallel computing has been a part of computational chemistry for over 20 years [1,2] and most major chemistry packages support some degree of parallelism. However, very few computational chemistry packages scale well on large numbers of processors. The massively parallel computational package NWChem [3] is one such package and it was designed and implemented to scale to large numbers of processors.

Only a very brief overview of the capabilities of NWChem is provided here. For a more complete overview of NWChem, the reader is referred to previous articles [3] and to the NWChem web page [4]. NWChem provides many methods to compute the properties of molecular and periodic systems using standard quantum mechanical descriptions of the electronic wavefunction or density. These include Gaussian-orbital based methods such as self-consistent field (SCF) molecular orbital (Hartree-Fock, HF) theory, density functional theory (DFT), second order Møller-Plesset perturbation theory (MP2), coupled cluster (CC) theory, and multi-configuration SCF (MCSCF). They also include plane-wave based DFT methods used in Car-Parrinello simulations. In addition to these quantum chemical approaches, NWChem can also perform classical molecular dynamics and free energy simulations using classical force fields. These methodologies may be combined to perform quantum molecular dynamics and mixed quantum mechanics/molecular mechanics simulations.

The underlying support for the parallel capabilities in the functionality outlined above and in the following sections is provided by the Global Arrays toolkit (GA) [5, 6]. These tools include the Memory Allocator (MA) to access local memory, the Global Arrays (GA) to provide portable shared-memory programming tools, the Aggregate Remote Memory Copy Interface (ARMCI) [7] to provide general-purpose, portable and efficient remote memory copy operations (one-sided communication) optimized for noncontiguous (strided, scatter/gather, I/O vector) data transfers, and the Parallel I/O (ParIO) [8] tool to extend the non-uniform memory architecture model to disk. Libraries from the Global Arrays toolkit interoperate with MPI [9] for message passing.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 168–177, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In this paper, we will briefly discuss key new functionality in the NWChem. The following sections discuss aspects of orbital-dependent density functionals, of the projected augmented wave module, the electron transfer module, single excitation theories, the tensor contraction engine work, and the molecular dynamics module and its application to simulations of biological membranes. We conclude with a brief summary.

2 Orbital-Dependent Density Functionals

Density functional theory is an important tool in computational chemistry, due to its predictive capabilities. The various approximate functionals available for the exchange and correlation energy can be viewed in the context of the Jacob's ladder [10] idea proposed by Perdew and coworkers. Essentially, the ladder has various rungs, each representing a subset of density functionals, with improved chemical accuracy obtained by climbing to higher rungs. The two lowest rungs of the ladder are the LDA (Local Density Approximation) and the GGA (Generalized Gradient Approximation).

There is a general agreement that development on the LDA and GGA levels is basically complete, and that further improvement in chemical accuracy will be reached with higher rungs of the ladder, namely rungs that incorporate the kinetic energy density and/or exact (Hartree-Fock) exchange [10,11]. Some of the most popular functionals in quantum chemistry are the hybrid functionals, which mix in a fraction of exact exchange to GGA exchange. The success of the hybrid functionals indeed suggests that exact exchange contains essential information, and that inclusion of exact exchange could help in the solution of major challenges like self interaction error, time dependent and excited state phenomena, and in the improvement of the accuracy of the exchange-correlation potential [10].

Use of the kinetic energy density and/or exact exchange, means the functionals depend explicitly on the Kohn-Sham orbitals. However, calculating the exchange-correlation potential for an orbital dependent functional cannot be done in the usual manner, by taking the functional derivative of the exchange-correlation energy with respect to the density, although various methods exist to address this problem. Below we discuss which methods are in place, preparing NWChem for the upcoming generation of orbital functionals.

The rung above GGAs on the ladder of approximations is the meta-GGA, which uses the kinetic-energy density (or sometimes the Laplacian of the density) as an additional functional ingredient. The inclusion of the kinetic energy density means the meta-GGA is necessarily an orbital dependent functional and as such must be appropriately handled in the code. There are various ways to implement a meta-GGA.

First, fully self-consistent orbitals and densities obtained from GGA calculations can be used as input to calculate meta-GGA energies. This capability was recently added to the code, as experience suggests that energies obtained this way are similar to fully self-consistent meta-GGA calculations.

Second, using a Gaussian basis, an orbital dependent exchange-correlation potential can be found by energy mimimization with the Kohn-Sham matrix elements. This method [12] is already implemented in NWChem for exact exchange, and the incorporation of the kinetic energy density term should introduce little numerical instability.

Third, an exchange-correlation potential can be obtained from the optimized effective potential (OEP) integral equation, the preferred method of calculating self-consistent meta-GGA energies. Solution of the OEP integral equation is very difficult, and at present has very few implementations for molecules[13]. Recently, a new approach[14] makes the OEP much more practical to implement; a method has been developed to iteratively solve a system of partial differential equations coupled to the Kohn-Sham equations. This promises a more computationally efficient OEP opens up the possibility of fully self-consistent solutions using orbital dependent density functionals in NWChem.

3 Electron Transfer Module

The newly implemented electron transfer (ET) module of NWChem calculates the electronic coupling energy (also called the electron transfer matrix element) between ET reactant and product states. The electronic coupling (V_{AB}), activation energy (ΔG*), and nuclear reorganization energy (λ) are all components of the electron transfer rate defined in Marcus' theory [15].

$$k_{et} = \frac{2\pi}{\hbar} V_{AB}^2 \frac{1}{\sqrt{4\pi\lambda k_B T}} exp[-\frac{\Delta G^*}{k_B T}] \quad (1)$$

These energies are illustrated below in Figure 1, which shows the energy profile for a simple two-state ET system. Curve A shows how the reactant state's potential energy changes along the reaction coordinate, and curve B describes the product state.

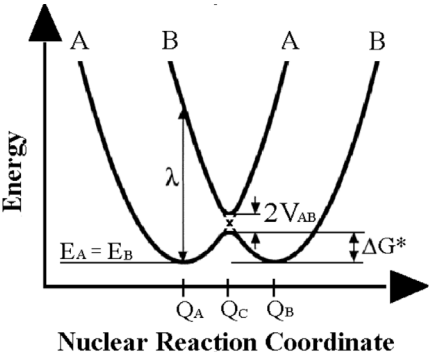


Fig. 1. Energy profile for a simple two-state ET system.

If the system has adequate energy to overcome the activation energy (ΔG^*), and if the electron transfer coupling energy (V_{AB}) is sufficiently large, the system can proceed to ET products, thereby completing the reaction. For organometallic systems, ET is said to be "adiabatic" (100% crossover), when V_{AB} is greater than about 200 cm^{-1}.

The ET module of NWChem utilizes the method of Corresponding Orbital Transformation [16] to calculate V_{AB} from the MO vectors of the reactant and product states, and uses new and existing NWChem routines and GA tools. For instance, the two-electron

contribution to V_{AB}, which is by far the most time consuming, is calculated using the Fock matrix routine from the SCF module. The amount of CPU time required in the calculation of V_{AB} is the same as a single Hartree-Fock iteration, so the most expensive part of a V_{AB} calculation is converging the wavefunctions of the reactant and product states.

The ET module has been applied to a variety of systems. In a recent publication [17] we calculated V_{AB} for a hematite tetramer, which yielded a hematite electron mobility in accord with the experimentally measured value. Similar tests were performed with an aqueous iron dimer, quinone dimers, and several diatomic molecules. Currently the NWChem ET module is being applied to study the electron pathway through flavocytochrome c_3 fumarate reductase (Ifc_3), which participates in the iron respiration of bioremediation bacteria.

4 Single Excitation Theories

NWChem now includes several single excitation theories for vertical excitation energy calculations, such as configuration interaction singles (CIS) [18], time-dependent Hartree-Fock (TDHF, also known as the random-phase approximation), time-dependent density functional theory (TDDFT) [19,20], and Tamm-Dancoff TDDFT [21] within a single algorithmic framework based on Davidson's iterative subspace method [22] or its extension to a non-Hermitian matrix eigenvalue problem [23]. These implementations permit the selective calculations of singlet or triplet excited states (for closed-shell systems) and excited states of a specified spatial symmetry symbol. They also compute oscillator strengths, transition moments, and S^2 spin expectation values of each excited state to assist in band assignments. Although not very efficient, geometry optimizations and frequency calculations can be performed on an excited state potential energy surface by virtue of NWChem's universal finite-difference capability.

The CIS and TDHF excitation energies of closed-shell systems are comparable to each other in accuracy, and are normally considered a zeroth-order description of the excitation process. There are indications that these methods are well balanced in describing Rydberg excited states, in contrast to TDDFT. However, for open-shell systems, the errors in the CIS and TDHF excitation energies are often excessive, primarily due to the multi-determinantal character of the ground and/or excited state wave functions of open-shell systems in a HF reference [24]. The scaling of the computational cost of a CIS or TDHF calculation per state with respect to the system size is the same as that for a HF calculation for the ground state, since the critical step of both methods is the Fock build, namely, the contraction of two-electron integrals with density matrices. In each cycle of Davidson's iterative solution of an eigenvalue problem, we usually need to process more than one density matrix; each density matrix corresponds to a trial vector of Davidson's algorithm. In the implementation of CIS and TDHF within NWChem, we process as many density matrices as fit in the available memory simultaneously; so, the number of two-electron integral evaluations, which are the most expensive in the Fock build, can be minimized. This strategy (multiple Fock build) is standardly adopted in HF and DFT Hessian calculations and the subroutine for this task can be reused in CIS and TDHF with no modification [25].

The accuracy of TDDFT varies with the exchange-correlation functional. In general, the exchange-correlation functionals that are widely used today and are implemented in NWChem work well for low-lying valence excited states. However, for high-lying diffuse excited states and Rydberg excited states in particular, TDDFT employing these conventional functionals breaks down and the excitation energies are substantially underestimated. This is because of the fact that the exchange-correlation potentials generated from these functionals decay too rapidly (exponentially) as opposed to the slow -1/r asymptotic decay of the true potential [26] Unlike CIS or TDHF, TDDFT provides a balanced description of radical excited states [27], and recently NWChem's TDDFT module was used to study 51 species of the radical ions of polycyclic aromatic hydrocarbons with a mean absolute error of ca. 0.3 eV in excitation energies [28]. The computational cost per state of TDDFT calculations scales as the ground state DFT calculations, although the prefactor of the scaling may be much greater in the former. Thus, TDDFT is applicable to fairly large molecules. In addition to yielding excitation energies in general good accord with experiment, it offers the advantage of the orbital picture familiar to chemists when discussing spectral properties.

5 Projected Augmented Wave

In conjunction with the group of Professor John H. Weare (UCSD), we have recently implemented into NWChem a massively parallel version of Blöchl's Projected Augmented Wave (PAW) method [29]. The PAW method is an all-electron density functional theory method that combines the adaptability of linear-augmented-plane-wave methods with the computational advantages of Car-Parrinello methods. As illustrated in Figure 2, with this method the all-electron Kohn-Sham orbital are defined in terms of a smooth wave function (expanded in plane-waves) and a deficit normalization part (expanded using a radial grid about each atom). The computational advantage is that all variational calculations are performed on smooth wave functions, while retaining all-electron information with the deficit normalization part (or augmentation charges).

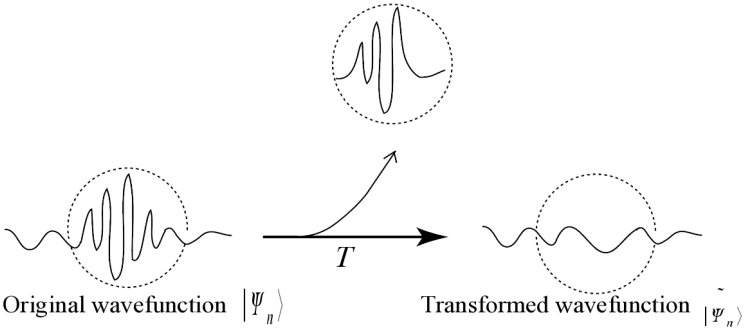


Fig. 2. Using the PAW transformation, the rapid variations around the nuclei are projected onto a local basis set.

The PAW method has become the approach of choice for calculating systems that include transition metals, compared to methods that use pseudopotentials with plane-wave basis sets. Indeed transition metals require extremely "hard" pseudopotentials which necessitate the use of very large and/or complicated basis sets. We note again here the fundamental advantage of plane-wave approaches is the ease of calculating first derivatives compared to approaches using Gaussian atom-centered basis set. With plane-waves, the Pulay first derivative terms (except for a small one-electron part from augmentation charges) do not exist, and their calculations are the most computational demanding step in force calculations. As a result, plane-wave approaches are easily combined with molecular dynamics (time-dependent) calculations.

We have performed simulations on a variety of molecular and solid-state systems using the parallel PAW code [30,31]. Results obtained for solids are found to be to be of good quality and simulations for a series of first-row transition metal dimers, monoxides and dioxides show good agreement with Gaussian DFT simulations and with experiment. These small transition metal-oxide molecules, which are well characterized experimentally, are a challenging test of the reliability of an ab initio method because they have a large number of low lying states, many with high spin multiplicity. These and other results have shown that

- The accuracy of the PAW is similar to the density functional calculations based on local basis sets.
- The convergence with respect to the plane wave basis set leads to practical calculations, even for very difficult systems (F, O, Fe, Cr, etc).
- The method is stable with respect to the choice of the local atomic basis set.
- For a given plane wave basis set size the execution times of PAW calculations are very similar to those of pseudopotential plane-wave methods.
- The size of the plane wave basis is smaller compared to norm-conserving pseudopotential plane-wave methods.

6 Tensor Contraction Engine

With the advent of massively parallel supercomputers, highly accurate many-electron theory calculations are becoming feasible or routine for a wider range of chemical systems. However, such calculations are often hindered not just by the limitation in computer resources but by the complexity of deriving the necessary equations and implementing them into efficient computer programs that support both sequential and parallel executions. Although complex, these symbolic manipulation processes bear much in common across different models of many-electron theory. Specifically, the vast majority of them are defined in the language of second-quantized creation and annihilation operators and the working equations invariably take the form of tensor algebraic expressions that are implemented into a sequence of matrix multiplications [32]. In this light, we have developed a symbolic manipulation program (Tensor Contraction Engine or TCE) automating the entire processes of formula derivation and parallel implementation of various models of many-electron theory [33]. TCE not only expedites the derivation and implementation processes, but it also facilitates parallelization and other laborious optimizations,

enhances portability, maintainability, and extensibility of synthesized programs, and helps design and test new models of many-electron theory.

Provided a definition of many-electron theory as quasi-vacuum expectation values of normal-ordered second-quantized operators, TCE (an interpreted, interactive, object-oriented program written in Python programming language) [33] performs valid contractions of creation and annihilation operators according to Wick's theorem, consolidates identical terms, and reduces the expressions into the form of multiple tensor contractions acted upon by index permutation operators. In various stages of these processes, TCE canonicalizes [34] the expressions of tensors, index permutation operators, etc., to perform rapid pattern matching operations. TCE subsequently performs strength reduction (determines the binary contraction orders with minimal operation and memory costs), factorization (eliminates common binary contractions and defines the intermediate tensors), and identifies reusable intermediates.

The resulting ordered list of binary tensor contractions, additions, and index permutations is translated into an optimized program that is in turn interfaced to NWChem. The programs automatically synthesized by TCE take advantage of spin symmetry, spatial (Abelian point-group) symmetry, and index permutation symmetry simultaneously to minimize the operation cost and storage requirement, adjust the peak local memory usage by index range tiling, and support multiple parallel I/O interfaces and dynamic load balancing for parallel executions. The index range tiling algorithm achieves the best compromise between the conflicting demand for the minimization of the number of arithmetic operations and the minimization of memory requirement. Consequently, the programs synthesized by TCE can compete with carefully hand-coded programs in operation and memory cost.

As an initial application, we invoked TCE to automatically derive the formulae of various models of spin-unrestricted configuration interaction theory (CISD, CISDT, CISDTQ), spin-unrestricted coupled-cluster theory (CCD, LCCD, CCSD, LCCSD, QCISD, CCSDT, CCSDTQ), and spin-unrestricted non-canonical many-body perturbation theory [MBPT(2), MBPT(3), MBPT(4)] and implement them into optimized parallel programs within NWChem. For many of these high-end correlation treatments, this constitutes the first reported automatic or parallel implementation, which implies that TCE can potentially change the way second-quantized many-electron theories are studied. Let us consider the example of CCSD, which has undergone perhaps the most thorough algorithmic refinement over the years. According to Stanton and coworkers [35], the number of arithmetic operations of their unrestricted CCSD program (the implementation in ACES II) is approximately $5/4\ O^2V^4 + 20\ O^3V^3 + 5/2\ O^4V^2$ in the leading order, where O and V are the number of α-spin occupied and virtual orbitals, respectively. The operation count of the TCE-synthesized CCSD program is approximately $5/4\ O^2V^4 + 45/2\ O^3V^3 + 25/2\ O^4V^2$. Remembering V » O in most applications and hence $O^2V^4 » O^3V^3 » O^4V^2$, the TCE-generated CCSD program can therefore compete with, albeit not outperform, the equivalent, carefully hand-coded program. A list of enhancements that are being considered for TCE's program generator includes storage space minimization via the loop fusion and space-time tradeoff techniques [36], and the development of a scalable parallel I/O interface for a global file system. TCE also offers an immediate avenue to automatically derive and implement existing and new models

of many-electron theory, which may include various relativistic many-electron theories, equation-of-motion coupled-cluster theory for excited states, and combined perturbation and coupled-cluster theories. With the adequate interface programs already in place, we believe that these and other models of many-electron theory can be most conveniently studied with NWChem and TCE.

7 Molecular Dynamics Simulations of Biological Membranes

The molecular dynamics (MD) simulation of biological membranes is generally more difficult to setup compared to simulations of other biological systems such as proteins, protein complexes and DNA that typically start from available three-dimensional structures from protein X-ray crystallography or NMR solution structure determination, and that typically consider only a single molecule or molecule complex in the model system. Biological membranes, however, derive their structural and functional properties from the fact that they are made up from a specific arrangement of many molecules for which little or no experimental data is available. Molecular modeling of such systems, therefore, relies on initial molecular assemblies that need to be constructed from primary molecular structures and application of computational relaxation methods.

The MD simulation preparation module of NWChem has a number of features to facilitate the construction of bio-molecular membranes. The typical sequence of steps consists of aligning of the lipid tails of the molecule that will form the membrane, placing copies of these aligned molecules on a two-dimensional grid perpendicular to the alignment axis, randomization of the orientation along the molecular alignment axis, 'collapsing' of the membrane by moving the molecules in the membrane plane to within a small, predefined distance from each other, and placement of counter-ions for lipid head groups that contain charged groups, or for lipopolysaccharides with charged saccharide moieties. Although several iterations may be required to tweak many of the parameters, this process can be executed in an automated way in NWChem.

Not only the setup, but also the equilibration of a membrane system is more complicated than many other bio-molecular systems. In order to obtain a molecular structure with the proper membrane integrity it is necessary to apply a number of restraints, especially during the equilibration phase of the simulation. NWChem allows for simulations with many restraining potentials that can be gradually turned on or off, for example to keep lipid tails of individual molecules perpendicular to the plane of the membrane, and to keep all head groups in the membrane plane. Since these membranes typically involve many identical molecules, specifying the details of these restraining potentials would be very tedious and prone to errors without the tools implemented in NWChem that automatically recognize identical molecules. These tools can be used to duplicate a single specified restraint potential or apply a single restraint potential to all other identical molecules in the system.

The equilibration is first performed on the constructed membrane in vacuo with restraining potentials applied to obtain and maintain membrane integrity. Once the molecules that form the model membrane are sufficiently dense, the system can be solvated. Special tools are available in NWChem to remove solvent molecules that are placed or diffuse into selected regions such as between the lipid tails. An important fea-

ture required for these simulations is that anisotropic pressure can be applied, since the compressibility of the membrane will be different in the plane compared to perpendicular to the plane of the membrane.

The membrane setup procedures can include the incorporation of other molecules, such as trans-membrane proteins. This procedure has been applied to an simulation of a lipopolysaccharide microbial membrane consisting of the lipopolysaccharide layer, a lipid layer, and an embedded trans-membrane protein [37].

8 Conclusions

In this paper, we have highlighted several of the key new functionalities in the computational chemistry code NWChem. These new functionalities range from tools to improve simulations of biological membranes to new orbital-dependent density functionals to the new projected augmented wave, electron transfer, and excitation modules to the tensor contraction engine for generating highly correlated methods. Future work in NWChem includes improvements to these capabilities as well as new functionality such as proton hopping in molecular dynamics simulations, an extended properties analysis module, excited state gradients and additional linear scaling methods. The authors encourage other developers to add functionality to NWChem. Further information concerning this may be obtained by sending an e-mail to nwchem-support@emsl.pnl.gov.

Acknowldgements. This research was performed in part using the Molecular Science Computing Facility (MSCF) in the William R. Wiley Environmental Laboratory at the Pacific Northwest National Laboratory (PNNL). The MSCF is funded by the Office of Biological and Environmental Research in the U.S. Department of Energy. The TCE work is funded by the U.S. Department of Energy, the Division of Basic Energy Science, Office of Science, as part of the SciDAC program. PNNL is operated by Battelle for the U.S. Department of Energy under contract DE-AC06-76RLO 1830.

References

1. Ostlund, N.S.: Int. J. Quant. Chem. Symp. **13** (1979) 15
2. Harrison, R.J., Shepard, R.: Annu. Rev. Phys. Chem. **45** (1994) 623
3. Kendall, R.A., Aprà, E., Bernholdt, D., Bylaska, E.J., Dupuis, M., Fann, G.I., Harrison, R.J., Ju, J., Nichols, J.A., Nieplocha, J., Straatsma, T.P., Windus, T.L., Wong, A.T.: Computer Phys. Comm. **128** (2000) 260.
   Bernholdt, D.E., Aprà, E., Fruchtl, H.A., Guest, M.F., Harrison, R.J., Kendall, R.A., Kutteh, R.A,. Long, X., Nicholas, J.B., Nichols, J.A., Taylor, H.L., Wong, A.T., Littlefield, R.J., Nieplocha, J.: Int. J. Quantum Chem. Symposium **29** (1995) 475
4. http://www.emsl.pnl.gov/pub/docs/nwchem
5. http://www.emsl.pnl.gov:2080/docs/parsoft/
6. Nieplocha, J., Harrison, R.J., Littlefield, R.J.: The Journal of Supercomputing, **10** (1996) 197

7. Nieplocha, J., Carpenter, C.: ARMCI: A Portable Remote Memory Copy Library for Distributed Array Libraries and Compiler Run-time Systems. Proc. 3rd Workshop on Runtime Systems for Parallel Programming (RTSPP) of International Parallel Processing Symposium IPPS/SPDP '99, Lecture Notes in Computer Science, Vol. 1586. Springer-Verlag, Berlin Heidelberg New York (1999)
8. Nieplocha, J., Foster, I, Kendall, R.: ChemIO: High-performance parallel I/O for computational chemistry applications, The International Journal of Supercomputer Applications and High Performance Computing, **12** (1998) 260–266
9. Message Passing Interface Forum. MPI: A Message-Passing Interface standard (version 1.1). Technical report, 1995. http://www.mpi-forum.org.
10. Perdew, J., Schmidt, K, Ed. Van Doren, V.E., Van Alsenog, K., Geerlings, P.: Density Functional Theory and its applications to Materials (AIP, Melville NY) (2001)
11. Boese, A. D., Handy, N.: J. Chem. Phys. **116** (2002) 9559
12. Neumann, R., Nobes, R., Handy, N.: Mol. Phys. **87** (1996) 1
13. Yang, W., Wu, Q.: Phys. Rev. Lett. **89** (2002) 143002
14. Kümmel, K., Perdew, J.: Phys. Rev. Lett. **90** (2003) 43004
15. Marcus, R. A., Sutin, N.: Biochimica Biophysica Acta **35** (1985) 437
16. Farazdel, A., Dupuis, M., Clementi, E., Aviram, A.: J. Am. Chem. Soc. **112** (1990) 4206
17. Rosso, K. M., Smith, D. M. A., Dupuis, M.: J. Chem. Phys. (accepted for publication)
18. Foresman, J.B., Head-Gordon, M., Pople, J.A., Frisch, M.J.: J. Phys. Chem. **96** (1992) 135
19. Jamorski, C., Casida, M.E., Salahub, D.R.: J. Chem. Phys. **104** (1996) 5134
20. Bauernschmitt, R., Ahlrichs, R.: Chem. Phys. Lett. **256** (1996) 454
21. Hirata, S., Head-Gordon, M.: Chem. Phys. Lett. **314** (1999) 291
22. Davidson, E.R.: J. Comput. Phys. **17** (1975) 87
23. Olsen, J., Jensen, H.J.Aa., Jørgensen, P.: J. Comput. Phys. **74** (1988) 265
24. Maurice, D., Head-Gordon, M.: J. Phys. Chem. **100** (1996) 6131
25. Windus, T., et al. (unpublished)
26. Casida, M.E., Jamorski, C., Casida, K.C., Salahub, D.R.: J. Chem. Phys. **108** (1998) 4439
27. Hirata, S., Head-Gordon, M.: Chem. Phys. Lett. **302** (1999) 375
28. Hirata, S., Head-Gordon, M., Szczepanski, J., Vala, M. (unpublished)
29. Blöchl, P.E.: Physical Review B, **50** (1994) 17953
30. Valiev, M., Bylaska, E.J., Gramada, A., Weare, J.H.: Ab Initio Molecular Dynamics Simulations using Density-Functional Theory, Reviews In Modern Quantum Chemistry: A Celebration Of The Contributions Of R. G. Parr, Ed. K.D. Sen (World Scientific, Singapore) (2002) 1684
31. Bylaska, E.J., Valiev, M., Kawai, R., Weare, J.H.: Computer Physics Communications, **143**, (2002) 11
32. Head-Gordon, M., Maslen, P.E., White, C.A.: J. Chem. Phys. **108** (1998) 616
33. Hirata, S. (unpublished)
34. Janssen, C.L., Schaefer III, H.F.: Theor. Chim. Acta **79** (1991) 1
35. Stanton, J.F., Gauss, J., Watts, J.D., Bartlett, R.J.: J. Chem. Phys. **94** (1991) 4334
36. Baumgartner, G., Bernholdt, D.E., Cociorva, D., Harrison, R., Hirata, S., Lam, C., Nooijen, M., Pitzer, R., Ramanujam, J., Sadayappan, P.: Proceedings of Supercomputing 2002 (2002)
37. Shroll, R.M., Straatsma, T.P.: Biopolymers **65** (2002) 395

Interfacing Computer Aided Parallelization and Performance Analysis

Gabriele Jost[1*], Haoqiang Jin[1], Jesus Labarta[2], and Judit Gimenez[2]

[1] NAS Division, NASA Ames Research Center, Moffett Field, CA 94035-1000 USA
{gjost,hjin}@nas.nasa.gov

[2] European Center for Parallelism of Barcelona-Technical University of Catalonia (CEPBA-UPC), cr. Jordi Girona 1-3, Modul D6,08034 – Barcelona, Spain
{jesus,judit}@cepba.upc.es

Abstract. When porting sequential applications to parallel computer architectures, the program developer will typically go through several cycles of source code optimization and performance analysis. We have started a project to develop an environment where the user can jointly navigate through program structure and performance data information in order to make efficient optimization decisions. In a prototype implementation we have interfaced the CAPO computer aided parallelization tool with the Paraver performance analysis tool. We describe both tools and their interface and give an example for how the interface helps within the program development cycle of a benchmark code.

1 Introduction

During the last decades a large amount of time and money has been spent on the development of large-scale scientific applications exposing an insatiable appetite for floating point operations per second. High performance parallel computer architectures have evolved to satisfy this demand. Often the structure of the existing codes can not fully exploit the parallelism provided by the hardware. Considering the enormous investment in these codes and the niche market status of scientific applications, there is a strong incentive not to re-implement the applications from scratch, but rather, employ a conversion process to produce versions of existing codes optimized for the current most powerful computer architecture.

In this paper we focus on shared memory parallel computer architectures which provide multiple processing units and a globally shared address space. Most compilers for shared memory parallel architectures support parallelization in the form of compiler directives such as the OpenMP standard [12]. The program developer has to insert the directives and specify the scope of the variables. The CAPO [6] parallelization support tool was developed at the NASA Ames Research Center to aid the devel

[*] The author is an employee of Computer Sciences Corporation.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 181–190, 2003.
© Springer-Verlag Berlin Heidelberg 2003

oper in this task. CAPO automates OpenMP directive insertion into existing Fortran codes and allows user interaction for an efficient placement.

When parallelizing an application, the developer cycles through OpenMP directives placement, performance analysis, and optimization. To support this process we have interfaced CAPO with a performance analysis tool. Paraver [13] was developed at CEPBA-UPC to enable the user to obtain a qualitative global perception of the application behavior as well as detailed quantitative analysis of program performance. Paraver allows the user to graphically inspect trace files collected during program execution.

The CAPO-Paraver interface is a step towards the development of a programming environment for scientific applications which allows the user to jointly navigate through program structure and performance data to optimize critical code segments while repetitive but tedious and error prone tasks are automated. This paper presents the basic motivation and the general idea of how the tools should be integrated. We describe our initial prototype implementation and give a demonstration of its potential use. The rest of the paper is structured as follows: In Sections 2 and 3 we describe CAPO and Paraver. We discuss the interface between the tools in Section 4 and give an example parallelization and optimization session in Section 5. In Section 6 we discuss related work and draw our conclusion in Section 7 where we also elaborate on our future plans.

2 The CAPO Parallelization Support Tool

CAPO was developed to automate the insertion of OpenMP directives with nominal user interaction. This is achieved by use of extensive interprocedural analysis from CAPTools [4], developed at the University of Greenwich, which provides a fully interprocedural and value-based dependence analysis engine. Details on the CAPO parallelization process can be found in [6]. CAPO performs the following steps to exploit loop level parallelism:

1. Identify parallel loops and parallel regions based on dependence analysis.
2. Merge parallel regions and use the NOWAIT clause on successive parallel loops if possible.
3. Place OpenMP directives including variable scope declarations and perform necessary code transformations.

Dependence analysis results from a CAPO session are stored in a data base. This contains the dependence graph and a list of unresolved questions that occurred during the analysis. Unresolved questions indicate conservatively defined dependencies, often due to unknown information about values of variables. The user may browse the list of questions and provide assertions to make a more precise dependence analysis possible.

The CAPO directives browser is designed to display information gathered during the parallelization such as the reasons for loops to be parallel or serial, and the relevant variables. For each subroutine the user can retrieve information about the loops it contains. The browser also provides user interfaces to declare certain variables as shared or private or to explicitly remove dependences.

3 The Paraver Visualization and Analysis System

The Paraver system allows performance analysis of task level, thread level and hybrid parallel programs. It has two major components: A tracing package and a graphical user interface to examine the traces, including an analysis module to calculate various statistics.

Accepting a variety of trace formats Paraver optionally provides its own tracing package, OMPItrace [11]. It allows for tracing of programs with multiple processes and multiple threads. Depending on the computer system, OMPItrace dynamically instruments certain runtime libraries. Examples of OpenMP related information dynamically instrumented and traced on our development platform (SGI Origin 3000) are:

- entry and exit of OpenMP runtime library routines,
- entry and exit of compiler generated routines containing the body of parallel loops,
- two hardware counters,
- the state of a thread (running, idle, synchronization, or in fork/join overhead).

User routines are not automatically traced on the SGI Origin, but OMPItrace provides library routines for source code instrumentation.

The trace collected during the execution of a program contains a wealth of information, which as a whole is overwhelming. The information must be filtered to gain visibility of a critical subset of the data. This can be done through timeline graphical displays or by histograms and statistics. Paraver provides flexibility in composing displays of trace data. The Paraver object model is structured in a three level hierarchy: Application, Task, and Thread. Each trace record is tagged with these three identifiers. The timelines can be displayed at each level. At the task level, the values for each thread are combined to produce a value for the task. The Paraver GUI allows to specify the computation of a given performance index from the records in the trace and then save it as a configuration file. These configuration files are used to display a view of the selected performance index.

4 The CAPO-Paraver Interface

An important step in the parallelization process is to determine whether the directives have been placed for efficient parallelization. When working with large applications the developer is confronted with the question of which code sections to focus on. We address this question by the use of trace information and an interface to a performance analysis tool. Although there are many aspects to performance optimization we will focus at this point on the parallelization aspect. The design of the interface between computer aided parallelization and performance analysis is summarized in Figure 1.

4.1 Selective Source Code Instrumentation

We have extended the source code transformation capability of CAPO to automatically insert calls to the OMPItrace library. The OMPItrace module does not automati-

cally trace entry and exit to user routines on our development platform. Even though the automatic tracing of user level routines is possible on some platforms, it may not always be desirable to trace all of the subroutines because of large instrumentation overhead. In our prototype we use the following simple heuristics. At this point we selectively instrument:

- routines that are not contained within a parallel region or a parallel loop,
- routines which contain at least one DO loop,
- outermost serial loops.

Parallelized loops and parallel regions are automatically traced by the OMPItrace package as described in Section 3. During the instrumentation process CAPO generates a file containing symbol information which will be used by Paraver to relate the performance trace data with the routine names. The instrumented, parallelized code can now be compiled and run to obtain a trace file which can be processed by Paraver.

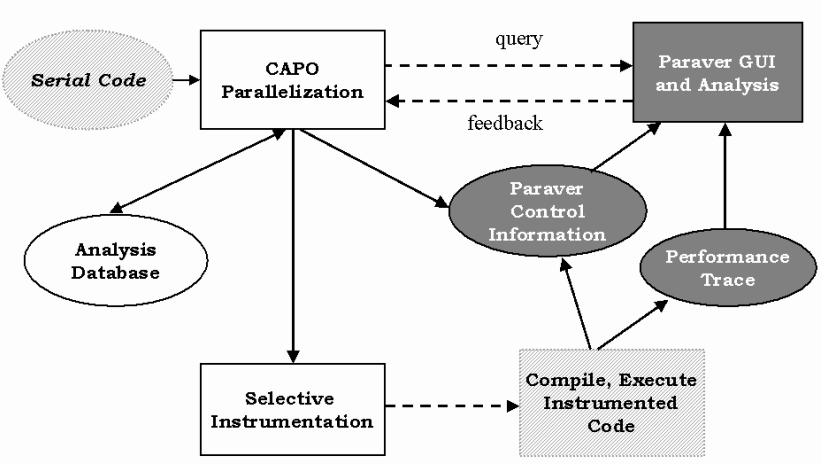


Fig. 1. The CAPO-Paraver interface. Solid lines indicate that data is automatically generated or read by the system components. Dashed lines indicate that user interaction is required. In our prototype querying the performance data is initiated by CAPO via Paraver configuration files. The outcome of the query is visually inspected by the user. Feedback to CAPO on what action to take is also provided by the user. The goal is to automate these steps

4.2 Display of Performance Metrics

We have designed a set of configuration files which show statistics related to the information displayed in the directives browser. The user will need to know:

- Where is the time spent?
- Is the time spent efficiently?

To address these questions the user may invoke Paraver from within CAPO. Paraver will load the previously generated trace file. The first view presents a diagram containing the percentage of time that each routine took. The time is presented on a task level, averaged over all threads. The exclusive time for a function call is reported

where exclusive time is the time for the routine minus the time for nested routines that are also instrumented. To give an indication of the quality of the parallelization we display the percentage of useful time for each thread. We define the useful time of a thread as time not spent in fork/join operations, synchronization or idle time. The Paraver analysis module calculates the percentage of useful time for each thread within the instrumented routines, as well as their mean and standard deviation. Each thread's average useful time should be high and the standard deviation low. CAPO automatically generates control information, in the form of configuration files and symbol information, which is then used by Paraver for calculating this performance metric and their display. Although the simple metrics described above are useful to identify routines with a bad parallelization ratio they do not conclusively point to the cause of the problem. A high standard deviation in useful thread time could for example also be due to an imbalanced workload among the loop iterations. The flexibility of the Paraver analysis module allows identification and addition of further metrics as our system evolves.

4.3 CAPO-Paraver Feedback Mechanism

CAPO supports querying the performance trace by generating control information and invoking Paraver. Our prototype requires the user to visually inspect the performance metrics calculated by Paraver and provide feedback to CAPO as indicated by the dashed lines in Figure 1. Our goal is to have CAPO query Paraver for performance metrics and retrieve the information without user interaction. We are working on a non-GUI based Paraver which will allow retrieving Paraver performance metrics in batch mode. With this, CAPO will be able to automatically query Paraver and correlate the outcome of the query with its loop analysis information. Our approach is to implement a rule based system where CAPO queries the performance trace, compares the metrics to given thresholds, relates the metrics to the program analysis information and then either performs certain transformations or requests further user input. An example for additional user input are assertions necessary to eliminate conservative dependences that prevent parallelization. Our prototype allows us to conduct experiments to gain experience on what the nature of the queries should be and how the outcome should affect the optimization process. The use of Paraver to provide profile data has the advantage that at any point the user can enter the process, inspect the metric requested by CAPO and carry out more analysis using the capability of the Paraver GUI and analysis module.

5 An Example Parallelization and Optimization Session

To demonstrate a parallelization session using the CAPO-Paraver interface we consider the BT benchmark from the NAS Parallel Benchmarks (NPB) [3]. We use the version as described in [5]. The BT benchmark solves three systems of equations resulting from an approximate factorization that decouples the x, y and z dimensions of the 3-dimensional Navier-Stokes equations. The development platform is an SGI

Origin 3000. To develop a parallel version of BT, the user starts a CAPO session loading the source code. Then CAPO performs the parallelization and generates the database. The user saves the CAPO generated source code containing OpenMP directives and the selective instrumentation as described in Section 4. Compiling and running the instrumented OpenMP code generates a performance trace file. In our demonstration the application runs on 4 threads. Paraver is invoked via a link provided by the CAPO user interface. Paraver loads the performance trace and a configuration file which calculates the percentage of time spent in each of the instrumented routines. The fragment of a snapshot is shown in Figure 2.

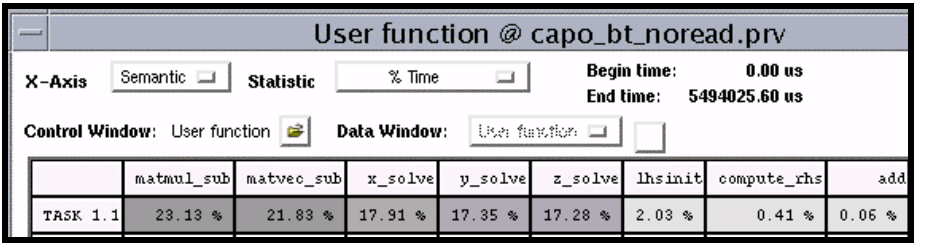


Fig. 2. Time distribution for the non-optimized parallel implementation of BT. Darker shading indicates a higher percentage of time

The useful time for each thread can be displayed by loading a second configuration file into Paraver, shown in Figure 3. The value of 0.96 for thread 1 in routine `z_solve` indicates that thread 1 was running user code 96% of the time. The value of 0.05 for thread 2 in routine `z_solve` indicates that the thread was running only 5% of the time. Routines `matmul_sub` and `matvec_sub` are obviously executed by only one of the 4 threads. The imbalance in useful time of the threads is an indicator for inefficient parallelization. All of these routines are major time consumers as seen in Figure 2.

Selecting routine `z_solve` in the directives browser and inspecting the parallelized loops reveals that the inner loops of the routine were chosen for parallelization (see Figure 4). Routines `matmul_sub` and `matvec_sub` are called from within routine `z_solve`, outside of the parallelized loop. For the outer loops, dependences had to be assumed. Inspecting the knowledge database (see Figure 4) shows that there are conservative dependences because the values of the entries in `gridpoints` are not known at compile time. These are the number of grid points for each dimension which are provided as input. They determine the loop lengths of the outer loops surrounding the parallelized loop. Since the user knows that the number of grid points is greater than 5 for each dimension, he can provide this information and repeat the analysis. The OpenMP directives are now placed on the outer loops of the solver routines. The new OpenMP code yields a much more balanced useful thread time for the time consuming routines as shown in Figure 5. The improved parallelization is reflected in a decrease of the overall runtime.

X-Axis: Semantic | Statistic: Average value | Begin time: 0.00 us | End time: 5494025.60 us

Control Window: User function | Data Window: useful

	compute_rhs	add	z_solve	y_solve	x_solve	lhsinit	matvec_sub	matmul_sub
THREAD 1.1.1	0.81	0.81	0.96	0.96	0.92	1	1	1
THREAD 1.1.2	0.82	0.84	0.05	0.05	0.05	0	0	0
THREAD 1.1.3	0.68	0.56	0.04	0.04	0.04	0	0	0
THREAD 1.1.4	0.66	0.57	0.04	0.04	0.04	0	0	0
Total	2.97	2.77	1.09	1.09	1.06	1	1	1
Average	0.74	0.69	0.27	0.27	0.26	0.25	0.25	0.25
Maximum	0.82	0.84	0.96	0.96	0.92	1	1	1
Minimum	0.66	0.56	0.04	0.04	0.04	0	0	0
Stdev	0.07	0.13	0.39	0.39	0.38	0.43	0.43	0.43

Fig. 3. View of the useful time per routine for each thread. Darker shading indicates a higher percentage of useful time

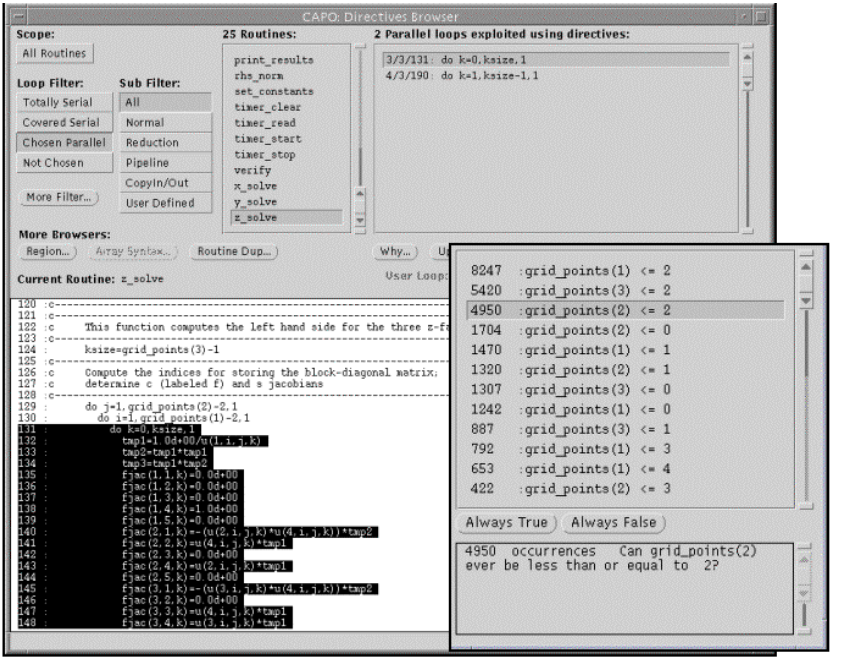


Fig. 4. The CAPO directives browser and knowledge database. The source code of the selected loop is highlighted

X-Axis Semantic Statistic Average value Begin time: 0.00 us End time: 312504.80 us

Control Window: User function Data Window: useful

	z_solve	y_solve	x_solve	compute_rhs	add
THREAD 1.1.1	0.96	0.98	0.96	0.85	0.86
THREAD 1.1.2	0.97	0.96	0.96	0.89	0.86
THREAD 1.1.3	0.64	0.64	0.64	0.64	0.58
THREAD 1.1.4	0.66	0.63	0.63	0.66	0.62
Total	3.21	3.21	3.20	3.05	2.92
Average	0.80	0.80	0.80	0.76	0.73
Maximum	0.97	0.98	0.96	0.89	0.86
Minimum	0.64	0.63	0.63	0.64	0.58
stdev	0.16	0.17	0.16	0.11	0.13

Fig. 5. Useful thread time after optimization. Darker shading indicates a higher percentage of useful time. The routines `matmul_sub`, `matvec_sub`, and `lhsinit` that appear in Figure 3 are not instrumented, since they are now enclosed by parallel loops within the solvers `x_solve`, `y_solve`, and `z_solve`. The remaining difference in useful thread time is caused by workload imbalance due to the loop length

We have chosen a small example because it allows us to focus on explaining the components of our system, rather than a particular application. The motivation for building our system, however, is based on the need for dealing with large application packages. We have used our prototype successfully on large scientific codes and will report on the results elsewhere [7].

6 Related Work

There are a number of commercial and research parallelizing compilers and tools for automatic parallelization and performance analysis tools that have been developed over the years. We can only name a few.

The SUIF Explorer [8] developed at Stanford University is an interactive parallelization tool based on the SUIF [14] compiler. It performs extensive static dependence analysis and also includes a set of dynamic analyzers to provide runtime information. Runtime information includes checking dependencies and time profiles for loops and routines. The user can provide assertions via a graphical user interface. The most notable difference to our approach is that by interfacing to a full performance analysis system like Paraver, we have more performance metrics available than just the time. This allows us a more flexible and detailed analysis and greater opportunity to detect performance problems. A commercial product for interactive parallelization is the ForgeExplorer [1]. It provides means to display dependence analysis information but does not allow extensive user interaction the way CAPO does.

An example of a commercial product for performance analysis is Vampir [16] which allows tracing and analysis of MPI codes. The CAPTools system provides an interface for generating VAMPIR traces and invoking VAMPIR from within CAP-

Tools for performance analysis of message passing applications. Two research projects on performance analysis are Paradyn [9], which is developed at the University of Madison, and Aksum, which is part of the ASKALON [2] project conducted at the University of Vienna. Both aim at the automatic detection of performance bottlenecks. A general performance model for OpenMP is proposed in [10]. Performance libraries based on this model have been developed for the TAU (Tuning and Analysis Utility) performance analysis framework ([10], [15]) and the EXPERT automatic event trace analyzer [17]. The advantage Paraver offers is a high level of flexibility in computing performance indices and statistics. This allows exploration of metrics of interest and their corresponding influence on the parallelization choices.

7 Conclusions and Future Plans

We have interfaced the CAPO computer aided parallelization tool with the Paraver performance analysis system to support the scientific programmer when parallelizing existing sequential codes. We have used the static analysis information available from CAPO to instrument the code. By employing the statistical analysis module from Paraver, performance metrics for critical parts of the code can be calculated and displayed. We have shown how the CAPO user interface displays loop analysis information and the performance statistics available from Paraver to point the user to parallelization problems.

As mentioned, we are working on the design of a non-graphical interface to the Paraver analysis module to be used by CAPO directly, without user interaction. Many common problems could be detected by CAPO automatically and correlated to the loop analysis information and the knowledge data base. A high standard deviation, for example, in the useful thread time is often an indicator that an inner loop within a loop nest has been parallelized. In many cases CAPO might be able to determine that the reason for this is a conservative dependence in the outer loop. If the reason is missing information for a variable value, CAPO can prompt the user for an assertion. Another opportunity is the automatic detection of workload imbalance within a parallel loop which can be improved by changing the scheduling of the loop iterations from static to dynamic. We also plan to address scalability issues by automatically comparing traces for different numbers of threads.

Not all performance problems in full-scale applications will be solved automatically. At this point it is still not clear where to draw the line between automated and user guided responsibilities. Experiments with our prototype will allow us to identify more tasks that can be automated. Our approach is to successively develop an environment where the repetitive, cumbersome and error-prone tasks are left to the tools, allowing the user to focus on the tasks that require ingenuity. This will help reduce the development time and increase the quality of the generated code.

Acknowledgements. The authors wish to thank the CAPTools team (C. Ierotheou, S. Johnson, P. Leggett, and others) at the University of Greenwich for their support on

CAPTools. This work was supported by NASA contract DTTS59-99-D-00437/A61812D with Computer Sciences Corporation/ AMTI, by the Spanish Ministry of Science and Technology, by the European Union FEDER program under contract TIC2001-0995-C02-01, and by the European Center for Parallelism of Barcelona (CEPBA).

References

1. Applied Parallel Research Inc., "ForgeExplorer," http://www.apri.com/.
2. ASKALON, http://www.par.univie.ac.at/project/askalon/.
3. D. Baily, T. Harris, W. Saphir, R. Van det Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0", RNR-95-020, NASA Ames Research Center, 1995.
4. C.S. Ierotheou, S.P. Johnson, M. Cross, and P. Leggett, "Computer Aided Parallelisation Tools (CAPTools) – Conceptual Overview and Performance on the Parallelisation of Structured Mesh Codes," *Parallel Computing*, 22 (1996) 163–195. http://captools.gre.ac.uk/
5. H. Jin, M. Frumkin, and J. Yan, "The OpenMP Implementations of NAS Parallel Benchmarks and Its Performance", NAS Technical Report NAS-99-011, 1999.
6. H. Jin, M. Frumkin and J. Yan. "Automatic Generation of OpenMP Directives and Its Application to Computational Fluid Dynamics Codes," *in Proceedings of Third International Symposium on High Performance Computing (ISHPC2000),* Tokyo, Japan, October 16–18, 2000.
7. G. Jost, H. Jin, "Computer Aided Optimization and Performance Analysis of Hybrid MPI/OpenMP Applications: A Case Study", NAS Technical Report, to appear.
8. Liao, S., Diwan, A., Bosch, R. P., Ghuloum, A., Lam, M., "*SUIF Explorer: An interactive and Interprocedural Parallelizer*", 7th ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming, Atlanta, Georgia, (1999), 37–48.
9. B.P. Miller, M.D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K.L. Karavanic, K. Kunchithhapdam and T. Newhall, "*The Paradyn Parallel Performance Measurement Tools",* IEEE Computer 28, 11, pp.37–47 (1995).
10. B. Mohr, A. D. Malony, S. Shende, and F. Wolf, "*Design and Prototype of a Performance Tool Interface for OpenMP*", Internal Report FZJ-ZAM-IB-2001-09, Research Centre Juelich, Germany, 2001.
11. OMPItrace User's Guide, https://www.cepba.upc.es/paraver/manual_i.htm
12. OpenMP Fortran/C Application Program Interface, http://www.openmp.org/.
13. Paraver, http://www.cepba.upc.es/paraver/
14. SUIF Compiler System, http://suif.stanford.edu/.
15. TAU: Tuning and Analysis Utilities, http://www.cs.uoregon.edu/research/paracomp/tau.
16. VAMPIR User's Guide, Pallas GmbH, http://www.pallas.de.
17. Wolf, F., Mohr, B., "Automatic Performance Analysis of SMP Cluster Applications", Tech. Rep. IB 2001-05, Research Centre Juelich, Germany, 2001.

Dynamic Performance Tuning of Distributed Programming Libraries[1]

Anna Morajko, Oleg Morajko, Josep Jorba, Tomàs Margalef, and Emilio Luque

Computer Science Department. Universitat Autònoma de Barcelona.
08193 Bellaterra (Spain)
ania@aows10.uab.es, olegm@aia.ptv.es,
{josep.jorba, tomas.margalef, emilio.luque}@uab.es

Abstract. The use of distributed programming libraries is very common in the development of scientific and engineering applications. These libraries, from message passing libraries to numerical libraries, are designed in a very general way to be useful for a wide range of applications. Therefore, there are several polices that must be adapted to the particular application, system and input data to provide the expected performance. Our objective is develop an environment for tuning the use of a distributed library on the fly according to the dynamic behavior of the applications. In this paper, we present as an example a tuning environment for PVM-based applications. We show potential bottlenecks when using PVM. We also include tuning scenarios that describe the evaluation of the application behavior and the solutions that can improve the performance.

1 Introduction

High performance computing is provoking a new evolution in many fields of science and engineering. Physicists, biologists or chemists have become developers and end-users of high performance applications running on powerful systems (parallel systems, clusters, or even Grid systems). However, the design and development of such applications from scratch is a complex task that requires a high degree of expertise and deep knowledge of the system and the programming capabilities.

To insulate the programmer from the low level details, several libraries ranging from communication and message passing to numerical methods and programming frameworks have been developed. These libraries offer a higher level of abstraction and facilitate the design and development of high performance applications. However, these libraries are developed in a general way to be useful for a wide range of applications. Therefore, to accomplish the performance expectations, a developer must tune the library use by choosing the best polices considering application requirements and the environment. Such tuning process requires a deep knowledge of the library that is not necessary for developing applications. Moreover, these adaptations do not depend only on the application features, but also on the input data

[1] This work has been supported by the MCyT (Spain) under contract TIC2001-2592, by the European Commission under contract IST- 2000-28077 (APART 2) and partially supported by the Generalitat de Catalunya – GRC 2001SGR-00218.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 191–200, 2003.
© Springer-Verlag Berlin Heidelberg 2003

or on the dynamically changing conditions of the application execution. Therefore, it is very hard to take into account all these variable conditions when developing applications. It is necessary then to tune the library usage on the fly during the application execution.

The goal is not to tune the library by modifying its source code, but to improve the way the library is used, e.g. switching a parameter in a library function call according to the variable environment conditions. This approach does not require analysis or tuning specific for an application, instead the optimization process is based on the features characteristic for the library.

The tuning is a complex and time-consuming task, and not feasible if it must be carried out manually by a developer. Therefore, it is beneficial to accomplish the performance expectations by using an automatic tuning environment. Such an environment is based on an expert knowledge of the possible bottlenecks and library limitations. It automatically inserts the required instrumentation to determine the current application behavior, decides what parameters must be modified and inserts the modifications dynamically to improve the performance of the whole application.

This approach is powerful because we focus on tuning the use of the library. Investigating the library it is possible to find its potential drawbacks. For each problem then the set of specific information can be determined, such as what should be measured to determine the behavior, which should be the desired behavior and which parameter should be modified to improve the application performance. The developers can concentrate on designing and developing an application, and submit it with the input data. After this the dynamic tuning environment takes care of controlling the application execution to ensure a good performance.

In the following sections of this paper, the dynamic tuning approach is presented. In Section 2, we introduce concepts and design of the MATE environment that supports dynamic tuning of parallel applications. Section 3 describes the aspects of PVM library considered as bottlenecks and presents their example optimizations. Section 4 shows practical experiments with dynamic tuning that we have conducted using MATE environment. Finally, Section 5 summarizes and concludes this work.

2 MATE

The main goal of dynamic tuning is to improve the application performance on the fly. This approach is suitable for applications with dynamic conditions, like variable behavior depending on the input data or variable behavior during the application execution. Running the application under control of a dynamic tuning system allows for adapting its behavior to the existing conditions. We propose a novel Monitoring, Analysis and Tuning Environment (called MATE) that provides dynamic automatic tuning of parallel applications. Our presentation of MATE focuses on its use with the PVM library [1].

MATE performs dynamic tuning in three basic and continuous phases: monitoring, performance analysis and modifications. This environment dynamically and automatically instruments and traces a running application to gather information about the application behavior. The analysis phase searches for bottlenecks, detects their causes and gives solutions on how to overcome them. Finally, the application is

dynamically tuned by applying given solution. Moreover, the application does not need to be re-compiled, re-linked and restarted.

2.1 Basic Concepts

To achieve the objective set out by our work and allow for the application modifications on the fly, it is necessary to use dynamic instrumentation [2]. The basic idea is to defer program instrumentation until it is in execution and insert, modify and delete this instrumentation dynamically during run-time. MATE utilizes a special library called DynInst [3], [4] for two main purposes:

- Monitoring – to instrument an application during its execution. It is possible to add/remove a code that collects information about the application behavior.
- Tuning – to modify an application behavior. It is possible to change the code of the running application to improve its performance.

Since a parallel application consists of many intercommunicating processes physically executed on different machines, we distribute the modules of MATE to machines where the processes are running. To improve performance of the whole application, we gather information about all processes at a central location.

The tuning methods must be kept simple. The changes cannot affect the correct functioning of the application. We cannot assume that an application variable can be modified without taking any precautions. These factors limit the application of dynamic tuning, hence it is desired to provide external specifications of the programs behavior [5]. In MATE we assume the following terms: measure points, performance model, tuning points/actions. A measure point is a location in a process where the instrumentation must be inserted to provide measurements. A performance model consists of activating conditions (conditions in the application behavior considered to be a bottleneck) and/or formulas that allow for finding the optimal conditions. Tuning points are the elements that may be changed to improve application performance. Tuning action represents the action to be performed on a tuning point.

2.2 Design

As it is shown in Figure 1, MATE consists of three main components that cooperate among themselves controlling the execution of the application: a set of distributed Monitor modules (Monitors), a global Analyzer module (Analyzer), and a set of distributed Tuner modules (Tuners).

2.3 Monitor

The Monitors are distributed to all machines where a parallel application is running. They control the processes and collect the events produced during the execution. Each Monitor instance is responsible for a single machine. As PVM-based applications can create new processes and add new machines during execution, the Monitor integrates two special PVM services: the “tasker service” responsible for new process creation

and the “hoster service” that controls machine management. The Monitor is able then to control virtual machine changes (add/remove host), as well as application processes (add/remove process). This information is always up-to-date and any changes are sent to the Analyzer module.

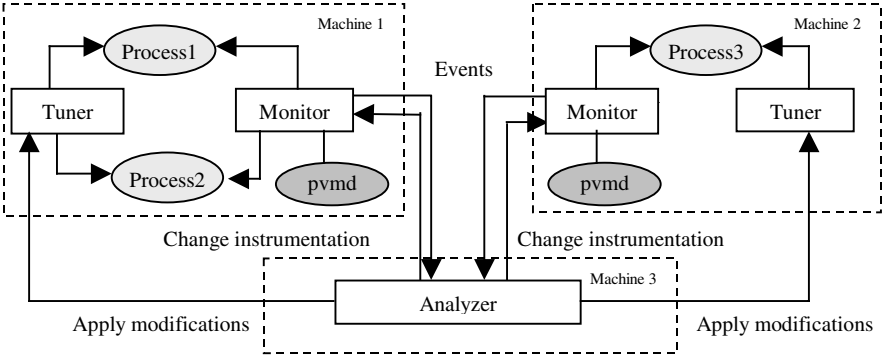


Fig. 1. Basic design of MATE in PVM environment.

To collect events, the Monitor dynamically inserts instrumentation into the original program execution at points needed to detect performance problems. Instrumentation generates events containing information about what happened, when and where. For example, if communication is monitored, the instrumentation can be inserted at the entry and/or exit of `pvm_send()` and `pvm_recv()` functions and each event should contain: timestamp, event type, source and target process identifiers, size of message. Generated events are sent to the Analyzer. This event flow can introduce high level of intrusion into the network and hence in some cases we aggregate the selected events. The Monitor is able to add or remove instrumentation dynamically. This capability minimizes the intrusion and permits to provide more precise information about the application behavior.

2.4 Analyzer

The Analyzer module is responsible for the automatic performance analysis of a parallel application “on the fly”. The Analyzer uses the knowledge (measure points, performance model and tuning points/actions) of possible problems and their solutions. All necessary information is specific for PVM problems and determined by the investigation of the library implementation. To be effective, the analysis is kept simple and decisions are taken in a short time.

When the Analyzer has been started, it receives from the Monitors information about the configuration of a virtual machine. This module is informed about all the changes in PVM virtual machine. The Analyzer chooses initial measure points and broadcasts them to all Monitors. When an application to be tuned starts, the Analyzer enters in a bottleneck search phase. It continuously receives requested events generated by different processes. By examining the set of events, the Analyzer extracts measurements. Then, it evaluates a performance model to determine the actual and optimal performance. When the Analyzer decides that the actual performance can be improved, it sends a request to the appropriate instance of Tuner,

determining what should be changed, and where. For example, when the Analyzer determines that in a process a particular PVM function should be invoked with a specific parameter value, the name of the function, together with a new parameter value, is sent to the Tuner.

Obviously, during the analysis, Monitor modules are collecting and providing new data to the Analyzer. The Analyzer may need more information about program execution to determine the causes of a particular problem. It can therefore request the Monitor to change the instrumentation dynamically. The Analyzer also informs a user about detected problems and undertaken actions.

2.5 Tuner

The Tuner modules automatically change the application execution by inserting modifications into the running processes. The Tuners manipulate the process image in memory by means of DynInst library, hence they do not need to access a source code or restart the application. Similarly to the Tracers, the Tuners must have access to all application processes. Therefore, the Tuner modules are distributed among all the machines where the application is running. The Tuner module waits for requests from the Analyzer. The request specifies a target process, a tuning point and a tuning action. When the Analyzer detects a problem and its solution is found, a tuning request is sent to an appropriate Tuner instance. The Tuner receives it and applies given solution dynamically to a specified process or processes on the machine where it is running. This module contains a set of predefined modifications that can be activated by the Analyzer. We consider the following tuning actions, among others:

- One time function invocation – calling a specified function in the application.
- Function parameter changes – the value of an input parameter is modified before a function body is executed.
- Function invocation – an additional function call is inserted into the application at a specified point.
- Function replacement – all calls to a given function are replaced with a call to another function.

3 Tuning Examples

In this section, we present some of the PVM tuning examples that we studied within MATE. All examples illustrate aspects of the PVM library that can significantly improve the application performance. First, we investigate the library searching for its characteristics that can cause significant bottlenecks in an application performance. Then, for each potential bottleneck, we describe an optimization scenario, namely: what should be measured to detect the problem (set of measure points), how to discover the problem (performance model and activating conditions), and a solution as to overcome the problem (tuning point/action).

In the following subsections we present potential bottlenecks and their optimization scenarios when using PVM library: communication mode, data encoding mode, and message fragment size. To demonstrate communication performance of the

PVM environment and indicate its problems, we used a master-worker benchmark program that exchanges messages of various sizes. The presented measurements were repeated thousands of times, and the average round-trip time and its standard deviation (i.e. error) were calculated. Experiments were conducted in the environment consisting of a cluster of homogenous workstations Sun UltraSPARC II, 440 MHz connected by 100Mb/sec network.

3.1 PVM Communication Mode

PVM tasks have two modes to establish communication with other tasks, the task-to-task mode (direct) and task-to-daemon-to-daemon-to-task mode (indirect). By default, the indirect mode is used so all the messages are routed through the daemons. The direct mode is available only on some architectures. In this mode the tasks bypass the PVM daemon and have a direct link to each other using a separate socket. Although the initial TCP set up time is larger, all subsequent communication between the same two tasks is usually faster. The primary drawback of this method is that each TCP socket consumes one file descriptor, so there is a limit on maximum number of opened connections. This mode is less scalable, but it is a faster transfer method.

In Table 1 we compare the measurements of PVM communication performance in both communication modes. We can observe that the change from indirect to direct mode results in significantly faster communication (up to 50%). We must also point out that in majority of typical environments used to run PVM applications (workstation clusters) this mode is available, but rarely used.

Table 1. Benefits gathered from changing communication mode in a round trip application.

MsgSize [B]	Indirect Time [ms]	Direct Time [ms]	Difference [ms]	Average Benefit %
1	1,08 (±0,04)	0,53 (±0,02)	0,55 (±0,06)	50,72%
10	1,09 (±0,04)	0,54 (±0,01)	0,55 (±0,05)	50,39%
100	1,15 (±0,04)	0,61 (±0,02)	0,53 (±0,07)	46,47%
1000	1,77 (±0,05)	1,18 (±0,03)	0,59 (±0,08)	33,45%
10000	10,66 (±0,23)	8,51 (±0,21)	2,15 (±0,44)	20,13%
100000	104,95 (±4,11)	84,17 (±3,91)	20,78 (±8,02)	19,80%
1000000	1059,64 (±30,08)	861,37 (±39,07)	198,27 (±69,16)	18,71%

The application can configure the mode explicitly, but by default the indirect mode is used. During the execution, we can detect the use of indirect mode by calling `pvm_getopt(PvmRoute)` and check conditions to use direct mode. This mode is available when the environment does not include shared-memory machines and the number of PVM tasks is smaller than system-dependent limit. The tuning action includes one-time function invocation `pvm_setopt(PvmRoute, PvmRouteDirect)` that activates the mode. To avoid reentrancy problems in PVM library implementation, the invocation must be synchronized with the application execution. Therefore, first the breakpoint is inserted at the entry of function `pvm_send()` and when it activates the actual invocation is performed.

3.2 Data Encoding Mode

When PVM transfers the data, it converts the data format transparently between machines that have different architectures. By default PVM encodes data using XDR (external data representation) standard, because it cannot know if the user is going to add a heterogeneous machine before this message is sent. If there is no heterogeneous machine, the next message will only be sent to a machine that understands the native format. The encoding phase therefore, can be skipped, what allows for avoiding data encoding costs and reducing execution time.

In Table 2 we can observe the benefits from data raw encoding mode in comparison to default XDR encoding obtained with our benchmark application (up to ~74%). We can see that encoding overhead grows together with message size. For message sizes less than 1KB, the difference is low. If bigger amount of data is sent, more time is required to encode/decode it. We conclude that the data raw mode is significantly faster and preferable in the typical homogeneous clusters.

Table 2. Benefits gathered from changing data encoding mode in a round trip application.

MsgSize [B]	XDR Time [ms]	Data Raw Time [ms]	Difference [ms]	Average Benefit %
1	0,53 (±0,02)	0,51 (±0,01)	0,01 (±0,03)	2,25%
10	0,53 (±0,02)	0,52 (±0,01)	0,01 (±0,03)	2,25%
100	0,61 (±0,02)	0,58 (±0,01)	0,03 (±0,03)	4,76%
1000	1,17 (±0,02)	0,98 (±0,01)	0,19 (±0,04)	16,38%
10000	8,53 (±0,30)	3,28 (±0,05)	5,24 (±0,34)	61,49%
100000	84,69 (±4,02)	23,36 (±1,20)	61,33 (±5,22)	72,42%
1000000	873,90 (±40,35)	227,37 (±19,81)	646,52 (±60,16)	73,98%

The application can configure the data encoding mode explicitly, but by default the XDR mode is used. MATE controls the addition of hosts to the virtual machine (tasker/hoster services) so it is able to detect if an application is executed in the homogeneous cluster. During the execution we can decide to use data raw mode if the condition of hosts' homogeneity is fulfilled. The tuning action includes insertion of instrumentation (entry of `pvm_initsend()`) that changes the encoding mode from XDR to data raw (parameter mode set to `PvmDataRaw`). Moreover, when a new machine with different architecture is about to be added, the Analyzer module can request to restore the XDR mode.

3.3 PVM Message Fragment Size

In PVM messages exchanged by the tasks are composed without a maximum length. This is accomplished internally by dividing bigger messages into fixed-size blocks called fragments. PVM uses a default fragment size of 4KB (implementation limits it to 1024KB). When sending large messages, a number of fragments must be allocated and then separately sent. This causes high fragmentation what may reduce performance. Therefore, by changing message fragment size, bandwidth can be increased significantly. When the fragment size increases, PVM dynamically allocates more memory while sending/ receiving messages, hence more data is sent/received per system call. The drawback to this strategy is increased memory usage. It must be also pointed out that fragment size changes do not give significant effects in indirect communication mode, because they do not affect the behavior of PVM daemons.

The results of experiments are presented in Table 3. All experiments were conducted using direct communication mode. The default fragment size proved to be the optimal choice for small message sizes (less than 4KB). However, we can observe that for bigger messages the benefits from using larger fragment sizes are significant (up to 55%). This example demonstrates that the optimal fragment size depends on the application behavior – size of data that is sent and received. Moreover, the optimal value can vary during the execution (due to program phases) and hence it is not enough to calculate it once, but it should rather be adapted to the application behavior.

Table 3. Benefits gathered by changing the message fragment size in a round trip application.

FragSize	4 KB	16 KB		64 KB		256 KB		512 KB	
MsgSize	Time [ms]	Time [ms]	Benefits [%]	Time [ms]	Benefits [%]	Time [ms]	Benefits [%]	Time [ms]	Benefits [%]
2 KB	1,77	1,75	1,11	1,77	-0,22	1,76	0,34	1,79	-1,45
4 KB	4,54	2,51	44,68	2,51	44,68	2,51	44,80	2,50	44,88
8 KB	7,99	4,13	48,37	4,14	48,16	4,25	46,85	4,12	48,45
64 KB	55,64	34,11	38,70	29,47	47,03	27,08	51,33	26,85	51,74
256 KB	226,45	133,08	41,23	113,31	49,96	107,99	52,31	104,93	53,66
512 KB	460,66	264,40	42,60	224,41	51,28	214,75	53,38	212,99	53,76
1024 KB	920,62	525,61	42,91	443,56	51,82	426,94	53,62	425,02	53,83
4096 KB	3730,53	2105,77	43,55	1739,84	53,36	1666,26	55,33	1657,24	55,58

The application can configure the fragment size explicitly, otherwise the default value is used. During the execution, we can query the actual size by calling `pvm_getopt(PvmFragSize)` and detect if the application changes this value by instrumenting the function `pvm_setopt(PvmFragSize, size)`. Currently, to calculate the optimal fragment size, we use the experimentally deduced formula:

OptimalFragSize = Average (message size) + Std deviation (message size)

The Analyzer module requires the number and sizes of transmitted messages. Therefore, the communication calls (e.g., `pvm_send()`, `pvm_recv()`, `pvm_mcast()`) must be instrumented to gather statistics. The analysis of collected data is performed periodically and separately for each task. However, the tuning action is not applied each time the new optimal value is calculated. Instead, the tuning action is triggered when the difference between current and optimal values exceeds a fixed threshold, and the estimated communication cost becomes significant during the period. The tuning action includes one-time function invocation `pvm_setopt(PvmFragSize, OptimalFragSize)` that changes the current fragment size. The invocation must also be synchronized as described in Section 3.1.

4 Tuning Experiments

This section presents the results of practical experiments with dynamic tuning that we conducted using MATE. We provide timings that characterize the performance obtained executing parallel application with our environment MATE. We evaluate dynamic tuning costs, i.e. the overhead incurred by MATE in comparison to traditional PVM implementation. The total intrusion time includes application startup (process load time, process image parsing time), instrumentation (adding, removing,

execution and reporting), analysis and tuning. Experiments show performance improvements originated from dynamic tuning when applying different scenarios. Each test was performed hundreds of times and the average of the wall clock execution times of the master process was taken. We show that tuning is effective and benefits are higher than the overhead introduced into the application execution.

4.1 NAS Parallel Benchmark – IS Kernel (Sort Application)

To conduct our experiments, we selected a computationally-intensive parallel program. We used Integer Sort (IS) kernel benchmark from NAS Parallel Benchmark suite [6]. The IS kernel ranks a large array of small integers as fast as possible using a bucket sort method. Communication costs are high (up to about 50% communication) in this application and are dominated by all to all data exchange messages, wherein each processor sends to all others this data which falls in the range of the recipient.

Table 4. Execution time of IS Kernel Benchmark in different tuning scenarios.

No.	Tuning scenario	Execution time [sec]	Tuning benefits [sec]	Intrusion [sec]
1.	PVM (no tuning)	732	-	-
2.	PVM + communication mode tuning	604	127 (17,5%)	21 (~3,5%)
3.	PVM + data encoding mode tuning	761	-29 (-3,9%)	21 (~2,8%)
4.	PVM + message fragment size tuning	769	-37 (-5,1%)	27 (~3,5%)
5	PVM + all scenarios	529	202 (27,7%)	28 (~5,3%)

Table 4 presents the results of the IS Kernel Benchmark experiments in different tuning scenarios. In the first scenario the application was executed under standard PVM 3.4 without any tuning and it was used as a reference result. The other tests were performed under PVM, but MATE monitored and tuned the application.

In the second scenario, the PVM communication mode was tuned by MATE. The MATE analyzer decided to use the direct mode, as by default the application used indirect mode and our experiments were conducted in a small NOW environment. We can observe 17,5% benefit in execution time caused by this tuning action. Such improvement can be explained by high computation-communication ratio (1:1). The measured intrusion did not exceed 3,5% of the total execution time.

In the third scenario, tuning the data encoding mode was tried. The analyzer did not perform any tuning actions. Originally the application used data raw encoding mode, so no improvement was possible. However, when the application was experimentally executed in XDR mode, the improvement reached 47% (as the exchanged data was integers). The intrusion resulted in 2,8% of the execution time.

In the forth scenario, MATE conducted message fragment size tuning. In this scenario the analyzer requested the instrumentation of send and receive primitives in all of the tasks to examine number and sizes of messages. The analysis indicated that the default fragment size was improper, because each of the tasks was sending a series of very small messages (4B and 16B) as well as big messages (over 1MB). The requested tuning action increased the fragment size. After the change, the application remained stable and no more tuning was performed. However, we cannot observe any benefits, because by default indirect mode was used. The benefits are significant when conducting the same experiment in direct mode. The intrusion reached 3,5%.

Finally, in the fifth scenario, we conducted all described tuning scenarios in the same execution. Both communication mode tuning and fragment size tuning was

applied successfully. Despite of the intrusion (about 5,3% resulted from more inserted instrumentation and higher volume of collected measurements), the introduced changes produced the best results, improving the total execution time up to 27,7%.

5 Conclusions and Future Work

Dynamic automatic performance tuning of distributed libraries appears as a powerful technique to accomplish a performance improvements of applications with a dynamic behavior. A complete dynamic automatic performance tuning environment (MATE) has been presented. MATE includes the monitoring, analysis and modifications of the application on the fly without stopping, recompiling or rerunning the application. In this paper we have focused on the tuning of PVM-based application. We have shown a set of problems that can appear as bottlenecks when using PVM library. We have proven that MATE is promising and is able to improve the performance of the application implemented in PVM by tuning the library usage. Obviously, environment causes intrusion and it is inserted into the application execution, but it is still smaller than the benefits obtained from the performed improvements.

The presented methodology is general and can be applied to improve the use of other libraries. Moreover, the set of conducted experiments gave us new ideas on future work and tuning examples. Many applications highly use memory. Therefore, it would be profitable to optimize memory usage by providing custom allocators as general-purpose allocators may be inefficient or inflexible. The tuning of communication protocol options correlated with message exchange can be beneficial as well. We also found interesting the tuning of work size unit or data distribution as it can reduce significantly the execution time.

References

1. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V. *"PVM: Parallel Virtual Machine, A User´s Guide and Tutorial for Network Parallel Computing"*. MIT Press, Cambridge, MA, 1994.
2. Buck, B., Hollingsworth, J.K. *"An API for Runtime Code Patching"*. University of Maryland, Computer Science Department, Journal of High Performance Computing Applications. 2000.
3. Hollingsworth, J.K., Buck, B. *"DyninstAPI Programmer's Guide. Release 3.0"*. University of Maryland, January 2002.
4. Paradyn Project *"Paradyn Parallel Performance Tools, User's Guide, Release 3.3"*. University of Wisconsin, Computer Science Department, January 2002.
5. César, E., Morajko, A., Margalef, T., Sorribes, J., Luque, E. "*Dynamic Performance Tuning Environment Supported by Program Specification*". Scientific Programming, 10, pp. 35–44. 2002.
6. Bailey, D.H., Harris, T., Saphir, W., Wijngaart, R., Woo, A., Yarrow, M. *"The NAS Parallel Benchmarks 2.0"*, Report NAS-95-020, December, 1995.

Parallel Program Debugging with MAD – A Practical Approach

Dieter Kranzlmüller[1] and Axel Rimnac[2]

[1] GUP, Dept. for Graphics and Parallel Processing
[2] Institute for Theoretical Physics
Joh. Kepler University Linz
Altenbergerstr. 69, A-4040 Linz/Austria, Europe
{dieter.kranzlmueller,axel.rimnac}@jku.at

Abstract. Debugging parallel programs can be very time-consuming and tedious, because the multiplicity of communicating processes increases the complexity of a program and the probability of incorrect behavior. Solutions are provided by debugging tools, which try to offer meaningful ways to investigate errors and their original causes. The MAD environment is a debugging toolset, which focuses on parallel and distributed programs. This paper discusses the application of MAD to real-world programs with practicability and usability as the main goals. The observations are based on a series of debugging sessions conducted for a specific application of theoretical physics, with the opinions of the tool developer on one side and the view of the application developer on the other side.

1 Introduction

Computational Science and Engineering (CSE) is strongly related to High Performance Computing (HPC), because HPC promises the performance needed by many CSE applications. The power of HPC architectures is usually provided by parallel and distributed systems, which represent a serious challenge for CSE application developers. The traditional software engineering process has to be adapted to cope with several concurrently executing and communicating processes, which attempt to jointly solve a given problem. This increases the complexity of parallel and distributed CSE applications and requires dedicated support from software development tools.

A basic necessity for software is correctness, which is verified during the testing and debugging phase of the software lifecycle. The program's behavior during runtime is inspected and compared to the given specification. When deviations from the specified behavior are observed, the developer tries to locate the original cause for the anomaly and eventually corrects the program's code. These activities are supported by so-called debugging tools, which provide techniques such as breakpointing and inspection for program comprehension [12].

Although many parallel debuggers exist, the majority of error detection work is still performed by placing "`print`" statements in the program and inspecting their output during execution. This seems almost archaic compared to the functionality provided by sophisticated debugging tools, such as DDBG [4], P2D2 [7], DETOP [13], and Totalview [5].

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 201–210, 2003.
© Springer-Verlag Berlin Heidelberg 2003

One reason for this situation may be that placing `print` statements is rather trivial compared to the learning curve of using a new debugging tool. Another reason may be that the textual representations chosen by the majority of debugging tools may be inappropriate to express the complexity of parallel and distributed programs [11]. Finally, there may be a discrepancy between the views of the tool developer and the application programmer on what is important about parallel debugging, or rather how and what the debugging tool should support.

This paper describes a case study of parallel program debugging for a particular computational science program. The idea was to investigate the usefulness of an existing debugging tool when developing a theoretical physics program. The approach was rather informal, with the two persons involved sitting together and detecting errors. Consequently, the following article contains an example set of errors detected during the course of debugging. Nevertheless, the findings of this cooperation seem to be an interesting point for a discussion about the practical usefulness of software development tools.

The paper is organized as follows. The next section describes the case study, an application from theoretical physics, and the MAD environment. Section 3 introduces the required preparations for error detection and presents simple automatic debugging features, which revealed several errors in the program. Section 4 shows some more advanced debugging features by using array visualization, while a summary and an outlook on future work conclude this paper.

2 Case Study: Debugging with the MAD Environment

2.1 Target Application

The application chosen as a target for analyzing the debugging process under "real world" conditions is a program that is currently used in the investigation of the scattering of single helium (^{4}He) particles on a macroscopic slab system consisting of ^{4}He particles near absolute zero temperature. Helium near absolute zero is known to be the only terrestrical liquid that shows purely quantum mechanical effects. Hence, the description of scattering has to be done in the framework given by the well-known Schrödinger equation. Therefore, the quantity containing maximum information on the system is the quantum mechanical wave function of the given many-body system.

The calculations are done in the so-called Correlated Basis Function (CBF) approximation which describes the energy loss of an incident particle to the target slab in terms of two-phonon decay processes, i.e., an incident particle may stick in the system by losing its energy to two quasiparticle excitations that can be viewed as surface waves on the system. The incident particle may also lose just part of its energy and leave the slab either as a reflected or transmitted particle[1].

The equation for the quantities of major interest, namely the probabilities of elastic reflection and transmission as well as the probability of inelastic scattering takes the form [2]

[1] A classical analog would be the process of throwing a small stone into a lake. At this process the stone may be reflected elastically or stick in the liquid and sink.

$$H_1(z)\psi(z) + \int \Sigma(z,z',\omega)\psi(z')dz' = \hbar\omega \int S(z,z')\psi(z')dz' \qquad (1)$$

which is a generalized one-dimensional Schrödinger equation. Here, $\psi(z)$ is the quantum mechanical wave function of the incident particle. $H_1(z)$ and $S(z,z')$ depend on the structure of the static background liquid, ω is the energy of the incident particle and $\Sigma(z,z',\omega)$ is the so-called nonlocal optical potential accounting for the possibility of the incident particle to lose energy. The self energy matrix $\Sigma(z,z',\omega)$ is given by

$$\Sigma(z,z',\omega) = \sum_{m,n} \int \frac{V_{mn}(p,q,z)V_{mn}(p,q,z')}{\hbar(\omega_m(q) + \omega_n(p) - \omega)} dp\, dq \qquad (2)$$

where the matrix elements $V_{mn}(p,q,z)$ are known ground state quantities and the denominator displays the dispersion relations of so-called Feynman excitation [3]. In determining Σ, parallelization is done by data-decomposition along indices m, n and summing up all contributions at the end of the calculation. This is by far the most time-consuming part of the computation.

Practical calculations have revealed that the execution time for a single run is about 12 hrs on 8 processors of an SGI Origin 3800 (equipped with R12000 processors at 450 MHz). This clearly justifies the use of a HPC system and parallel software engineering for application development. The program is written in FORTRAN 90 and uses the Message Passing Interface standard MPI [10] for communication and synchronization. The following sections describe the development of the application described above from a debugging point of view.

2.2 The MAD Environment

The **M**onitoring **A**nd **D**ebugging environment **MAD** [8] is a toolset for debugging parallel programs based on the message-passing paradigm. The principal idea of MAD is to display observed program interactions as space-time diagrams and to apply this graphical representation for error detection. Based on this idea, two kinds of errors can be distinguished [9]:

- *Local errors* represent incorrect program behavior without influence of parallelism. They occur on a single process and do not involve communication or interaction with other processes. These kinds of errors can be investigated with traditional sequential debuggers.
- *Global errors* are established due to the parallel nature of the application. They are connected to operations which would not exist in sequential programs. As a consequence, the occurrence of incorrect behavior and its original source are often located on different processes. Due to these characteristics, the usability of sequential debuggers is questionable for tracking global errors.

For the remainder of this paper, we will focus on global errors, because their existence is determined by the parallelism constructs introduced when developing programs for an HPC system. Nevertheless, it is obvious that debugging local errors may also be needed, for example when tracing back sources of global errors.

The debugging process of MAD is performed using a series of independent modules. The two most important modules of MAD are

- the monitoring tool **NOPE** - **NO**ndeterministic **P**rogram **E**valuator, and
- the visualization tool **ATEMPT** - **A T**ool for **E**vent **M**ani**P**ula**T**ion.

While these core modules sufficiently cover the majority of debugging activities, several other analysis tools extend the functionality of MAD. Examples include trace file converters for various formats, additional visualization modules, and a control- and data flow analysis tool [8].

3 Program Instrumentation and Automatic Error Detection

The initial step for program debugging is instrumentation, which is applied to insert code containing monitoring functionality into the target program. The monitoring code is then responsible for extracting program state data for subsequent analysis activities. In MAD, monitoring is performed with NOPE, which is inserted into the target program through source code instrumentation. While source code instrumentation is less sophisticated than other techniques, e.g. dynamic instrumentation [6], its simplicity is a major advantage. The drawback with this method is that the source code has to be recompiled in order to integrate the monitoring functionality. However, this is probably acceptable since it does not require more effort than inserting `print` statements into the program.

With the monitoring tool NOPE, instrumentation is performed in two ways:

(1) *Automatic instrumentation* is used for all functions known a priori. This includes the library function calls of MPI as well as some system functions. The monitoring code is inserted by replacing the include file `mpif.h` with its counterpart of NOPE - `monitor.h` - which replaces the original function calls through preprocessor directives.
(2) *Manual instrumentation* is required whenever behavior needs to be observed, which is specific to the target application. This includes switching on and off the monitoring functionality as well as monitoring of user-defined data structures. In this case, the user has to insert monitoring functionality in the code in the same way as `print` statements are used.

For the target application, we applied both kinds of instrumentation. While automatic instrumentation was turned on for the complete debugging session, manual instrumentation required several iterations to extract the state information of interest.

The most important characteristic of instrumentation seems to be simplicity. A debugging tool will only be utilized, if instrumentation and monitoring is less complicated than placing `print` statements.

After recompiling the program with the monitoring functionality of NOPE included, another program execution is initiated. The state data extracted by NOPE is stored in tracefiles, which can afterwards be analyzed with the visualization tool ATEMPT. Instead of exhaustively discussing the features of MAD, the following paragraphs describe a few situations, that occurred during the debugging sessions of the target application.

Situation 1: Early in the debugging session, the program hangs for no apparent reason. The diagnosis `print` statements indicate that the execution of the program is blocked at some point, but does not reveal the exact location in the source code nor the actual reason thereof.

Among the reasons why software hangs, the most typical case is a deadlock due to an unfulfilled condition at a blocking operation. An example from MPI is the blocking receive `MPI_Recv`, which holds a process' execution until a specific message arrives. If the message does not arrive - for whatever reason - the process' execution is blocked forever. This kind of error is a so-called *isolated receive* [9].

The problem with this kind of situations is that blocked executions and program failures are difficult to observe. The reason is based on the fact, that the monitoring tool needs to store the monitoring data to disk. Yet, if the operation of a process is blocked, the important information may still be in some intermediate monitoring or file buffer. If the program's execution is terminated, the data from the file buffer is lost.

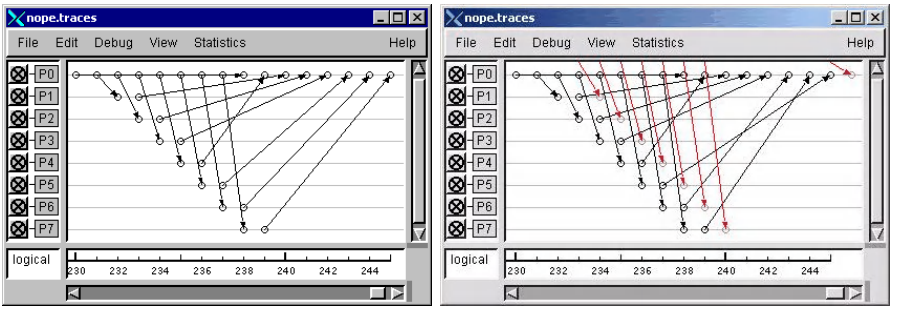


Fig. 1. Missing trace data due to program failure.

An example of this phenomenon is shown in Figure 1. The left screenshot displays the program's execution as observed during execution with the ATEMPT visualization tool. The program hangs due to a blocking receive on process P0, which is actually not shown in Figure 1. Instead, it seems as if the program terminated without any reason.

With NOPE, tracefile flushing can manually be turned on or off at any place during the program's execution. After turning tracefile flushing on and restarting the program, the right screenshot is observed. The program still hangs for the reason mentioned above, but the cause of the hanging is clearly visible on process P0. The applied correction was to correct the iteration counter for the blocking receive.

Please note, that the visualization of an error provides only a first indication of the incorrect behavior. A pre-requisite for correcting the error is identification of the corresponding line of source code. The appropriate feature in MAD is source code connection, which is provided when clicking on the visual representation of an event. A new window opens, displaying event attributes and the filename and line number of the responsible statement.

A different solution to this problem is described in [14]. To avoid the problem of file buffering, shared memory buffers and an external reader process are provided. This way,

the interesting information can be stored in the tracefiles by a second process, which is not affected by the blocking operation. However, while the benefits of this approach are obvious, its implementation is more difficult than turning on trace file flushing.

In addition to isolated receives, ATEMPT is also able to automatically detect other anomalies in communication patterns. This includes isolated send events or events with different message length as in the following example:

> **Situation 2:** The results of the application contained incorrect values if the number of elements in a particular vector was not divisible by an integer.

In this case, the automatic detection of ATEMPT revealed the incorrect communication pair even before the incorrect results have been verified. Figure 2 presents an example of a communication pair, where the amount of data provided by the sender process P1 does not fit into the message buffer at the receiver P0. Following the receive on P0, the process is missing some data and possibly accessing incorrect message buffers.

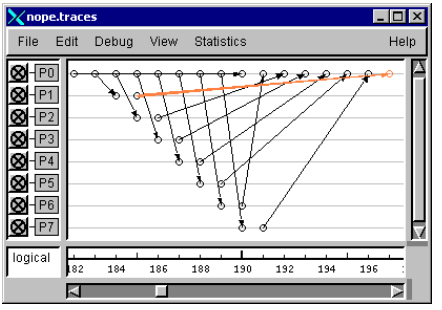


Fig. 2. Different message length at sender and receiver.

The automatic detection of communication errors as described above revealed most MPI related errors in the program with a few executions and correction cycles. We believe, that the benefits of these simple techniques are most usable during debugging.

4 Advanced Debugging Techniques – Array Visualization

Besides errors in communication statements, parallelization of a program often involves semantic changes to the program's behavior, including partitioning and distribution of data. In this context, a special feature of MAD for debugging is array visualization (similar functionality is provided by the Prism debugger [1]). With array visualization is it possible to graphically display the contents of up to three dimensions of an arbitrary array. The array itself may be distributed across the available processes using High Performance Fortran-like distribution styles (block, cyclic,...).

Using array visualization requires manual instrumentation to specify the structure of the array local to each process. The monitor NOPE provides dedicated function calls for array visualization. However, due to the number of possibilities for distributing an

array across multiple processes, these function calls may represent a major obstacle for the in-experienced debugging user.

The most simple approach of instrumenting a program for array visualization is achieved by placing the function call

```
monARRAYTRACE(void *array, trcArrayInfo info)
```

The variable `array` is a pointer to the start address of the local array, while `info` describes the structure of the local array and its relation to the parts on the other processes. This description includes the variable type of the elements, the number of dimensions and the size of the array in each dimension. The distribution of the array onto the processes is expressed in HPF-like descriptions. Further details can be specified with overlapping areas at process borders.

An example of an array visualization event is shown in the top-left screenshot of Figure 3. After a broadcast from process P0, every process provides its corresponding part of the global matrix as indicated by the line across all processes. By clicking on the array visualization event, an additional window opens and displays the reconstructed global array as shown with the examples in Figure 3.

Situation 3: The array visualization feature has been used to display a 3D-array during processing within the target application. Each process provides only the local array, which is then assembled into the global array. The values stored in the array are displayed as a heat-diagram, with different values indicated by different colors.

With the help of the array visualization feature, distribution errors can often be spotted immediately. While the parallel program should usually ensure a smooth gradient across process borders, errors in array limits or processing are often observed.

The top-right screenshot of Figure 3 shows the 3D-array as computed by the application. For illustration and to emphasize the incorrect behavior, we changed the source code to store a simple function into the parts of the array distributed across all processes. As shown in the middle-left screenshot, the computation of the distributed array seems to be wrong. Stripes along dimension x indicate, that the user did not specify the correct array dimensions on each process. This is underlined by the distribution map of the array as shown in the middle-right screenshot. The global array is distributed across the 8 processes blockwise along dimension z.

The bottom-left screenshot of Figure 3 displays the global array after correcting the array boundaries on each process. The disturbance at the process borders has been removed and the computation seems correct. In order to further investigate the 3D-array, mechanisms for modifying the array visualization are provided. An example is shown in the bottom-right screenshot, where a slice of the array along dimension z has been removed in order to display the inside of the matrix.

The array visualization proved a valuable tool during the course of debugging. While the instrumentation of arrays is certainly too difficult for the in-experienced users, the graphical display of distributed data structures represents a huge benefit. Achieving the same knowledge with `print` statements is certainly a much tougher task.

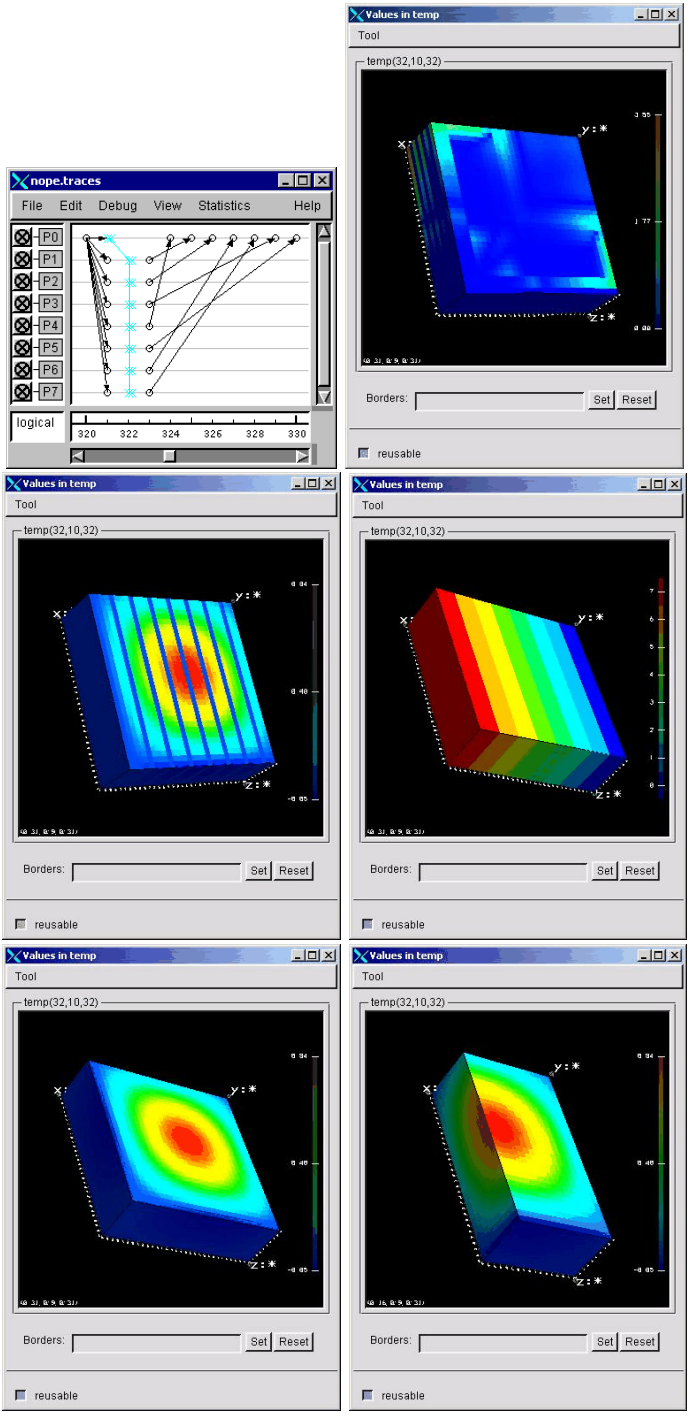


Fig. 3. Examples of array visualization.

5 Conclusions and Future Work

Debugging parallel and distributed CSE applications is a difficult task, which requires dedicated support from software tools. By applying the MAD environment for a real-world application, several discrepancies in viewpoints from the tool developer to the application programmer occurred. While tool developers seem to be interested in solving the most complex problems of program debugging, application programmers seem to be dissatisfied with the functionality of available debugging tools. In fact, the common practice of using `print` statements for debugging seems to arise from the simplicity of the approach.

On the positive side of our experience with real-world applications, several features of MAD proved vital during the debugging process. The automatic analysis features revealed a series of minor bugs in the code, which were easily traced back based on the space-time diagram and its integrated source code connection. Among the advanced analysis techniques, the verification of data distribution routines with array visualization provided substantial benefits, even though the instrumentation was sometimes tricky.

While debugging the theoretical physics application is now closed, several improvements of the MAD environment are being planned based on the insights provided by this work. On one hand, the automatic analysis activities will be enhanced by pattern analysis, which is expected to reveal repeated communication patterns or minor deviations thereof. On the other hand, the array visualization work will be extended to include additional distributed data structures and their visualization, as well as to simplify the instrumentation process.

Acknowledgments. This work has been supported by a series of colleagues at the Johannes Kepler University, including Prof. Jens Volkert and Prof. Eckhard Krotschek. We gratefully acknowledge J. Messner from the Zentraler Informatikdienst (ZID) of the Johannes Kepler Universität Linz for providing part of his thesis to the MAD project and for assistance in the course of the debugging procedure described in this paper.

References

1. Allen, D., Bowker, R., Jourdenais, K., Simons, J., Sistare, S., and Title, R.: "The Prism Programming Environment", Proc. Supercomputer Debugging Workshop 91, Albuquerque, New Mexico, USA, pp. 1–7 (November 1991).
2. Campbell, C.E., Krotscheck, E., and Saarela, M.: "Quantum Sticking, Scattering, and Transmission of ^{4}He atoms from Superfluid ^{4}He Surfaces" Physical Review Letters Vol. 80, No. 10, pp. 2169–2172 (1998)
3. Clements, B.E., Krotscheck, E., and Tymczak, C.J.: "Multiphonon excitations in boson quantum films" Physical Review B Vol. 53, No. 18, pp. 12253–12275 (1996)
4. Cunha, J.C., Lourenco, J.M., Antao, T.: "An Experiment in Tool Integration: the DDBG Parallel and Distributed Debugger", EUROMICRO Journal of Systems Architecture, Vol. 45, No. 11, pp. 897–907 (1999).
5. Etnus, LLC: "TotalView Debugger", `http://www.etnus.com/Products/TotalView` (January 2003)

6. Hollingsworth, J.K, Miller, B.P., Cargille, J.: "Dynamic Program Instrumentation for Scalable Performance Tools", Proc. SHPCC, 1994 Scalable High Performance Computing Conference, Knoxville, TN, USA, pp. 841–850 (May 1994).
7. Hood, R.: "The p2d2 Project: Building a Portable Distributed Debugger", Proc. SPDT'96, ACM SIGMETRICS Symposium on Parallel and Distributed Tools, Philadelphia, USA, pp. 127–136 (May 1996).
8. Kranzlmüller, D., Grabner, S., Volkert, J.: "Debugging with the MAD Environment", Parallel Computing, Vol. 23, No. 1-2, pp. 199–217 (Apr. 1997)
9. Kranzlmüller, D.: "Event Graph Analysis for Debugging Massively Parallel Programs", PhD Thesis, GUP, Joh. Kepler University Linz, http://www.gup.uni-linz.ac.at/~dk/thesis (Sept. 2000).
10. Message Passing Interface Forum: "MPI: A Message-Passing Interface Standard - Version 1.1", `http://www.mcs.anl.gov/mpi/` (June 1995).
11. Pancake, C.M.: "Visualization Techniques for Parallel Debugging and Performance-Tuning Tools", in: Zomaya, A.Y., " Parallel Computing: Paradigms and Applications", Intl. Thomson Computer Press, pp. 376–393 (1996).
12. Rosenberg, J.B.: "How Debuggers Work: Algorithms, Data Structures, and Architecture", John Wiley & Sons, New York (1996).
13. Wismüller, R., Oberhuber, M., Krammer, J., and Hansen, O.: "Interactive Debugging and Performance Analysis of Massively Parallel Applications", Parallel Computing, Vol. 22, No. 3, pp. 415–442 (March 1996).
14. Wismüller, R., Dozsa, G., and Drotos, D.: "Using OMIS for On-line Monitoring in the GRADE Programming Environment", Proc. DAPSYS 98, Austrian-Hungarian Workshop on Distributed and Parallel Systems, Budapest, Hungary, pp. 177–184 (Sept. 1998).

Enhancing Prediction Accuracy in PCM-Based File Prefetch by Constained Pattern Replacement Algorithm

Inchul Choi and Chanik Park

Department of Computer Science and Engineering/PRIL
Pohang University of Science and Technology
Kyungbuk, Republic of Korea
{sharpic, cipark }@postech.ac.kr

Abstract. In modern file systems, I/O latency is still major bottleneck of performance and predictive file prefetching is one of promising approaches that can enhance I/O performance of file system. To utilize predictive file prefetching to file system, there should be a file access pattern prediction model that can predict future file access. Partitioned Context Model(PCM) [2] is known as one of the most accurate file access pattern prediction models[3]. In order to predict longer sequence, the order of PCM must be increased. However, the prediction accuracy of PCM decreases when PCM is in high order. Careful analysis reveals that the pattern replacement algorithm in the PCM is the major cause in decay of the prediction accuracy. The pattern replacement algorithm destroys file access patterns without successful training of newly occurred file access patterns. We proposed the constrained pattern replacement algorithm to overcome this adverse effect by revising replacement condition. The simulation results using the DFSTrace system trace[13] show that the proposed algorithm improves prediction accuracy without any extra cost by 3.5% compared to traditional pattern replacement algorithm of PCM(about 40% of the accuracy bound of 7%).

1 Introduction

Despite impressive advances in file system throughput resulting from technologies such as high-bandwidth networks and disk arrays, file system latency has not improved enough.In many cases, it has become worse because of the speed gaps to follow the modern processor. File system I/O remains one of the major bottlenecks against the operating system performance. Predictive file prefetching is one of promising approaches to reduce file system latency. For predictive file prefetching to be effective, however, access prediction should be accurate and far enough in advance to effectively mask out the prefetch latency.

Partitioned Context Model(PCM) is known as one of the most accurate file access pattern prediction model[3] and it can be used for sequence prediction by using a high order context model. We discovered that when we increase the

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 213–222, 2003.
© Springer-Verlag Berlin Heidelberg 2003

order of the model for the prediction of file sequences, the prediction accuracy of PCM decreases. To find the cause of this effect, we analysed the file access pattern learning algorithm of PCM. The pattern replacement algorithm of PCM is found to be the one of major cause of decay of prediction accuracy in a high order PCM. The pattern replacement algorithm, which we call the unconstrained pattern replacement algorithm, of PCM can destroy the file access pattern in a partition without successful training of the newly occurred file access pattern, and it is the cause of the loss of high order file access pattern.

We propose a new replacement algorithm to overcome this adverse effect of the unconstrained pattern replacement algorithm. In the proposed replacement algorithm, when the full partition is met, the test proceeds after decaying of patterns in the partition whether the unconstrained pattern replacement algorithm deletes the same higher order nodes, especially the parent node of the newly arrived file access pattern. If true, decayed patterns in the partition is restored to its previous state to keep the state information. In this situation, the training of new pattern rely on the low order context in the other partitions. This algorithm ensures the conservation of the state information of the full partition by preventing useless loss of patterns in the full partition. Useless loss of patterns in the full partition is a major cause of decay of the prediction accuracy in high order PCM. So the proposed constrained pattern replacement algorithm enhances the high order prediction accuracy of PCM. The result of the simulation using the DFSTrace system shows that the proposed algorithm improves prediction accuracy without any extra cost by about 3.5% (about 40% of the accuracy bound of 7% between the FMOCM and the PCM), even if we increase the order to 74% of the partition size for file sequence prediction.

The remainder of this article is organized as follows. Section 2 briefly describes file access pattern prediction models. In section 3 we analysed the pattern learning algorithm of PCM and present the problem of replacement algorithm. Then we propose the new pattern replacement algorithm to enhance high order prediction accuracy of PCM. Section 4 presents our simulations and results. In section 5 we conclude the paper.

2 Related Work

In this section we present a brief background on the file access pattern prediction models and detail more about Partitioned Context Modeling.

2.1 File Access Pattern Prediction Model

File accesses follow previous patterns with a high probability. Last-successor prediction model is very simple prediction model which predicts that an access to file A will be followed by the same file that followed the last access to A, but from trace-based simulation[3] it correctly predicted 72% of access events. Another prediction model, the graph-based model is more complicated than last-successor model. The graph-based model keeps frequency counts for

all files accessed within a window following an access to each file. However, the graph-based model sees extremely poor prediction accuracy, 40% on average as compared to the last-successor's accuracy 72%. More accurate prediction model is Finite Multi-Order Context Model(FMOCM) which uses text compression technique to track file access patterns. The FMOCM has about 81% prediction accuracy and it reduced the difference between last-successor and an ideal predictor by greater than on third. But, this high prediction accuracy is achieved at the cost of significantly greater state space overhead. So, improved FMOCM which called a Partitioned Context Model(PCM) is proposed. This model addresses the state space overhead problem of FMOCM, while achieving comparable prediction accuracy with FMOCM. Next section, we detailed about Partitioned Context Modeling with Context Modeling which PCM is based on.

2.2 Partitioned Context Modeling (PCM)

First, we describe about context modeling. A context model(CM) is one that uses preceding events to model the next event. For example, in the string 'object' the character 't' is said to occur within the context 'objec'. The length of a context is termed its *order*. Techniques that predict using multiple contexts of varying orders(e.g.'ec', 'jec', 'bjec') are termed Multi-Order Context Models. To prevent the model from quickly growing beyond available resources, most implementations of a multi-order context model limit the highest order modeled to some finite number m, hence the term *Finite Multi-Order Context Model*.

A context model uses a trie, a data structure based on a tree, to efficiently store sequences of symbols. Each node in this trie contains symbol(e.g. a letter from the alphabet, or the name of a specific file). By listing the symbols contained on the path from the root to any individual node, each node represents an observed pattern.

The state space for this model is proportional to the number of nodes in this tree, which is bounded by $O(n^m)$, where m is the highest order tracked and n is number of unique files. On a normal file system the number of files can range between 10 thousand and 100 million, such space requirements are unreasonable.

To address the space requirements of FMOCM, the Partitioned Context Model is proposed. This model divides the trie into partitions, where each partition consists of a first order node and all of its descendants. The number of nodes in each partitions is limited to a static number that is a parameter of the model. The effect of these changes is to reduce the model space requirements from $O(n^m)$ to $O(n)$.

3 Constrained Pattern Replacement Algorithm in PCM

3.1 Pitfalls of Traditional Pattern Replacement Algorithm

FMOCM and PCM have basically the same pattern learning algorithm. However, PCM has a static partition size limit, so it can't store all of the patterns

it has learned previously. To learn newly occurred file access pattern while preserving static partition size, PCM destroy patterns in a partition that is less probable to occur. Namely, PCM replace less probable patterns with newly occurred patterns and so pattern replacement algorithm is learning algorithm of PCM. The PCM replacement algorithm is as follows. When a new node is needed in a full partition, all node counts in the partition are divided by two (integer division), any nodes with a count of zero are cleared to make space for new nodes. If no space becomes available, the access is ignored. The pattern replacement algorithm of PCM is very simple and has no constraint when it destroys the patterns in a partition. From now we call this simple replacement algorithm as Unconstrained Pattern Replacement Algorithm. When PCM is at low order, Unconstrained Pattern Replacement Algorithm causes the model to adapt faster to new access patterns.As a result, prediction accuracy can be slightly increased. But, when PCM is in high order, the prediction accuracy of PCM decreases as the order increases. The cause of this problem reside in the difference between PCM and FMOCM, and the only difference between FMOCM and PCM is in pattern learning algorithm, the Unconstrained Pattern Replacement Algorithm. The Unconstrained Pattern Replacement Algorithm is based on the decaying of patterns in the partition and it has no constraint condition for pattern destroying.

As the order of PCM increases, the number of pattern decaying increases also because the partition size is limited to a static number. An increase in the number of pattern decaying can result in lower average node count. When the average node count in partition is low, the Unconstrained Pattern Replacement Algorithm can cause problems. Figure 1 describes the trie structre trained by sequence 'ABCABEF' and situation that the new access to 'G' is occured. The Unconstrained Pattern Replacement Algorithm can cause problems from the situation like in Figure 1.

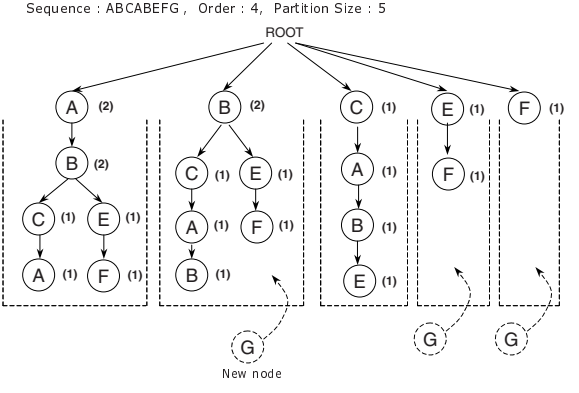


Fig. 1. Example trie before Unconstrained Pattern Replacement for the sequence ABCABEFG

In Figure 1, the new node 'G' which represents the pattern 'BEFG' is about to added to second partition, but the partition is full, so pattern decaying (integer division by 2) occurs and zero count nodes are cleared. Figure 2 shows situation after pattern decaying in the second partition. After Unconstrained Pattern Replacement in the second partition, the number of nodes are severely dropped and the parent node of new node 'G' is cleared out. To store the pattern 'BEFG' by adding new node 'G', nodes represents 'BEF' should not be cleared out. However, when we use the Unconstrained Pattern Replacement Algorithm, this cannot be achieved. As a result, patterns are lost without learning the newly occurred pattern. Situations similar to Figure 1 are more often met when the order of model is high, because high order means low average node count in the partitions. So, if we overcome this adverse effect, we can expect increase of prediction accuracy in high order PCM.

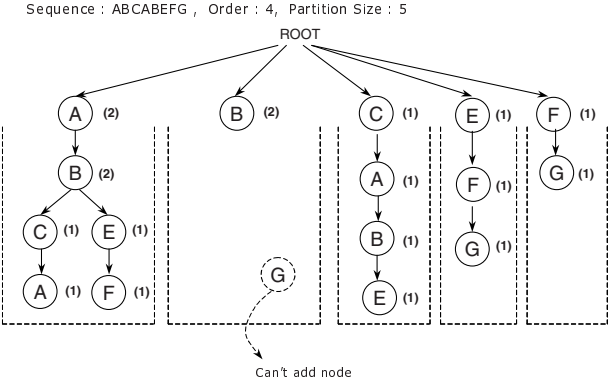


Fig. 2. Example trie after Unconstrained Pattern Replacement for the sequence ABCABEFG

3.2 Constrained Pattern Replacement Algorithm

To overcome the adverse effect of Unconstrained Pattern Replacement Algorithm of PCM in high order, we propose a new replacement algorithm which has constraint condition for pattern decaying. We call this new replacement algorithm as Constrained Pattern Replacement Algorithm. The Constrained Pattern Replacement Algorithm check whether the parent node of new arrived file access pattern is deleted after decaying of patterns. And if this is true, restore the partition to its previous state.

Pseudo-code of Constrained Pattern Replacement Algorithm

```
begin
{
   If ( Is Partition Size Full ? )
   {
      Divide count of all nodes in trie by 2;
      Clear any nodes with a count of zero;

      If ( Is parent node of new node exist ? )
      {
            If ( Is Partition Size Full ? )
                break;
            else
                Add new node;
      }else{
            Restore cleared nodes and node counts;
            break;
      }

   }else{
      Add new node;
   }

}until ( number of new nodes to add == 0 )
```

The Constrained Pattern Replacement Algorithm can prevent the useless decaying of patterns in partition without learning of newly occurred pattern. For example, if we use Constrained Pattern Replacement Algorithm in situation of Figure 1, we decay patterns in the partition and have a situation like Figure 2. Then we check whether the parent node of newly arrived pattern is cleared out and find out it is true in second partition. So, patterns in the second partition is restored to their previous state. Figure 3 shows the trie structure after Constrained Pattern Replacement Algorithm is applied to Figure 1.

From Figure 3, we can know that the Constrained Pattern Replacement Algorithm keeps the state information of second partition to prevent the useless decaying of patterns.

4 Simulation Results

To compare the prediction accuracy of PCM with our Improved PCM(PCM with Constrained Pattern Replacement Algorithm), we used trace data collected with the DFSTrace system. For simulation we selected the workload of barber machine which represents the characteristics of file server workload. The machine barber was also a server with the highest rate of system calls per second. For comparison

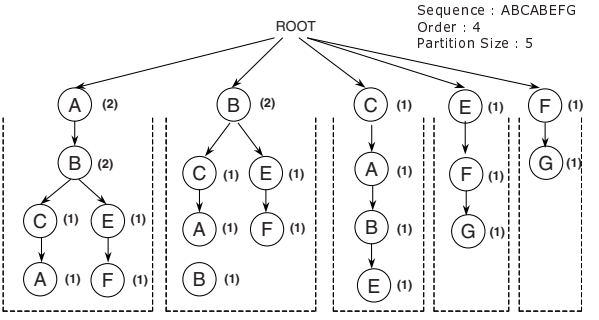


Fig. 3. Example trie after Constrained Pattern Replacement for the sequence ABCABEFG

purposes, we define measure *Useful Prediction*: U_p and *Prediction Accuracy*: P_a as follows.

$$U_p = \frac{Successful\ \ prediction\ \ count}{Total\ \ file\ \ access\ \ count} \tag{1}$$

$$P_a = \frac{Successful\ \ prediction\ \ count}{Total\ \ prediction\ \ count} \tag{2}$$

U_p represents the usefulness of prediction and P_a represents the prediction efficiency. From above definition we compare the PCM with Improved PCM by difference of U_p and P_a. Definition of D_U and D_P are as follows.

$$D_U = U_{p(ImprovedPCM)} - U_{p(PCM)} \tag{3}$$

$$D_P = P_{a(ImprovedPCM)} - P_{a(PCM)} \tag{4}$$

We simulated PCM and Improved PCM with Partition Size 32,48 and predict most probable file from the longest context. To comparison purpose, we measure D_U, D_p with increasing order. In Figure 4 D_U and D_p is positive in high order and has maximum value 3.5% which means Improved PCM has 3.5% better U_p and P_a. The result of partition size 48 in Figure 5 is also similar. Considering the maximum difference of U_p between FMOCM and PCM is 7%, improvement in prediction accuracy is corresponds to approximately 40%.

Figure 6 and Figure 7 compare U_p between traditional pattern replacement algorithm and constrained pattern replacement algorithm. As we can see from Figure 6 and Figure 7, the Constrained Replacement Algorithm improved prediction accuracy of PCM in high order.

5 Conclusion

In this paper we proposed a new replacement algorithm to enhance the prediction accuracy of PCM in high order. We analysed pattern learning algorithm

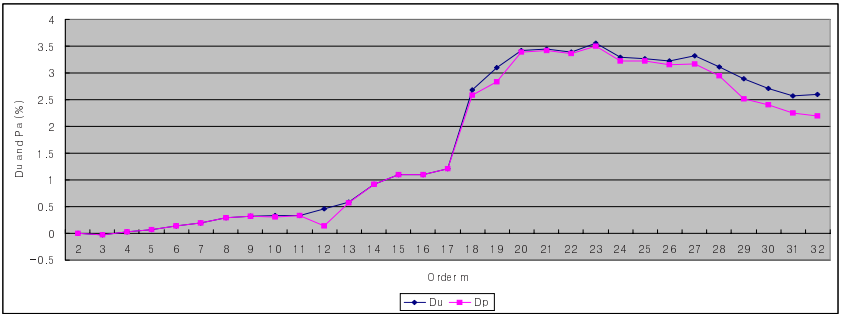


Fig. 4. D_U and D_p when Partition Size is 32

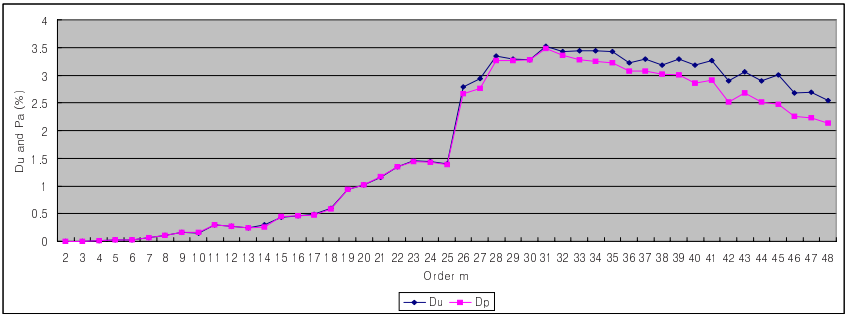


Fig. 5. D_U and D_p when Partition Size is 48

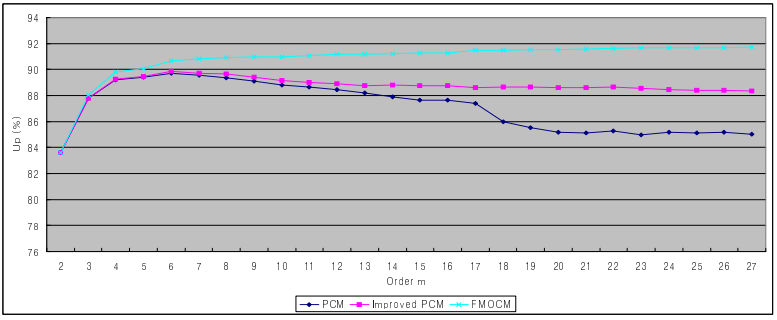


Fig. 6. Comparison of Useful Prediction between traditional pattern replacement algorithm and constrained pattern replacement algorithm (Useful Prediction vs. order of PCM when *partition size* is 32)

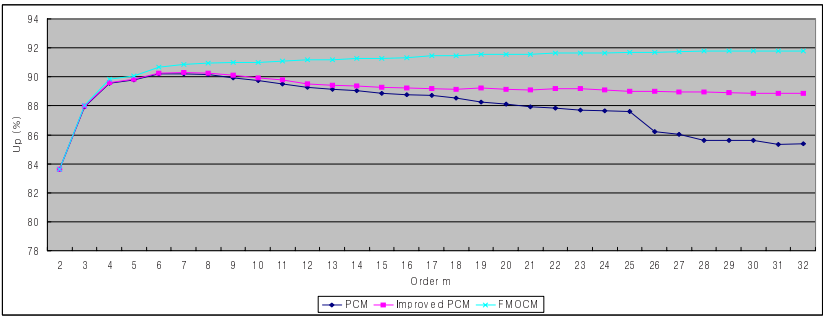


Fig. 7. Comparison of Useful Prediction between traditional pattern replacement algorithm and constrained pattern replacement algorithm (Useful Prediction vs. order of PCM when *partitioin size* is 48)

of PCM and discovered the cause of decrease of prediction accuracy in high order. We also proposed Constrained Pattern Replacement Algorithm to prevent useless loss of patterns in partition. Consequently, we enhanced the prediction accuracy of PCM in high order. From the trace-based simulation results, using DFSTrace system trace, we can see Improved PCM which uses Constrained Pattern Replacement Algorithm has a better prediction performance than PCM. Improved PCM has maximum 3.5% improved U_p and P_a in high order, which corresponds to 40% improvement compared to maximum 7% difference between FMOCM and PCM.

Acknowledgement.

The authors would like to thank the Ministry of Education of Korea for its financial support toward the Electrical and Computer Engineering Division at POSTECH through its BK21 program. This research was also supported in part by HY-SDR IT Research Center.

References

1. T. M. Kroeger and D. D. E. Long, "Design and implementation of a predictive file prefetching algorithm," in *Proceedigs of the 2001 USENIX Annual Technical Conference*, Boston, MA, Jan. 2001.
2. T. M. Kroeger and D. D. E. Long, "Predicting file system actions from prior events," in *Proceedigs of the USENIX 1996 Annual Technical Conference*, pp. 319-328, USENIX, Jan. 1996.
3. T. M. Kroeger and D. D. E. Long, "The case for efficient file access pattern modeling." in *IEEE Proceedings of the Seventh Workshop on Hot Topics in Operating Systems (HotOS-VII)*, Rio Rico, Arizona, Mar. 1999.
4. T. M. Kroeger and D. D. E. Long, "Predicting file system actions from prior events," *Master's thesis, University of California*, at Santa Cruz, 1997.

5. G. Kuenning, "The design of the seer predictive caching system," in *Workshop on Mobile computing Systems and Applications*, pp.37-43, IEEE Computer Society, Dec. 1994.
6. H. Lei and D. Duchamp, "An analytical approach to file prefetching," in *Proceedigs of USENIX 1997 annual Technical Conference*, USENIX, Jan. 1997.
7. V. N. Padmanabhan and J. C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," in *Proceedings of the 1996 SIGCOMM*, ACM, July 1996.
8. J. S. Vitter and P. Krishnan, "Optimal prefetching via data compression," *Journal of the ACM*, vol. 43, pp. 771-793, Sep. 1996.
9. K. M. Curewitz, P. Krishnan, and J. S. Vitter, "Practical prefetching via data compression, " in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*, pp. 257-266, Washington, D. C., May 1993.
10. J. Griffioen and R. Appleton, "Reducing file system latency using a predictive approach, " in *USENIX Summer Technical Conference*, pp. 197-207, June 1994.
11. T. M. Madhyastha and D. A. Reed, "Input/output access pattern classification using hidden Markov models, " in *Proceedings of the Fifth Workshop on Inpu/Output in Parallel and Distributed Systems*, (SanJose, CA), pp. 57-67, ACM Press, Nov. 1997.
12. A. Amer and D. D. E. Long, "Noah: Low-cost file access prediction through pairs, " in *Proceedings of 20th IEEE International Performance, Computing, and Communications Conference (IPCCC 2001)*, pp. 27-33, Phoenix, Arizona, Apr. 2001.
13. L. Mummert and M. Satyanarayanan, "Long term distributed file reference tracing: Implementation and experience, " *Sofware - Practice and Experience (SPE)*, 26(6):705-736, June 1996.
14. Ahmed Amer, Darrell D. E. Long, and Randal C. Burns, "Group-Based Management of Distributed File Caches, " *The IEEE International Conference on Distributed Computing Systems (ICDCS 2002)*, Vienna, Austria: July 2002.
15. Ahmed Amer, Darrell D. E. Long, Jehan-Francois Paris, and Randal C. Burns, "File Access Prediction with Adjustable Accuracy, " *Proceedings of the IEEE International Performance Conference on Computers and Communi-cation (IPCCC 2002)*, Phoenix, Arizona: Apr. 2002.
16. J.G. Cleary, W. Teahan, and I.H. Witten, "Unbounded length contexts for PPM, " J.A. Storer and M. Cohn (eds.), *Proc. IEEE Data Compression Conference*, Snowbird, Utah, 1995.

Parallel I/O Scheduling in Multiprogrammed Cluster Computing Systems

J.H. Abawajy

School of Computer Science,
Carleton University, Ottawa, Canada.
abawjem@scs.carleton.ca

Abstract. In this paper, we address the problem of effective management of parallel I/O in multiprogrammed cluster computing systems by using appropriate I/O scheduling strategies. The performance of I/O devices lags behind the performance of processors and network systems, resulting in I/O becoming the bottleneck in current systems. This gap is expected to increase in the future since I/O performance is limited by physical motion. Therefore, it is imperative that novel techniques for improving I/O performance be developed. Parallel I/O is a promising approach to alleviating this bottleneck. However, very little work exists with respect to scheduling parallel I/O operations explicitly. We propose new I/O scheduling algorithms and evaluate the relative performance of the proposed policies against the most commonly used approaches. Our preliminary results show that the proposed policies outperform current methods and can substantially enhance the performance of real-world scientific applications.

1 Introduction

In this paper, we focus on I/O-intensive large scale parallel applications. In addition to multiple processors, the parallelization of I/O operations and the use of multiple disk drives are required for achieving high system performance for the workloads of interest. The aggregate processing power and the storage capacity provided by cluster computing systems have lead to the widespread adoption of such systems as high performance computing platforms for various types of large-scal applications including I/O-bound applications such as digital libraries and image repositories.

Although processor and network technologies have seen impressive advances in performance, the lack of commensurate improvements in I/O devices has resulted in I/O becoming a major bottleneck in these systems. Due to physical limitations of I/O devices, significant improvements in access latency relative to processor and network speeds are unlikely. It is, therefore, imperative to find techniques that can improve the access performance of storage devices despite their high latency.

Recently, the I/O bottleneck in cluster computing systems has begun receiving increasing interest. In particular, parallel I/O has drawn increasing attention

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 223–229, 2003.
© Springer-Verlag Berlin Heidelberg 2003

in the last few years as a promising approach to alleviating I/O bottleneck in cluster computing systems. Parallel I/O combines a set of storage devices and provides interfaces to utilize them in concert. However, it has been shown that peak performance is rarely attained from these coordinated storage devices [1]. As a result, a number of approaches for more efficiently utilizing these resources have been continuously researched and developed [3,5,10,11].

Widely applicable solutions, however, will require an integrated approach which addresses the problem at multiple system levels, including applications, systems software, and architecture [14]. We propose that within the context of such an integrated approach, scheduling parallel I/O operations will become increasingly attractive and can potentially provide substantial performance benefits. Parallel I/O scheduling is concerned with scheduling of parallel I/O operations with the goal of minimizing the overall I/O response times.

As noted in [2], comparatively a lesser amount of work exists on parallel I/O scheduling. To this end, we present a set of I/O scheduling algorithms for multiprogrammed cluster computing environments running parallel I/O workloads. We present simulation results indicating that the proposed algorithms can produce a substantial improvement over previous algorithms.

The rest of the paper is organized as follows. Section 2 discusses previous work related to scheduling of parallel I/O operations. Section 3 presents the system and workload models used in this paper. Section 4 describe the system environment used in this study along a set of I/O scheduling algorithms we have implemented. Section 5 describe the testing environment along the system and workload parameters. The results of the experiments are also discussed. The conclusions and future directions are given in Section 6.

2 Related Work

By combining large numbers of storage devices and providing the system software to utilize them in concert, parallel I/O has extended the range of problems that may be solved on high performance computing platforms. However, the conglomeration of storage systems and providing convenient access alone is not enough to achieve maximum possible performance. In [8], for instance, it was found that when two processes concurrently read or write, the throughput per process drops to far less than half of the throughput obtained when only one process is making I/O requests. Thus, blindly allowing multiple processors to access files on the same disk partition simultaneously can be detrimental to performance.

In the last few years, several techniques to improve parallel I/O performance have been proposed in the literature. These techniques span from tuning the parallel applications [13] to scheduling the I/Os. One dimension that has been pursued in attacking the I/O bottleneck problem is the development of parallel and distributed file systems (e.g., Parallel Virtual File System (PVFS) [5]) and I/O runtime libraries (e.g., [6,10]) that distribute the datasets over several nodes and providing low-level data access mechanisms. Also a number of strategies

such as prefetchin and caching as well as fairly low-level techniques such as disk striping and interleaving have been proposed in the literature.

Another class of research is those that are focusing on the *data access* strategies such as collective I/O [3]. Although these *data access* strategies perform well for applications with specific I/O characteristics, they perform poorly for applications that do not confirm with the underlying assumption on which these strategies were built on.

Although the development of parallel/distributed file systems and access strategies have helped to ease the performance gap, but I/O still remains an area requiring significant performance improvement [1]. Also, currently many of the parallel filesystems and I/O runtime libraries address maximizing the performance of individual parallel applications having intense I/O requirements [14].

In multiprogrammed environments, where a number of applications are competing for resources, an important dimension and the focus of this paper is the effective management of parallel I/O by using appropriate I/O scheduling strategies. Research in I/O scheduling strategies for parallel workload by far is the least studied area [2]. In [14], scheduling a batch of I/O operations in parallel computer systems is discussed. An approach for scheduling parallel I/O that focus on uniprogrammed systems that run a single job at a time in isolation is discussed in [2]. In some cases, these techniques have also been extended to multiprogrammed environments that execute multiple parallel jobs simultaneously under the assumption that information such as the application execution time is known *a priori* [9,2]. Unfortunately, it is very difficult if not impossible to know the pending I/O service demands of a job rendering the policies proposed in [9,2] of theoretical importance than practical one. Also, most studies have not investigated the potential performance problems of handling large outstanding I/O requests in a multi-workload environments.

3 System Model

The *cluster computing* of interest has **P** workstations of similar architecture, connected by a fast interconnection network. As in [5], the **P** workstations are divided into a set of compute node and a set of I/O servers. Data is declustered among the I/O nodes to allow parallel access to different files and parallel access to the same file. This approach increases the performance and scalability of the system. An alternative approach is instead of dedicating extra processors for I/O, some compute processors can be part-time I/O servers as suggested in [7]. However, this strategy requires solving a number of difficult problems such as selection and placement of I/O servers [7] and its merit will be considered in the future.

We assume that parallel jobs arrive to the system in a stochastic manner with requirement for both compute and I/O resources. Each parallel job is composed of several tasks that can be scheduled to executed on a single workstation. In this paper, we focus on parallel I/O scheduling and we do not address the problem of

parallel job scheduling as this problem has been addressed in [15]. Data required to run a task is fetched before the task is run if it is not already present in the workstation on which the task is assigned to run.

All I/O requests from a job is sent to the data scheduler in the I/O subsystem, which coordinates the I/O request on multiple I/O nodes for greater efficiency. Typically each tasks in a parallel application will send a request to each I/O server when performing an I/O operation. It is easy to imagine that for a large parallel application a large number of I/O requests might be in service on an I/O Scheduler at one time. This large number of I/O requests provide us with an opportunity to optimize by selecting the order in which tasks will be serviced. Therefore, the order with which the I/O device services the I/O requests can affect the overall performance of the system. Upon arrival, if an I/O request cannot be serviced immediately, it is placed into an I/O wait queue and scheduled using I/O scheduling approaches discussed next.

4 Parallel I/O Scheduling Approaches

In a multiprogrammed environments, performance can be drastically reduced when multiple processors make requests concurrently due to contention for shared resources. Concurrent access to shared resources like disks or network interface causes significant overhead and can slow down applications. That also induces a significant performance gap in both I/O and communication, depending on the number of processors concurrently accessing the resources. In this section, we discuss briefly the classical I/O scheduling policy and propose two new policies.

First come First Served (FCFS) Policy: The FCFS I/O scheduling strategy is the most commonly used approach. In FCFS scheme, the scheduler maintains a single I/O wait queue, where all pending I/O requests are kept until they are serviced. When an I/O server becomes available, the scheduler schedules a pending request from the wait queue using the FCFS policy.

I/O Bunch Policy: This is a two-level scheduling policy where at the top level the scheduler creates a set of *I/O Bunches* from the I/O requests arriving to the data scheduler while the bottom layer maps each *I/O Bunch* onto the data servers. The scheduler creates n *I/O Bunches*, $B_1, ..., B_n$, each B_i is characterized as follows:

1. composed of I/O requests for a given file;
2. Inherits the arrival time of the oldest I/O request in the bunch; and
3. At most there are f I/O requests in the bunch, where f is a tunable parameter that can be controlled either at compile-time or dynamically at run-time.

At each scheduling round, the scheduler assigns a B_i to the I/O servers based on the order of B_i arrival time.

Job-based Policy: This policy is also a two-level scheduling policy and uses the concept bunching. However, the policy harnesses knowledge of I/O characteristics of the jobs in scheduling I/O requests, which include the number of CPU-I/O phases, the type and size of the requests. Note that unlike the policy proposed in [9], we explore information that can be easily extracted from the job either at the time of submission or at the compile-time. A simplified policy job-based policy we studied considers the size of the I/O requests. Each bunch, B_i, has a set of I/O requests of size less than or equal to **S**. Each bunch is then assigned to the I/O server where the request are served.

5 Performance Analysis

We compared the performance of the three I/O scheduling policies discussed in the previous section using I/O completion times as a performance metric. The system is composed of 14 workstations where two of them are used as an I/O servers. The file sizes range from 1 MB to 16 MB. Files were written and read by making from 1 to 128 request per file. A delay was placed between each request ranging from zero to one second. This delay represents the time for which a real application would perform computation or wait for some reason other than I/O. Table 1 summarizes the default parameters and values used in the experiments.

Table 1. Default Parameters and values.

Parameters	Values
Number of CPUS	14
I/O Types	CW, W, R
File sizes	1MB to 16MB
Requests/file	1 to 128
Requests size	1 to 128
Delays	0ms, 100ms, 500ms, 1000ms

The preliminary results of the experiments show that the proposed policies perform between 13% to 22% better than the FCFS policy under all system and workload parameters we tested. The job-based policy outperforms the other two policies.

This can be explained by the fact that FCFS does not consider disk access order at all, and may result in significant disk head movement when servicing multiple requests.

Our results suggests that the naive approach to scheduling I/O results in a large number of I/O operations, each of which is often for a very small amount of data. The added network cost of performing an I/O operation across the network is often high due to latency. We also observed that on single disk I/O

servers, as I/O queues are filled with large requests spanning many files, there is the potential for high seek penalties that can lead to variability in request time, lower I/O performance and throughput. Our results also suggests that using knowledge of the application I/O patterns, it is possible to achieve much more efficient I/O than more general solution.

These results are based on limited performance analysis and should be considered as a preliminary. We are undertaking rigorous testing to verify the results.

6 Conclusions and Future Directions

Parallel I/O is a promising approach to alleviating the problem of I/O bottleneck in cluster computing systems. We have been investigating an effective management of parallel I/O by using appropriate I/O scheduling strategies. Although parallel I/O scheduling can improve the performance, very little work exist on scheduling parallel I/O operations explicitly. We proposed new efficient I/O scheduling algorithms and evaluated the relative performance of the policies against the most commonly used I/O scheduling approach. These new I/O scheduling approaches have been shown, through simulation, to have superior performance to existing schemes and can substantially enhance the performance of real-world scientific applications. We are currently investigating their performance through experimentation with I/O traces taken from the scientific and non-scientific application domains, including web-servers and interactive applications. We are also looking at implementing and testing the proposed scheduling policies on .

References

1. R. Ross and W. Ligon, "Server-Side Scheduling in Cluster Parallel I/O Systems," *Special Issue of Calculateurs Paralleles,* 2003.
2. Fangyu Chen, Shikharesh Majumdar, "Performance of Parallel I/O Scheduling Strategies on a Network of Workstations," *ICPADS 2001,* pages: 157–164, 2001.
3. Phillip M. Dickens and Rajeev Thakur, "Evaluation of Collective I/O Implementations on Parallel Architectures," *Journal of Parallel and Distributed Computing,* **61**(8),(2001):1052–1076.
4. R. Ross, D. Nurmi, A. Cheng, and M. Zingale, "A Case Study in Application I/O on Linux Clusters," *Proceedings of SC2001, Denver, CO, November,*(2001).
5. P. Carns, W. Ligon III, R. Ross, and R. Thakur, "PVFS: A Parallel File System For Linux Clusters," *Proceedings of the 4th Annual Linux Showcase and Conference,* (Atlanta, GA, October 2000,) pp. 317–327.
6. Rajeev Thakur, William Gropp, and Ewing Lusk, "On Implementing MPI-IO Portably and with High Performance," *Proc. of the Sixth Workshop on I/O in Parallel and Distributed Systems,*(May 1999), pp. 23–32.
7. Y. Cho, M. Winslett, S. Kuo, Y. Chen, J. Lee, and K. Motukuri, "Parallel I/O on Networks of Workstations: Performance Improvement by Careful Placement of I/O Servers," *In Proceedings of the HiPer'98, High Performance Computing on Hewlett-Packard Systems (Annual conference of the HP2EUG),Switzerland, October 1998,* pages 104–111, Zurich.

8. Y. Cho, M. Winslett, S. Kuo, J. Lee, and Y. Chen, "Parallel I/O for Scientific Applications on Heterogeneous Clusters: A Resource-utilization Approach," *Proceedings of the 13th ACM International Conference on Supercomputing,Rhodes, Greece,* June 1999.
9. P. Kwong, S. Majumdar, "Scheduling of I/O in Multiprogrammed Parallel Systems," *Informatica,* **23**(1),(April 1999):67–76.
10. Kent E. Seamons. "Panda: Fast Access to Persistent Arrays Using High Level Interfaces and Server Directed Input/Output," *PhD thesis,* Dept. of Computer Science, University of Illinois at Urbana-Champaign, May 1996.
11. R. Arpaci-Dusseau, E. Anderson, N. Treuhaft, D. Culler, J. Hellerstein, D. Patterson, and K. Yelick. "Cluster I/O with River: Making the Fast Case Common," *In Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems, Atlanta, Georgia*, 1999.
12. Sandra Johnson Baylor, Caroline B. Benveniste, and Y. Hsu. "Performance evaluation of a parallel I/O architecture," *Technical Report RC 20049, IBM T. J. Watson Research Center,* May 1995.
13. A. Acharya, M. Uysal, R. Bennett, A. Mendelson, M. Beynon, J. Hollingsworth, J. Saltz, and A. Sussman. "Tuning the Performance of I/O-Intensive Parallel Applications," *In Proceedings of the Fourth Annual Workshop on I/O in Parallel and Distributed Systems,* pages 15–27, May 1996.
14. Ravi Jain, Kiran Somalwar, John Werth, J.c. Browne, "Heuristics for Scheduling I/O Operations," *IEEE Transactions on Parallel and Distributed Systems,* March 1997 (Vol. 8, No. 3), pp. 310–320.
15. J. H. Abawajy, "An integrated resource scheduling approach on cluster computing systems," *In Proceedings of the PDSECA/IPDPS-03), Nice, France,* April 22–26, 2003.

Design and Evaluation of Distributed Smart Disk Architecture for I/O-Intensive Workloads

Steve Chiu, Wei-keng Liao, and Alok Choudhary

Department of Electrical and Computer Engineering,
Northwestern University, Evanston, IL 60208 U.S.A.
{schiu,wkliao,choudhar}@ece.northwestern.edu

Abstract. Smart disks, a type of processor-embedded active I/O devices, with their on-disk memory and network interface controller, can be viewed as processing elements with attached storage. The growing size and access patterns of today's large I/O-intensive applications require architectures whose processing power scales with the storage capacity. We evaluate a distributed smart disk architecture with representative I/O-intensive workloads including TPC-H queries, association rule mining, data clustering, and 2-D fast Fourier transform applications to study the proposed architecture.

1 Introduction

A hard disk drive typically consists of one or more disk platters, read/write heads, and other embedded general- or special-purpose processors. While much of the processing power on the disk is currently used for scheduling and optimizing disk bandwidth, the rapid advancements of processor and memory technologies promise to provide future hard disk drives with application-level processing capabilities to perform more complex operations directly at the drive [1]. For I/O-intensive applications that process large amount of disk-resident data, various storage architectures have been proposed to exploit the parallelism available from the large number of disks by offloading certain elementary operations to the disk drive. We study the benefits of a distributed smart disk (SD) architecture using TPC-H queries, association rule mining, clustering, and fast Fourier transform.

The remainder of this paper is organized as follows. Section 2 briefly reviews the work related to various SD architectures. In Section 3, we describe a distributed SD architecture in single- and multiple-disk configurations. Section 4 explains the algorithms and implementations of our workloads. Section 5 presents our experiments and results. Conclusions and future work are in Section 6.

2 Background

Five major factors catalyzed the evolution of storage architectures: I/O-bound workloads, improved disk drive attachment technologies, increased on-drive transistor density, emergence of new interconnects, and cost of storage systems [2]. Typically,

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 230–241, 2003.
© Springer-Verlag Berlin Heidelberg 2003

a SD consists of an embedded processor, a controller, on-disk memory, local disk space, and a network interface. The embedded processor would have a speed of 300 – 500 MHz, and the memory size between 16 and 128 MB, while the disk space would be in tens of GB or more.

2.1 Database Machines and Parallel Databases

The concept of performing application-level processing on the disk originates from the research in database machines in the late 1970s. The limited disk bandwidth and the complexity of programming special-purpose hardware at that time eventually lead to the extinction of the database machines. Nevertheless, parallel database research has since produced efficient algorithms for large distributed-memory database servers that employ commodity processing elements [3]. This development and the growing processing power on the disk drive have prompted some researchers to suggesting that all application-level processing would eventually be performed at the disk [4].

2.2 Active Disks, IDISKs, and Smart Disks

Processor-embedded disk architectures changed the processing model of large databases by offloading I/O-intensive tasks to the disk. Thus, rather than sending to the host, data are filtered or processed directly at the disk, significantly reducing the network traffic between host and disk. Taking this approach, an Active Disk architecture was proposed by assuming application-level on-disk processing and large on-disk memory [5]. Active Disks use a stream-based programming model to address the software structure and implementation, in which host-resident code interacts with disk-resident code using streams [6]. The Active Disks work in [7] investigated scan-based algorithms for databases, nearest neighbor search, frequent sets, and edge detection. The Intelligent Disks (IDISKs) [8], proposed to succeed networks of workstations, uses on-disk integrated processor-in-memory called IRAM, and can perform direct disk-to-disk communication.

The Smart Disks architecture [9] introduced operation bundling to optimize query execution. Execution of each operation bundle is offloaded to the disk. While data transfer between disks was through the central unit, there was less involvement by the central unit in the overall query execution. Results from [9] showed that Smart Disks outperformed both host-based and cluster systems in comparable configurations.

3 Distributed Smart Disks (SD) Architecture

We propose a fully distributed SD architecture, where a SD group consists of SDs interconnected by a switched network. Applications such as database queries can be applied onto the group from a remote client. Figure 1 illustrates this single SD group (SDG) architecture. To exchange, combine, or re-distribute data within an SDG, data is communicated between the involved SDs without being routed through the client. For data queries to multiple database machines located remotely across the network, communication within a SDG as well as between several SDGs will be necessary. As

shown in Figure 2 of the multiple-SDG architecture, inter-SDG communication would occur when databases reside on separate SDGs, which may belong to different storage area networks that are connected to a common router. This architecture exploits data parallelism to meet the storage, computation, and communication requirements of future I/O-intensive applications. The architecture would utilize emerging interconnect technologies for low-latency and high-bandwidth data communication.

3.1 Storage Interconnect

Since the design of the SD architecture pushes data queries down to the disk, a network with high connectivity is favorable as its communication infrastructure. Traditional I/O architectures employ PCI/PCI-X bus protocols to communicate with storage systems and networks. Recent research on network-attached storage systems suggests the use of switched networks to maintain sufficient I/O bandwidth. We use the InfiniBand Architecture (IBA) [10] to represent the storage interconnect. IBA networks employ channel-based switched fabric, host/target channel adaptors (HCA/TCA), switches and routers to provide simultaneous point-to-point communications between end nodes. All end nodes communicate through switches with any other nodes in a subnet; or via a router to another end node.

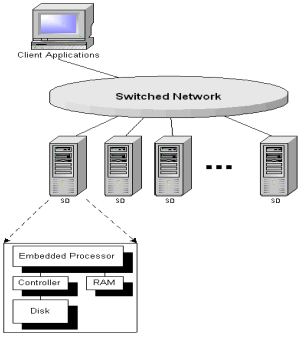


Fig. 1. The single smart disk group (SDG) architecture

4 Workloads: TPC-H, ARM, Data Clustering, and 2-D FFT

4.1 TPC-H Queries

For distributed SD architectures, performing parallel database operations is one of the logical approaches to exploit data parallelism. We used full queries Q_1, Q_6 and Q_{12} from the TPC-H benchmark suite to define our database workload.

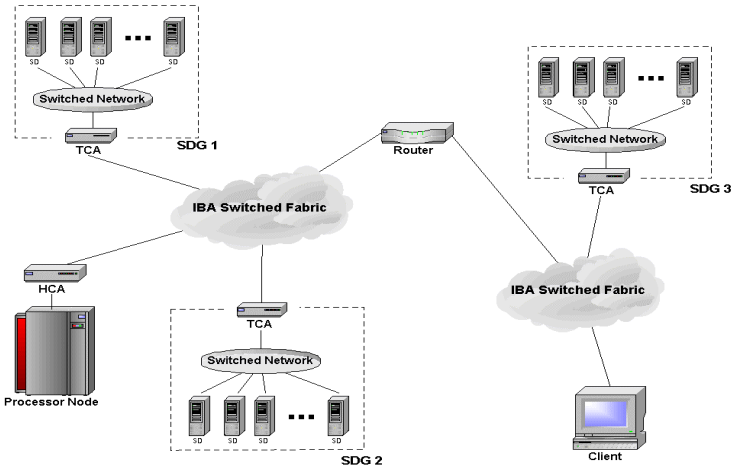


Fig. 2. The multiple-SDG architecture over an InfiniBand-based network

4.1.1 Query Primitives

We offloaded the *scan*, *join*, *sort*, *group-by* and *aggregate* primitives to the SDs, and implemented these primitives in parallel. We assume the database tables are evenly distributed across all SDs so that full parallelism of database queries can be exploited. In our implementation, the entire processing of a primitive is performed on the SDs without the remote client's participation or management. The *scan* primitive is implemented by sequentially retrieving and evaluating every tuple in the disk-resident data. The main purpose of *scan* is to filter data for further processing. Due to the limited amount of on-disk memory, out-of-core paging strategy was implemented. Since *scan* involves only local disk access, there is no inter-disk communication. For the remaining primitives, disk-to-disk communication is required. We adopted an out-of-core parallel bucket sort [11] for our *sort* implementation. It has four stages: sampling, re-distribution, internal sort, and external merge. Since *group-by* and *aggregate* are usually combined with *sort* and each is performed with a global communication, an analysis on *sort* would cover *group-by* and *aggregate*.

4.1.2 *Join* with Two SDGs

Two scenarios exist for the *join*: the two tables to be joined can reside in the same or different SDGs, or likewise, in one or more database machines. The former is the traditional implementation of a parallel *join*. In this work, we are interested in the latter scenario. Assume that the two tables to be joined, R and S, reside in SDG 1 and 2, respectively. Parallel join of these two tables will thus involve both intra- and inter-SDG communications. Assuming R is evenly partitioned across P SDs and S is evenly partitioned across Q SDs (P need not equal Q), the *join* processing proceeds as illustrated in Figure 3. Each SDG, denoted "SDG 1" or "SDG 2", starts by locally hashing its R or S tuples into "destination files" (or send buffers) for each of the SDs in both SDG 1 and SDG 2. Each SD then reads the tuples from its local "destination files", and sends the tuples to their destination SDs. Meanwhile, each SD also receives tuples destined for it from both SDGs. Upon completion of this re-distribution phase, each

SD will hold its R and S tuples, which are joined locally using a simple nested-loop join algorithm.

4.1.3 Full Query Execution

We assume that input tables are initially distributed across the SDs, and use PSQL's (a PostgreSQL database) *explain* command [12], along with TPC-H executable query text files to generate the query plans. The processing flow of these full TPC-H queries proceeds according to the full query execution protocol specified in Table 1.

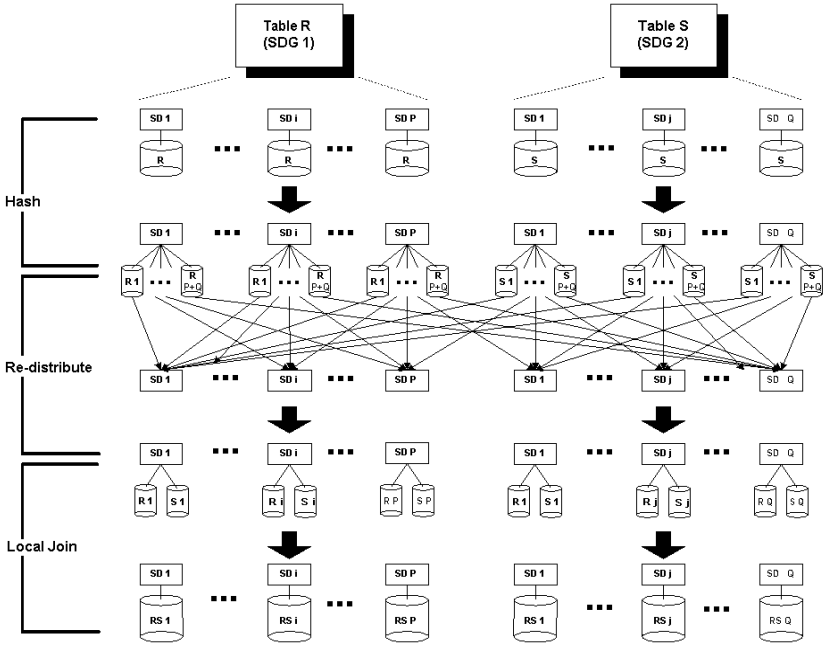


Fig. 3. Processing of *join* from two separate SDGs.

Table 1: TPC-H full query execution protocol

TASK	REMOTE CLIENT	SMART DISKS
Query parsing/optimization	Parse and optimize query	—
Query plan generation	Generate query plan	—
Query execution	Send the query plan to SDs Await completion signal from SDs Receive signal from the SDs	Receive the query plan Execute the query plan – Perform primitives – Re-distribute data – Store results locally – Communicate data Signal client of completion
Response to user	Receive results from the SDs Respond to user	Send results to client

4.2 Association Rule Mining (ARM) and Frequent Set Counting (FSC)

ARM techniques have been used to help address marketing questions such as "the 90% of customers who buy A probably also buy B". Given a collection of transactions T and each transaction contains items from an itemset I, an association rule takes on the expression $X \Rightarrow Y$, where itemset $X \subseteq I$ and itemset $Y \subseteq I$. An association rule $X \Rightarrow Y$ holds in T with a confidence C and a support S, if at least $C\%$ of all the transactions (in T) containing X also contain Y, and $X \Rightarrow Y$ exists in at least $S\%$ of all the transactions in T [13,14]. Typically, an ARM process consists of two phases. In the first phase, a set of *frequent* itemsets is found. In the second phase, the rules that satisfy the minimum S and the minimum confidence C are identified.

4.2.1 Count Distribution and Hybrid Distribution Methods

Our ARM FSC primitive exploits the total memory of an SD system for higher efficiency. The Count Distribution (CD) method is based on the *Apriori* algorithm in [15]. In the CD method, each SD executes the *Apriori* algorithm serially, followed by a global reduction to obtain the final results. The Hybrid Distribution (HD) method partitions the frequent itemsets into sufficiently large sections, then assign a group of SDs to each section. Thus, the HD method dynamically configures the processor grid for more effective load balancing.

4.3 Clustering for Large Data Sets

Clustering techniques help discover interesting patterns (called clusters) from large data sets. Such patterns often exist in high-dimensional space. Common applications that exploit clustering techniques include collaborative filtering [16], organization of document datasets [17], image processing and seismic studies, to name a few. Efficient clustering techniques address data and noise that both exist in high-dimensional spaces and subspaces, which result in an exponential growth of the search space for clusters. We implemented a data clustering primitive based on the parallel adaptive-grid and density-based algorithms studied in [18]. Define $S=A_1\times A_2\times...\times A_d$ as a d-dimensional numeric space, where $A=\{A_1,A_2,...,A_d\}$ is the attribute set and $D=\{D_1,D_2,...,D_d\}$ is the domain set. Assume $r=(r_1,r_2,...,r_d)$ is a d-dimensional input record. The space S is divided into a grid of non-overlapping cells, where each cell C is defined to be $c_{1k'}\times c_{2k''}\times...\times c_{dk'...'}$ and for all $i \in \{1,2,...,d\}$, $c_{ik'} \subseteq D_i$ and $\cup_k c_{ik} = D_i$. That is, $c_{ik'}=[l_{ik'},u_{ik'})$ is the partitioning interval for A_i. The cell C is *dense* if the fraction of the total data points contained in C is greater by some given factor than the value expected if the data were uniformly distributed in S. A cluster is thus a union of connected dense cells. The clustering primitive first computes the histogram for A_i, adaptively setting the bin size, then builds candidate dense cells and perform subspace clustering on the local records r to compute the local dense cells. Reduction is performed to obtain global data on the histogram, candidate dense cells, dense cells, and bounds for the dense cells.

4.4 Two-Dimensional Fast Fourier Transform (2-D FFT)

FFT is used in many scientific problems. The memory requirement on the SD motivates our evaluation of out-of-core 2-D FFT. The FFT takes $O(N \lg N)$ time to compute the discrete Fourier transform of an $N \times N$ matrix. For a distributed-memory model such as our SD architecture, the 2-D FFT can be performed using a parallel out-of-core transpose- and redistribution-based algorithm [19,20] as follows:

(1) Distribute input matrix A by rows as (BLOCK, *).
(2) Perform 1-D FFT on the rows of matrix A.
(3) Perform 2-D transpose on the intermediate matrix B from step (2).
(4) Perform 1-D FFT on the rows of B^T.

We evaluated the out-of-core 2-D FFT over a 2048×2048 complex matrix on the distributed SD architecture. Our implementation is based on the distributed-memory 2-D FFT algorithm prescribed in [21].

5 Experiments and Results

We first compared the performance of the fully distributed SD architecture with that of the partially distributed SD architecture, using TPC-H Q_1, Q_6 and Q_{12}. Then, we evaluated the performance and scalability of the fully distributed SD architecture for the ARM FSC, data clustering, and 2-D FFT. The following platform was used to simulate the workloads: 32 Pentium III Linux PCs running at 500 MHz, each with 64 MB of RAM and a local disk of 6 GB. The PCs are connected via a switched network. We used TPC-H database generator [22] to populate synthesized data into tables with scale factors (SF) of 0.1 and 1.0. *DiskSim* 2.0 simulator [23] was used to simulate the cost of accessing the disk drive.

5.1 Fully vs. Partially Distributed SD Architectures

We simulated with 4, 8, 16, and 32 SDs, page size of 8 KB, on-disk memory size of 32 MB, and SF of 0.1 and 1.0. We measured the execution times of TPC-H Q_1, Q_6 and Q_{12}. Figures 4 and 5 suggest that the fully distributed SD system provides further performance improvement over the partially distributed system. We used the formula T_{PD}/T_{FD} to evaluate the improvement obtained with a given number of SDs, where T_{PD} denotes the query execution time on the partially distributed SD system and T_{FD} denotes that on the fully distributed SD system. For example, in processing Q_6, hardly any noticeable improvement was achieved regardless of the number of SDs used, since Q_6 contains less than 0.01% of communication time. On the other hand, Q_1 contains about 10.4% communication time, while Q_{12} averages 0.56%. The low proportion of communication in Q_{12} is attributed to the highly selective *scan* that was performed prior to the *join*, *sort*, and *aggregate* operations in that query. Consequently, when running at an SF of 1.0 (the validation data size required by the TPC-H benchmark) the T_{PD}/T_{FD} values for Q_1 ranged from 2.88 to 3.41, and from 1.20 to 2.30 for Q_{12}, as

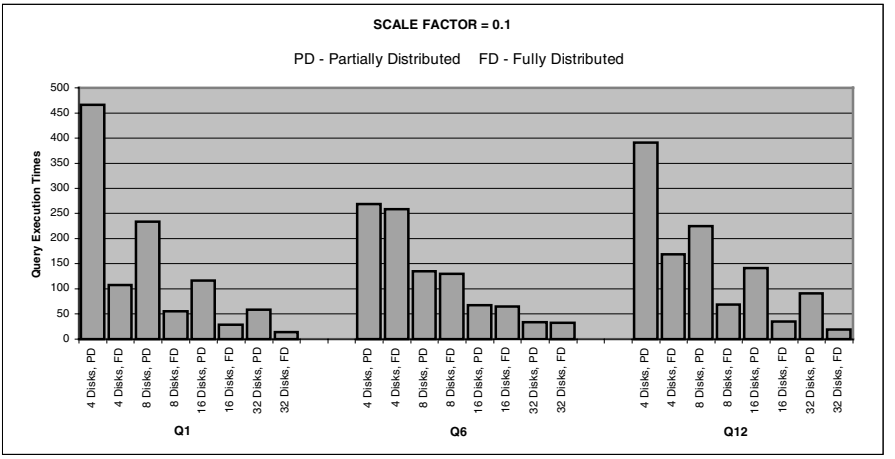


Fig. 4. Performance of partially vs. fully distributed SDs on Q_1, Q_6 and Q_{12} with SF of 0.1

expected. Increasing the data size from SF=0.1 to SF=1.0, however, exposed the inefficiency of our nested loop-based *join* primitive.

5.2 Association Rule Mining Frequent Set Counting

A synthetic transaction database was generated using the data generator from IBM [24]. The database had an average transaction size of 20, each transaction containing up to 25 distinct items. 100K transactions were generated for each SD. Figures 6 and 7 present the results for our ARM FSC primitive executed on SD systems with 2, 4, 8, 16 and 32 disks, using data page sizes of 4K, 8K, 16K, 32K and 64K bytes, and 64MB on-disk memory. The single-SDG processing model was assumed in this experiment. Figure 6 shows the results when the primitive uses the CD method; Figure 7 shows when the primitive uses the HD method.

Note that since data size increases linearly with the number of SDs, keeping the response time constant demonstrates scalability. Except for the data page size of 16 KB, the performance of the two methods are quite comparable at the 5% minimum support level– an indication that the hash tree for the frequent itemsets was able to fit into the available (64 MB) on-disk memory used in our evaluation.

5.3 Data Clustering Primitive

The data-clustering primitive discussed in Section 4.3 was executed over a data set of 2,234,961 records (≈ 179 MB), which contains 20-dimensional data with 5 clusters, and each cluster is of 5 dimensions. As with the ARM FSC primitive, we experimented with SD systems of 2, 4, 8, 16 and 32 disks, using data page sizes of 8K, 16K, 32K and 64K bytes with 64 MB on-disk memory. We also executed the primitive on a system with only 1 SD, where the processing is effectively serial.

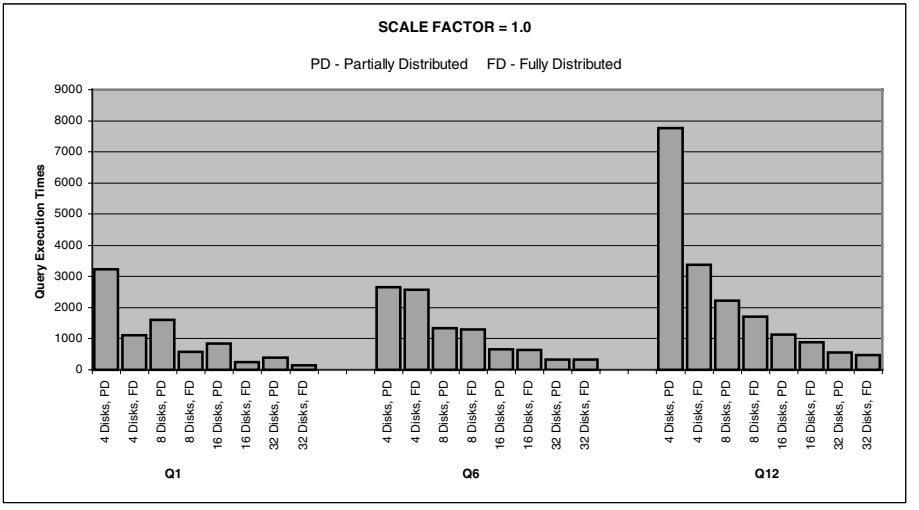


Fig 5. Performance of partially vs. fully distributed SDs on Q_1, Q_6 and Q_{12} with SF of 1.0

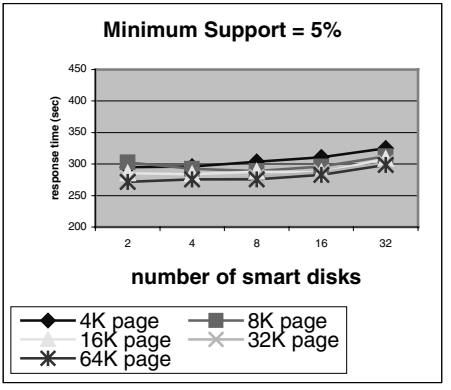


Fig. 6. Performance of FSC with the CD method

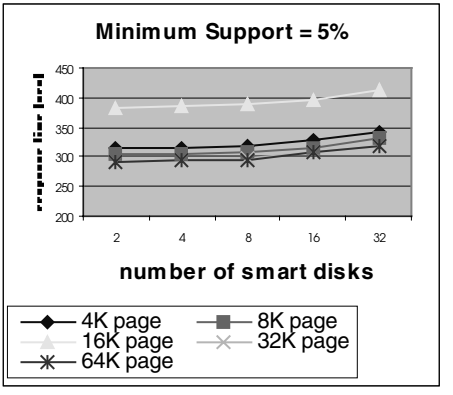
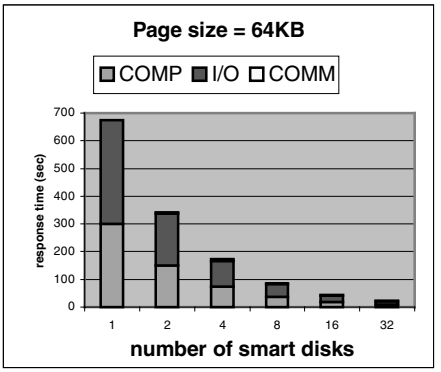


Fig. 7. Performance of FSC with the HD method

Figure 8 shows the performance of the clustering primitive when system size is scaled up, with the page size fixed at 64 KB. The results demonstrated that both the compute time taken to populate the candidate dense cells and the I/O time decreased when the number of SDs increases. Results for page sizes of 8 KB, 16 KB and 32 KB showed similar trend. Figure 9 presents the total response times of the clustering primitive versus number of SDs for all four page sizes. The results show a reduced response time as we increased data page size.

5.4 Two-Dimensional FFT

We evaluated the performance of the 2-D FFT primitive with 1, 2, 4, 8, 16 and 32 SDs for various data page sizes and 64 MB on-disk memory, and normalized all execution times with respect to the base case where 1 SD with a data page size of 4 KB are used.



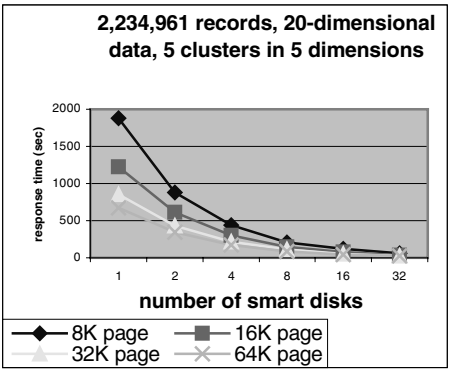


Fig. 8. Performance of clustering primitive **Fig. 9.** Overall scalability of clustering primitive

Our results show larger speedups for smaller data pages at smaller system size. As we increase the number of SDs, such difference in speedups diminishes. Figure 10 presents our results for the 2-D FFT workload.

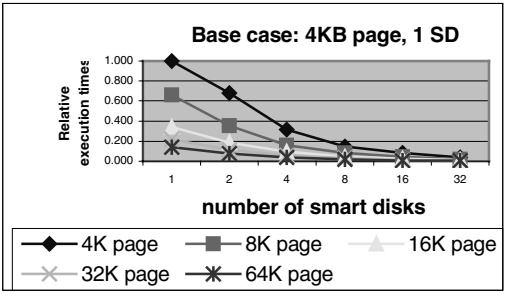


Fig. 10. Relative execution times of 2-D FFT

6 Conclusions and Future Work

For disk-based storage systems, increasing processing power on the disk results in offloading user-level code closer to data. Built on the progress to date, we showed that a distributed Smart Disk model is feasible. We evaluated a distributed Smart Disk architecture using TPC-H queries, ARM FSC, data clustering and 2-D FFT workloads. Our results suggest that such architecture would outperform a partially distributed, and thus, a centralized system. Our results also demonstrate that the Smart Disk architecture scales well with increased system size and data size. For future work, query optimization techniques for distributed-memory parallel databases can be studied to exploit cost models and load-balancing techniques.

References

1. W. Hsu, A. Smith and H. Young, "Projecting the Performance of Decision Support Workloads on Systems with Smart Storage (SmartSTOR)". Report No. UCB/CSD-99–1057. August 1999.
2. G. Gibson and R. Van Meter, "Network Attached Storage Architecture". Communications of the ACM, 43(11). November 2000.
3. D. Dewitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Processing". Communications of the ACM, 36(6). June 1992.
4. J. Gray and A. Reuter, Transaction Processing: Concepts and Technique. Morgan Kaufmann.
5. M. Uysal, A. Acharya and J. Saltz, "Evaluation of Active Disks for Decision Support Databases". International Conference on High Performance Computing Architecture. January 2000.
6. Acharya, M. Uysal and J. Saltz, "Active Disks: Programming Model, Algorithms and Evaluation" International Conference on ASPLOS. 1998.
7. E. Riedel, C. Faloutsos and D. Nagle, "Active Disk Architecture for Databases". Carnegie Mellon University. Report No. CMU-CS-00-145. April 2000.
8. K. Keeton, D. Patterson and J. Hellerstein, "A Case for Intelligent Disks". SIGMOD Record, 27(3).
9. G. Memik, M. Kandemir and A. Choudhary, "Design and Evaluation of Smart Disk Architecture for DSS Commercial Workloads", International Conference on Parallel Processing.
10. InfiniBand Trade Association, "InfiniBand Architecture Specification Volume 1 Release 1.0.a".
11. X. Li, P. Lu, J. Schaeffer, J. Shillington, P. S. Wong and H. Shi, "On the Versatility of Parallel Sorting by Regular Sampling". Parallel Computing, Vol. 19, pp. 1079–1103. 1993.
12. Yu and J.Chen, "The PostGreS95 User Manual". University of California, Berkeley. July 1995.
13. E-H Han, G. Karypis and V. Kumar, "Scalable Parallel Data Mining for Association Rules". IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 3. May/June 2000.
14. S. Orlando, P. Palmerini, R. Perego and F. Silvestri, "An Efficient Parallel and Distributed Algorithm for Counting Frequent Sets". VECPAR 2002 HPC for Computational Science. June 2002.
15. R. Agrawal and J. C. Shafer, "Parallel Mining of Association Rules". IEEE Transactions on Knowledge and Data Engineering, 8(6): 962–969. December 1996.
16. L. H. Ungar and D. P. Foster, "Clustering Methods for Collaborative Filtering". AAAI Workshop on Recommendation Systems. 1998.
17. Y. Zhao and G. Karypis, "Evaluation of Hierarchical Clustering Algorithms for Document Datasets". University of Minnesota Technical Report #02–022.
18. H. Nagesh, S. Goil and A. Choudhary, "Parallel MAFIA: Parallel Subspace Clustering for Massive Data Sets". Data Mining for Scientific and Engineering Applications. Academic Publishers. 2001.
19. R. Bordawekar, A. Choudhary and J. Ramanujam, "Compilation and Communication Strategies for Out-of-Core Programs on Distributed Memory Machines". CIT Center for Advanced Computing Research. Scalable I/O Initiative Technical Report. November 1995.

20. M. Kandemir, A. Choudhary and J. Ramanujam, “Improving Locality in Out-of-Core Computations Using Data Layout Transformations”. Carnegie Mellon University. May 1998.
21. LLNL website: http://www.llnl.gov/computing/tutorials/workshops/workshop/mpi/exercise.html.
22. Transaction Processing Performance Council, Soft Appendix for TPC-H and TPC-R Benchmarks.
23. G. Ganger, B. Worthington and Y. Patt, “The DiskSim Simulation Environment Version 2.0 Reference Manual”. Carnegie Mellon University. December 1999.
24. IBM Quest Data Mining Project website: http://www.almaden.ibm.com/cs/quest/syndata.html.

An Adaptive Load Balancing Algorithm for Large Data Parallel Processing with Communication Delay

Kenji Imasaki and Jemal H. Abawajy

Carleton University, Ottawa, Canada,
{kenji,abawjem}@carleton.ca,
http://www.scs.carleton.ca/

Abstract. Achieving good load balance in cluster environment is difficult due to its dynamic nature. Besides, the data to be processed may be transfered from geographically distant sites at a fluctuating transfer rate. This paper proposes a load balancing algorithm to deal with such fluctuating transfer rate. The novelty of the algorithm is to use buddy concept in processing node grouping and the data distribution statistics obtained at the source sites. We use single join processing in database query processing as its application.

1 Introduction

Cluster computing technology is now fully blossomed due to the technology advances in processors, memory and high speed network. In a cluster, processing nodes, which are built from commodity products, are connected by LAN and heterogeneous in nature. There are interactive users who log-in/out from time to time so that it creates dynamic load changes and makes it difficult for applications to achieve its maximum performance.

Examples of such applications can be run on clusters are scientific computing and database processing. The input data for these applications such as measured data from sensors or XML structured database data, are large in size and may reside geographically remote site. Thus, the data has to be transfered to the cluster prior to the processing. The data transfer rate is unpredictable since it source site may be distant from the cluster and the connection between them may not be stable. At the same time, the applications should be parallelized because of their huge data sizes.

As a result, it is important to develop a good load balancing algorithm on a cluster to cope with data transfer rate fluctuation and dynamic load change within the cluster. The purpose of the algorithm is to minimize the response time and maximize the system utilization. In this paper, we will present an load balancing algorithm to deal with fluctuating data transfer rate and dynamic load changes. The novelty of the algorithm is to use buddy concept in processing node grouping and the data distribution statistics obtained at the source sites. The

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 242–247, 2003.
© Springer-Verlag Berlin Heidelberg 2003

data is very helpful in the load balancing decision. We use single join processing in database query processing as its application.

This paper is organized as follows: Section 2 describes related work. Section 3 presents our load balancing algorithm with system architecture description. Section 4 will conclude the paper.

2 Related Work

The fluctuating data transfer rate problem is tackled in database query processing area. Several data integration systems [2] [3] [4] [5], which process queries over geographically distributed databases, have been developed. Most of the research have been concentrating in developing algorithms to deal with the fluctuating data transfer rate. Amsaleg et al. [1] proposed query scrambling in which the sub-query are "scrambled" according to the arrival rate of the data. Urhan [12] proposed XJoin in which several kinds of joins are executed while arriving data is stalled.

Single join processing, which is dealt in this paper, is a classic problem in the database query processing. Hash join based algorithms are the best algorithm [11]. To deal with data skew, several load sharing/balancing algorithms are proposed. Their comparison is summarized by Hua et al.[6].

Imasaki et al. proposed adaptive single hash join algorithms [7] [8] and multiple join algorithms [9] [10] on a workstation cluster. Their algorithms are called chunk-HJ which combines a chunking method with hash join to manage dynamic changes that occur in cluster environment.

3 Load Balancing Algorithms

This section first explains the query processing architecture that we used and then introduces the load balancing algorithm on the architecture.

3.1 The Query Processing Architecture

The query processing architecture which is dealt in this paper is shown in Figure 1. We assume processing relations are distributed over the Internet. A Database Management System (DBMS) is running on a remote site and reads relations from the database according to users' requests.

We use a cluster (called main cluster) to process queries submitted by users within the cluster. The main cluster with a large number of processing nodes is responsible for the query processing. The Query Manager (QM) runs on one of the processing nodes and is responsible for executing queries. The Join Manager (JM) is responsible for executing joins. Processing nodes (P) within the main cluster are divided into several *buddies*. The JM decides the work distribution among the buddies. The function of the buddy will be explained later in detail. The buddy manager (BM) is invoked for each buddy and is responsible for work distribution among processing nodes within the buddy.

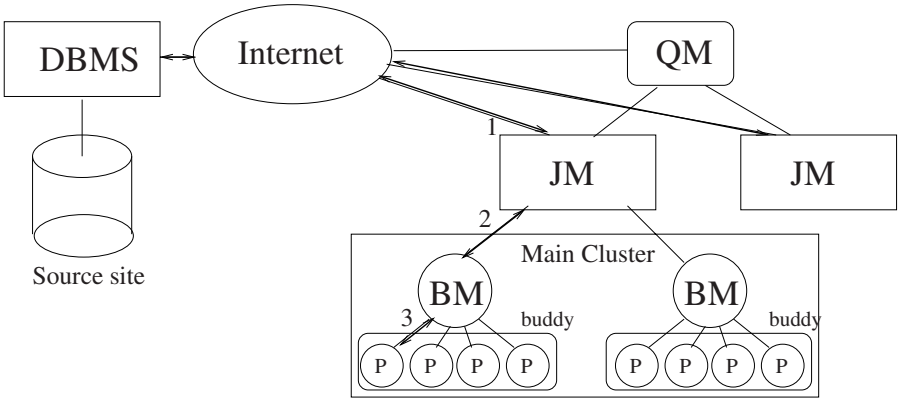


Fig. 1. The Query Processing Architecture

3.2 The Proposed Algorithm

The proposed algorithm for single join processing is explained by using the notation given in Table 1. The algorithm is divided into three parts as follows:

Interaction between source site and QM/JM ('1' in Figure 1)

1. Source sites ($DBMS_R$ and $DBMS_S$) start reading R^i and S^i, apply predetermined hash function to them and get R_h and S_h.
2. After reading *sampleSize* of R and S, DBMSs start sending the bucket distribution statistics to JM.
3. JM decides the BG assignments from R_h and S_h using bucket distribution statistics and load balancing consideration and let the source sites know the decision.
4. Source sites ($DBMS_R$ and $DBMS_S$) send buckets which belongs to the same BG together to JM.

Interaction between JM and BM ('2' in Figure 1) JM controls the load balancing among BMs by changing the number of processing nodes for each buddy at a certain interval based on the amount of work the buddy will process in future. There are two kinds of such statistics: $BG_x^{arrived}$ and BG_x^{source}. Their relationship is shown in Figure 2, together with BG_x^{past} in the case of the number of BGs is 4. In this figure, the number of tuples that belongs to BG_1 ($BG_1^{arrived}$) is small right now. However, it will increase in the future as BG_1^{source} is rather large. The amount of this time delay for the increase depends on the transfer speed of the network between the source site and the main cluster.

Thus, the number of processing nodes for BG_x (p_x) is decided assuming p processing nodes are available as follows:

$$p^{arrived} = x * p$$

Table 1. Notation used in algorithm description

R,S	two relations involved in the join
$DBMS_R$	DBMS for relation R
$DBMS_S$	DBMS for relation S
R^i	ith R segment
R_h	hth R hash bucket after applying the hash function
BG_x	a bucket group which consist of several R/S buckets
BG_x^i	ith segment of BG_x; the unit for work allocation
sampleSize	sampling size
p	the total number of available processing nodes
p_x	the number of processing nodes for a buddy
$p^{arrived}$	the total number of processing nodes for arrived data
p^{source}	the total number of processing nodes for source data
$p_x^{arrived}$	the number of processing nodes for a buddy for arrived data
p_x^{source}	the number of processing nodes for a buddy for source data
BG_x^{past}	the processed part of BG_x
$BG_x^{arrived}$	the arrived part of BG_x
BG_x^{source}	BG_x at the source site

$$p^{source} = (1 - x) * p$$

$$p_x = (BG_x^{arrived})/(\sum_i BG_i^{arrived}) * p_x^{arrived} + (BG_x^{source})/(\sum_i BG_i^{source}) * p_x^{source}$$

where x is the constant and and it is set between 0 to 1; x is set to 0 when source site is very close to the main cluster and/or the network speed is very fast; x is set to 1 in the opposite case.

With the new p_x, JM adjusts the number of processing nodes for each buddy.

Load Balancing Within Buddy ('3' in Figure 1) The algorithm within the buddy can be any parallel single join algorithm. We use the chunking-HJ proposed by Imasaki et al.[8]. It has robust performance that takes into account the following factors: data size, degree of data skew, degree of background load, or any combination of these. The algorithm does not require any knowledge of distribution and it does not involve pre-processing of the input relations.

4 Concluding Remarks

This paper proposed an adaptive load balancing algorithm for query processing with large communication delay. The algorithm uses buddy concept and data distribution statistics obtained at source sites. As we described in Introduction, this algorithm can be used in not only database processing but also other applications dealing with large amount of data.

We are currently evaluating the algorithm using an event-driven simulation programs. The fluctuating transfer rate is simulated by collected trace data.

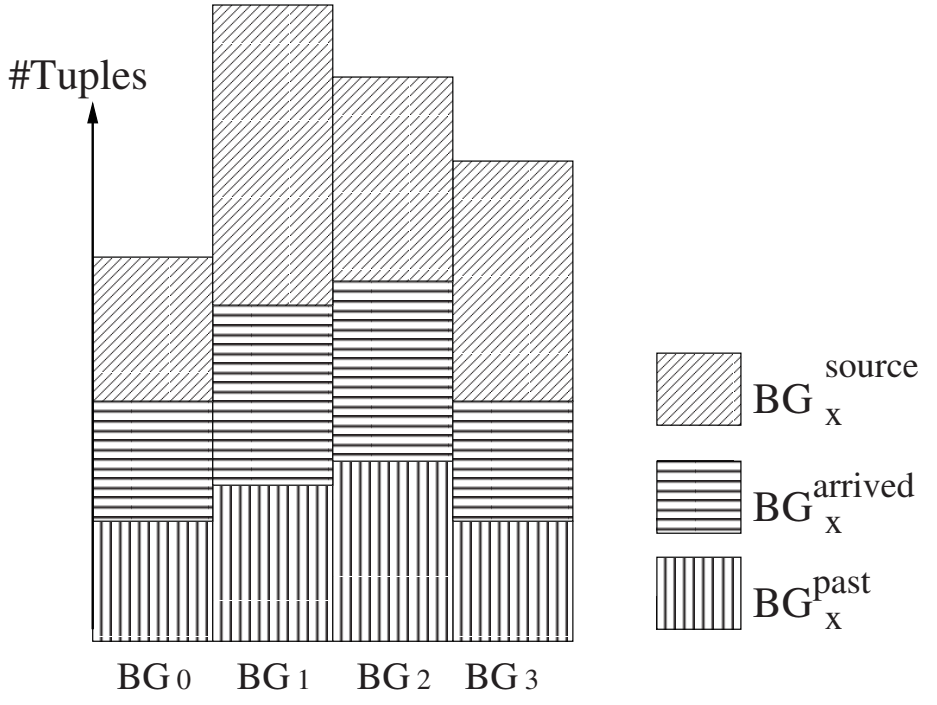


Fig. 2. Bucket Group Distribution in the case its number is 4

References

1. L. Amsaleg, M. J. Franklin, A. Tomasic, and T. Urhan. Scrambling Query Plans to Cope with Unexpected Delays. In *Fourth International Conference on Parallel and Distributed Information Systems (PDIS '96)*, pages 208–219, Miami Beach, Florida, USA, Dec. 1996. IEEE Computer Society Press.
2. R. Avnur and J. M. Hellerstein. Eddies: Continuously Adaptive Query Processing. In W. Chen, J. Naughton, and P. A. Bernstein, editors, *The 2000 ACM SIGMOD International Conference on Management of Data*, volume 29(2) of *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pages 261–272, Dallas, Texas, May 2000. ACM Press.
3. L. Bouganim, F. Fabert, C. Mohan, and P. Valduriez. A Dynamic Query Processing Architecture for Data Integration Systems. In *16th International Conference on Data Engineering*, pages 425–434. IEEE Computer Society Press, Mar. 2000.
4. L. Bouganim, F. Fabert, C. Mohan, and P. Valduriez. Dynamic Query Scheduling in Data Integration Systems. *IEEE Data Engineering Bulletin: Special Issue on Adaptive Query Processing*, 23(2):42–48, June 2000.
5. J. M. Hellerstein, M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. Shah. Adaptive Query Processing: Technology in Evolution. *IEEE Data Engineering Bulletin: Special Issue on Adaptive Query Processing*, 23(2):7–18, June 2000.

6. K. A. Hua and W. Tavanapong. Performance of Load Balancing Techniques for Join Operations in Shared-nothing Database Management Systems. *Journal of Parallel and Distributed Computing*, 56(1):17–46, Jan. 1999.
7. K. Imasaki and S. Dandamudi. Performance Evaluation of Nested-loop Join Processing on Networks of Workstations. In *Proceedings of the Seventh International Conference on Parallel and Distrubuted Systems*, pages 537–544, Iwate, Japan, July 2000.
8. K. Imasaki and S. Dandamudi. An Adapive Hash Join Algorithm on a Network of Workstations. In *International Parallel and Distributed Processing Symposium (IPDPS)*, Fort Lauderdale, Florida, Apr. 2002.
9. K. Imasaki, H. Nguyen, and S. Dandamudi. Performance Comparison of Pipelined Hash Joins on Workstation Clusters. In *9th International Conference on High Performance Computing (HiPC2002)*, pages 264–275, Bangalore, India, Dec. 2002.
10. K. Imasaki, H. Nguyen, and S. Dandamudi. Performance of Pipelined Nested Loop and Hash Joins on a Workstation Cluster. In *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, June 2002.
11. D. A. Schneider and D. J. DeWitt. A Performance Evaluation of Four Parallel Join Algorithms in a Shared-nothing Multiprocessor environment. *SIGMOD Record*, 18(2):110–121, June 1989.
12. T. Urhan and M. J. Franklin. XJoin: Getting Fast Answers From Slow and Bursty Networks. Technical Report CS-TR-3994, University of Maryland, College Park, Feb. 1999.

A Flexible Multiagent Parallel File System for Clusters

María S. Pérez[1], Jesús Carretero[2], Félix García[2], José M. Peña[1], and Víctor Robles[1]

[1] DATSI. FI. Universidad Politécnica de Madrid. Spain
[2] Departamento de Informática. Universidad Carlos III de Madrid. Spain

Abstract. The emergence of applications with greater processing and speedup requirements, such as *Grand Challenge Applications* (GCA), involves new computing and I/O needs. Many of these applications require access to huge data repositories and other I/O sources, being the I/O phase a bottleneck in the computing systems, due to its poor performance. In this sense, parallel I/O is becoming one of the major topics in the area of high-performance systems. Existing data-intensive GCA have been used in several domains, such as high energy physics, climate modeling, biology or visualization. The I/O problem is not solved in this kind of applications. New approaches are required in this scene. This paper presents MAPFS, a multiagent architecture, whose goal is to allow applications to access data in a cluster of workstations in an efficient and flexible fashion, providing formalisms for modifying the topology of the storage system, specifying different data access patterns and selecting additional functionalities.

1 Introduction

Nowadays, there is a growing interest in the development of high-performance I/O systems, because the I/O phase has become a bottleneck in the computing systems due to its poor performance. In this sense, one of the major goals of high-performance computing is to provide an efficient access to data, being parallel I/O one of the most relevant issues in this field.

Currently, there are different parallel file systems, such as Galley [11], Parallel File System (PFS) [4] and Portable Parallel File System (PPFS) [8], which offer high-performance services to access resources. Many of these systems have been widely developed for parallel machines and are not suitable for clusters of workstations. Nevertheless, in parallel computing there is an increasing trend towards the usage of clusters, mainly because of their prices and their ease of integration. In this sense, the Parallel Virtual File System (PVFS) [3] is used in dedicated clusters of workstations.

On the other hand, parallel file system optimizations provide improved I/O operations. The usage of hints related to different aspects of data distribution and access patterns allows parallel file systems to increase the performance of these operations. Processes such as caching or prefetching are useful approaches used in addition with these last ones [12], [2]. The **agent** technology [5], [7] is a suitable framework for integrating these functions in the storage system, because of its adaptability to different domains and its capability to achieve process autonomy.

This paper presents MAPFS, a multiagent architecture, whose goal is to allow applications to access data in a cluster of workstations in an efficient and flexible fashion,

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 248–256, 2003.
© Springer-Verlag Berlin Heidelberg 2003

providing formalisms for modifying the topology of the storage system, specifying different data access patterns and selecting additional functionalities.

The outline of this paper is as follows. Section 2 presents MAPFS as a flexible infrastructure for data-intensive applications in a cluster environment. Section 3 shows the results obtained for the evaluation of applications using MAPFS. Section 4 describes a comparison between MAPFS and PVFS. Finally, section 5 summarizes our conclusions and suggests further future work.

2 MAPFS Overview

The emergence of applications with greater I/O access requirements, named data-intensive applications or I/O-intensive applications, demands new I/O solutions. Examples of data-intensive and Grand Challenge applications [6] include data mining systems, data warehousing [9], high energy physics applications [15], and satellite data processing [1]. These applications may require access to data sources distributed among different nodes. Moreover, typical data-intensive applications require access to terabyte size datasets, which must be processed in an efficient way, in order to increase the performance of the applications executed on the cluster. Furthermore, data-intensive applications are very different depending on the kind of functional requirements and access patterns. It is critical for I/O system to be flexible enough to match these demands. The usage of hints, caching and prefetching policies or different data distribution configurations are optional features, which can reduce latency and increase I/O operations performance.

MAPFS [13] is a multiagent architecture, which provides flexibility in different aspects:

- System topology configuration: Ability to change system topology, setting the I/O nodes and their relationships. This feature is achieved through the usage of **storage groups**, which are described in section 2.2.
- Access pattern specification: Although MAPFS is a general purpose I/O system, it can be configured in order to adapt to different I/O access patterns [14].
- There are different reasons to allow some functionalities (such as caching or prefetching) to run in parallel on different nodes and even on the data servers. Moving executions to data servers may reduce network latency and traffic. The agent technology is useful in this scene, because of the agents properties, such as autonomy, proactivity and reactivity. MAPFS is composed of a multiagent subsystem, which is responsible of making several independent tasks.

2.1 MAPFS Architecture

MAPFS is based on a client-server architecture using general purpose servers. In the first prototype, NFS servers are used. NFS [10] has been ported to different operating systems and machine platforms and it is widely used by many servers worldwide. Data is distributed through the servers belonging to a storage group, using a stripe unit.

In the client-side, it is necessary to install a MAPFS client, which provides a parallel I/O interface to the servers.

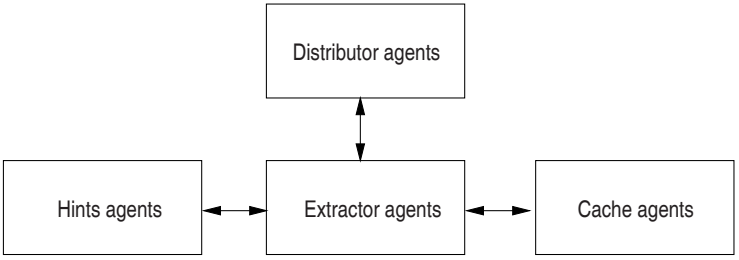


Fig. 1. Taxonomy of agents used in MAPFS

Additional multiagent subsystems provide several functionalities and they are executed on different nodes. Every storage group is associated to a multiagent subsystem. These multiagent subsystems use an agent hierarchy, which solves the information retrieval problem in a transparent and efficient way. The taxonomy of agents used in MAPFS is composed of:

- Extractor agents: They are responsible for information retrieval, invoking parallel I/O operations.
- Distributor agents: They distribute the work load to the extractor agents. These agents are placed at the higher level of the agents hierarchy.
- Caching and prefetching agents: They are associated with one or more extractor agents, caching or prefetching their data.
- Hints agents: They must study applications access patterns to build hints improving data access.

Figure 1 represents the relation among these agents. The taxonomy of agents can be extended to provide additional functionalities.

The most usual configuration is to run these subsystems on the data servers, helping to reduce network traffic. In this case, a major requirement is to install a technology that supports distributed execution of agents. MAPFS provides this optional functionality.

2.2 Storage Groups

The concept of grouping is fundamental in every aspect of the life. Edwin P. Hubble, which is considered the founder of the observational cosmology, said in the thirties that the best place for searching a galaxy is next to another one, describing the concept of galaxy grouping. Like in real life, computer science has a significant number of groupings, e.g. process group or user group.

A *storage group* is defined in MAPFS as a set of servers clustered as groups. These groups take the role of data repositories and can be built applying several policies, trying to optimize the accesses to all the storage groups. Some significant policies are:

- Grouping by server proximity: Storage groups are built based on the physical distribution of the data servers. Storage groups are composed of servers in close proximity to each other. This policy optimizes the queries addressed to a storage group because of the similar latency of messages sent to servers.

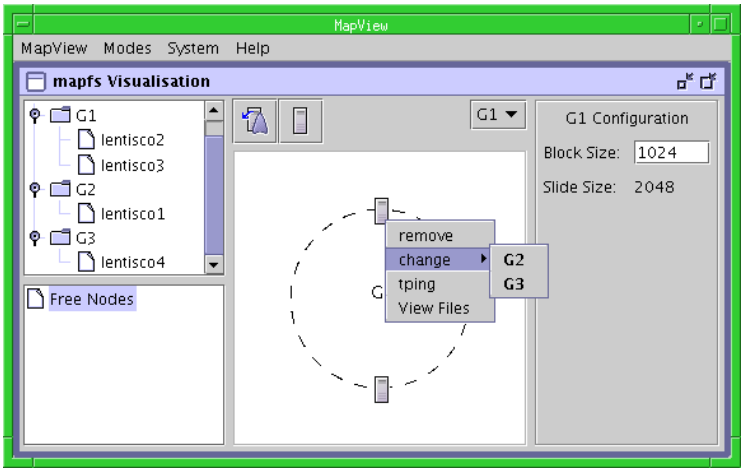


Fig. 2. MAPFS interface showing the system topology

– Grouping by server similarity: Storage groups are composed of servers with similar processing capacity. This policy classifies storage groups in different categories, depending on their computational and I/O power.

The main advantages of storage groups are:

1. Logic abstraction of the concept of storage server: As a partition is a logic abstraction of the physical disk, the storage group is also a logic abstraction of the storage server concept.
2. Dynamic management of servers: As we mentioned previously, the usage of storage groups provide dynamic management of servers through the MAPFS interface (see figure 2).
3. Efficiency of the storage operations: The policies used in the system provide a way of increasing the global efficiency of the system.
4. Load balancing: It is possible to use a concrete storage group in order to optimize system load balancing, depending on the load of the remaining storage groups.
5. Transparent migration: In addition to the MAPFS interface, the system can change the distribution of storage groups in a transparent way in order to optimize different aspects related to the system performance.

The system topology can be changed dynamically. In this case, data must be reconstructed, because the system must map the files stored in the servers of the previous topology to the new configuration. Data reconstruction degrades the performance of the I/O system. In order to avoid data reconstruction, MAPFS defines two types of storage groups, main storage groups and secondary groups, which form a lattice structure between them. Secondary groups are used for storing new data, avoiding the migration of the files data. In fact, this approach postpones data reconstruction until the system runs a de-fragmentation operation, which is used for deleting secondary groups and simplifying the storage system description.

2.3 Access Pattern Specification

There are a huge number of applications using parallel file systems. These applications have very different requirements and data access patterns. Therefore, it is desirable that the underlying file system allows these applications to provide information about the layout of the data used by them. This information is given as hints, which are used for improving read and write operations performance. MAPFS uses these hints to access data. For example, storage systems using hints may provide greater performance because they use this information to decrease cache faults and to prefetch the data most probably used in next executions. In other words, the more information it has been used, the less uncertainty in the future access guesses and, therefore, the better prefetching and caching results.

In MAPFS, hints can be obtained in two ways:

1. Given by the user, that is, the user application provides to I/O system the necessary specifications.
2. Built by the multiagent subsystem. If this option is selected, the multiagent system must analyze the access pattern of the applications in order to build hints improving data access.

If hints are provided by the user application, it is necessary for the system to provide syntactic rules for setting the system parameters, which configure the I/O system. On the other hand, if the multiagent subsystem creates the hints, it is also necessary to store them in a predefined way. In any case, lexical and syntactic rules must be introduced in the system.

The system is configured through several operations, which are independent of the I/O operations, although these last ones use the former operations. The configuration operations are divided in:

- Hints Setting Operations: Operations for establishing the hints of the system. Therefore, they can set and get the values of the different fields of the hints.
- Control User Operations: Higher level operations that can be used directly by the user applications to manage system performance.

Figure 3 shows the three subsets of the Storage System API. As can be seen in the figure, there are three ways of accessing the Hints Setting Module:

1. The I/O operations may ask for hints values and even modify them.
2. The control user operations may modify the hints. This is the normal way for the user applications to interact with hints.
3. To make the system flexible, the Hints Setting Module may be accessed directly through the Hints Setting API. The multiagent system may build and modify hints through this interface.

The MAPFS I/O API was described in [13]. The hint setting operations are the following ones:

- *Mapfs_Hints * mapHintsNew(int block_ident)*: This operation creates a new hint structure for the block with identifier *block_ident*.

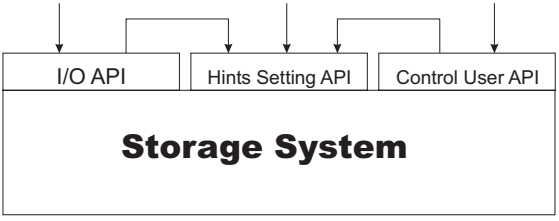


Fig. 3. Storage System API

- *void mapHintsFree(Mapfs_Hints *hint)*: This operation releases a hint structure.
- *int mapHintsSet(Mapfs_Hints *hint, int code_field, void * value)*: This operation modifies a field of the hint structure with a value. The operation returns 0 if successful and -1 otherwise.
- *void * mapHintsGet(Mapfs_Hints *hint, int code_field)*: This operation returns the value of a concrete field of the hint structure. If the field is not defined, *mapHintsGet()* returns NULL.

Hints in MAPFS are built based on the concept of attribute. An attribute can take different values, which provides useful information for the I/O operations. The control user operations have a similar structure:

- *Mapfs_CtrlUser * mapCtrlUserNew(Mapfs_Tuples *tupl)*: This operation creates a new Control User structure for a set of tuples represented by *tupl*.
- *void mapCtrlUserFree(Mapfs_CtrlUser *ctrlUser)*: This operation releases a Control User structure.
- *int mapCtrlUserSet(Mapfs_CtrlUser *ctrlUser, int code_field, void *expr)*: This operation modifies a field of the Control User structure with an expression. The operation returns 0 if successful and -1 otherwise.
- *void *mapCtrlUserGet(Mapfs_CtrlUser *ctrlUser, int code_field)*: This operation returns the value of a concrete field of the Control User structure. If the field is not defined, *mapCtrlUserGet()* returns NULL.

Control User operations are higher level operation using expressions instead of simple attributes, which are translated to hints by the MAPFS system. These expressions are formed by boolean operations applied to attributes. An example is shown in the section 3.

3 MAPFS Evaluation

In order to validate our implementation, it is necessary to evaluate its performance. Experiments were run on a cluster of four nodes Athlon 650MHz, with 256 MB of RAM memory, attached to a Gigabit network. This cluster constitutes our trial storage group.

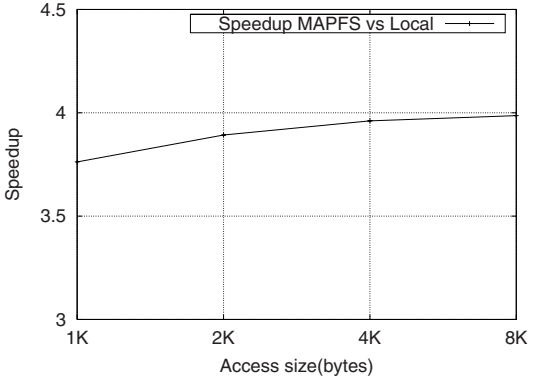


Fig. 4. Speedup of the MAPFS solution on a cluster of 4 nodes versus Local solution

Our experiment consists of four processes running a multiplication of two matrices, where the matrices are stored in the cluster, using MAPFS as underlying platform. The resultant matrix is also stored in a distributed fashion. A prefetching multiagent subsystem is used, which is responsible for prefetching rows and columns of the matrices. In this case, the hints provided by the applications are the indexes of the matrix row and the matrix column of the element calculated in every iteration. It is possible to prefetch data used in later executions, using this information. The control user structure provided by the application follows this expression:

```
row=i AND column=j
```

Therefore, the sequence of configuration operations made by the application are:

1. `Mapfs_CtrlUser *mapCtrlUser = mapCtrlUserNew(data);`
2. `mapCtrlUserSet(mapCtrlUser, position, "row=i AND column=j");`

This control user structure is translated to hints by the multiagent subsystem. Hints are used as data tags for processing the elements in an efficient manner, skipping non-related elements in the cache.

This experiment was compared to another one, which consists in multiplying the same matrices stored on the local disk through the usage of a traditional I/O system. The size of the matrices was 100 MB.

Figure 4 shows the speedup of the MAPFS solution versus local solution, varying the access size used in the I/O operations. As can be seen, the speedup is very close to 4, the number of nodes, which is the maximum speedup, limited by the “Amdahl law”.

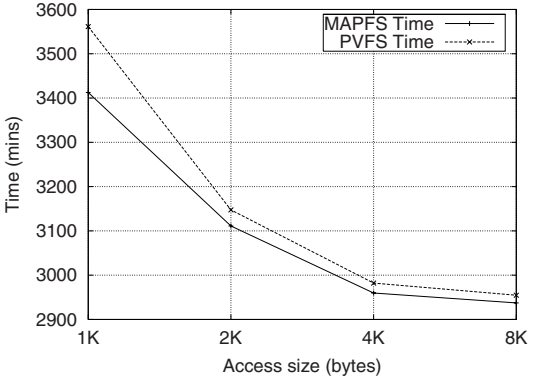


Fig. 5. Execution time of multiplication of two matrices with MAPFS and PVFS

4 Comparison with PVFS

The best way of evaluate MAPFS is to compare the performance of an application using this parallel file system and another parallel file system. PVFS [3] has been chosen due to the following reasons:

1. It is a parallel file system developed for Linux clusters.
2. It provides high bandwidth for concurrent read/write operations from multiple processes or threads to a parallel file.
3. It is robust and scalable.
4. It gives support for multiple APIs, including a native PVFS API, the UNIX/POSIX API and MPI-IO API. We have used the first one for implementing the application.

Again, we have compared a multiplication of two matrices distributed among different nodes both in MAPFS and PVFS. Figure 5 shows the execution time of the application using both parallel file systems. As can be seen in the figure, the difference is significant, although it decreases as the access size is increased. It is important to emphasize the fact that PVFS is a high-performance parallel file system.

5 Conclusions and Future Work

In this work we have presented MAPFS, a new multiagent architecture for high performance I/O in clusters. MAPFS provides the following properties: (i) system topology configuration; (ii) access pattern specifications by applications, and (iii) usage of a multiagent subsystem in order to run specific functionalities, such as caching or prefetching processes.

MAPFS has been compared to both a local solution and PVFS, a high performance parallel file system developed for Linux clusters, concluding that MAPFS provides a resultant speedup very close to the maximum one, improving the results of PVFS.

As future work, it would be interesting to evaluate the performance of the system with other applications, using more complex hints.

References

1. Anurag Acharya, Mustafa Uysal, Robert Bennett, Assaf Mendelson, Michael D. Beynon, Jeff Hollingsworth, Joel Saltz, and Alan Sussman. Tuning the performance of I/O-intensive parallel applications. In *Proceedings of the Fourth ACM Workshop on I/O in Parallel and Distributed Systems*, May 1996.
2. Pei Cao, Edward W. Felten, and Kai Li. Implementation and performance of application-controlled file caching. In *Operating Systems Design and Implementation*, pages 165–177, 1994.
3. P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, October 2000.
4. R. Esser and R. Knecht. Intel Paragon XP/S— architecture and software environment. In *Proceedings of Supercomputer '93, Lecture Notes in Computer Science, Mannheim, Germany*, 1993.
5. Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag*, 1996.
6. Grand Challenging Applications. *http://www.mcs.anl.gov/projects/grand-challenges*.
7. N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems Journal*, 1(1):7–38, 1998.
8. James V. Huber Jr., Christopher L. Elford, Daniel A. Reed, Andrew A. Chien, and David S. Blumenthal. PPFS: A high performance portable parallel file system. In *Proceedings of the 9th ACM International Conference on Supercomputing*, pages 385–394, Jul 1995.
9. R. Kimball. *The Data Warehouse Toolkit*. John Wiley and Sons, Inc., 1996.
10. Network Working Group. *NFS: Network File System Protocol Specification*, March 1989. RFC 1094.
11. Nils Nieuwejaar and David Kotz. The Galley parallel file system. In *Proceedings of the 10th ACM International Conference on Supercomputing*, pages 374–381. ACM Press, May 1996.
12. R. Hugo Patterson, Garth A. Gibson, and M. Satyanarayanan. A status report on research in transparent informed prefetching. *ACM Operating Systems Review*, 27(2):21–34, 1993.
13. María S. Pérez, Félix García, and Jesús Carretero. A new multiagent based architecture for high performance I/O in clusters. In *Proceedings of the 2nd International Workshop on Metacomputing Systems and Applications MSA'2001*, September 2001.
14. María S. Pérez, Ramón A. Pons, Félix García, Jesús Carretero, and Víctor Robles. A proposal for I/O access profiles in parallel data mining algorithms. In *3rd ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, June 2002.
15. A. Shoshani, L. Bernardo, H. Nordberg, D. Rotem, and A. Sim. Storage management for high energy physics applications. *Computing in High Energy Physics*, 1998.

A New Proportional-Share Disk Scheduling Algorithm: Trading-Off I/O Throughput and QoS Guarantee*

Young Jin Nam and Chanik Park

Department of Computer Science and Engineering
Pohang University of Science and Technology/PIRL
Kyungbuk, Republic of Korea
{yjnam,cipark}@postech.ac.kr

Abstract. Proportional-share algorithms are designed to allocate an available resource, such as a network, processor, or disk, for a set of competing applications in proportion to the resource weight allotted to each. While a myriad of proportional-share algorithms were made for network and processor resources, little research work has been conducted on disk resources which exhibit non-linear performance characteristics attributed to disk head movements. This paper proposes a new proportional-share disk scheduling algorithm which accounts for overhead caused by disk head movements and QoS guarantees in an integrated manner. Performance evaluations via simulations reveal that the proposed algorithm improves I/O throughput by 11–19% with only 1–2% QoS deterioration.

1 Introduction

The prevalence of streaming services increases the odds of sharing disk resources. As a result, the traffic control on the disk resource called storage QoS (Quality of Service) has been gaining in significance in order to satisfy the requirements of different applications. It is known that partitioning such disk resources as bandwidth plays a key role in satisfying the different QoS requirements of various types of applications, such as best-effort applications and real-time applications [1]. Despite its importance, research on storage QoS is still in its infancy, having mainly focused on an underlying disk scheduling algorithm.

A disk resource exhibits different characteristics from other resource types, such as processors and networks because high overhead is typically involved in processing I/O requests. Few disk scheduling algorithms proportionally share a disk resource, such as YFQ [2] and Cello [1]. Among them, the YFQ algorithm is based on packet-based fair queuing algorithms in the network, *i.e.*, WFQ [3] to choose a subsequent I/O request to be scheduled and SFQ [4] to maintain

* The authors would like to thank the Ministry of Education of Korea for its financial support toward the Electrical and Computer Engineering Division at POSTECH through its BK21 program. This research was also supported in part by HY-SDR IT Research Center.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 257–266, 2003.
© Springer-Verlag Berlin Heidelberg 2003

a global virtual time. Considering that achieving the ultimate QoS guarantee requires integrated scheduling and management for various resources in an underlying system, such as a processor, network, or disk, the YFQ-like approach employing a packet-based fair queuing algorithm is more desirable. Unfortunately, proportional-share disk scheduling algorithms, while preserving a given QoS feature, inevitably suffer from performance degradation. In order to improve their disk I/O performance, the YFQ algorithm attempts to select I/O requests mainly based on a QoS guarantee, and then attempts to reduce disk overhead by batching to the set of I/O requests selected in the previous phase. After investigating the operations of this scheduling algorithm, we determined that the effectiveness of disk overhead reduction is restricted by separating the operation of I/O request selection from the operation of reducing disk overheads. Moreover, while we can achieve a better I/O performance with a larger batch size, the size of the batch cannot be arbitrarily increased in actual systems.

This paper proposes a new proportional-share disk scheduling algorithm which considers the issues of disk I/O overhead reduction and I/O bandwidth provisioning in an integrated manner, thus overcoming the shortcoming of limited batch sizes found in the YFQ algorithm. Our basic idea derives from the fact that combining the operation of I/O request selection with the operation of reducing disk overheads will increase the odds of reducing disk overheads while maintaining a certain level of QoS guarantee.

2 The Proposed Algorithm

We begin by providing a few notations. Assume that N different I/O workloads exist. The k-th reservation queue for the k-th I/O workload is denoted as $RQ_k = \{r_k^1, r_k^2, ...\}$, where r_k^i refers to the i-th I/O request in the k-th reservation queue. The notation of l_k^i represents the request size of r_k^i. RQ_k requires a different QoS requirement denoted with w_k; that is, a different amount of disk bandwidth. Actual bandwidth alloted to the k-th I/O workload is $\frac{w_k}{\sum_i w_i} B$, where B is the disk bandwidth. As with packet-based fair queuing algorithms [3, 4], each reservation queue RQ_k maintains a virtual start time and a virtual finish time, denoted by S_k and F_k, respectively. Besides, a global virtual time $v(t)$ is maintained.

2.1 Overall Architecture

The proposed algorithm consists of a set of reservation queues (RQ), the base QoS sequence generation (BQS) module, and the disk overhead reduction (DOR) module, as shown in Figure 1. The reservation queues are used by the BQS and DOR modules. The proposed algorithm aims to produce an expanded sequence of I/O requests denoted with $S_{expanded}$, which not only preserves a given QoS feature, but also achieves a better disk I/O performance based on a base sequence of I/O requests denoted with S_{base}, which is generated by the strict QoS-enforcing

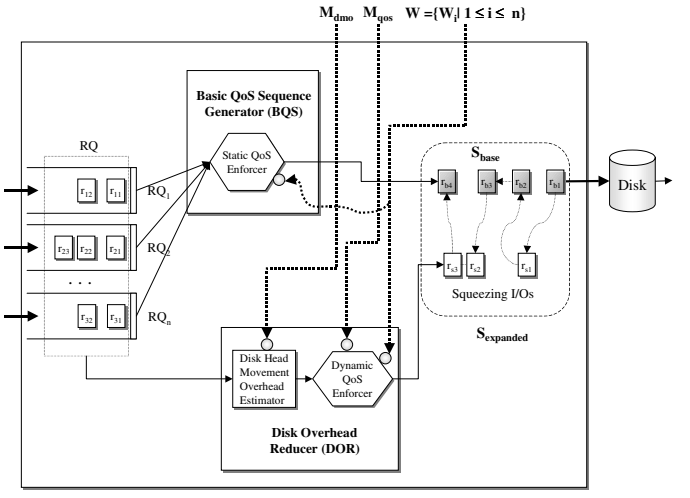


Fig. 1. Architecture of the proposed algorithm

module (the BQS module). Each I/O workload has its own reservation queue called RQ_k to which a corresponding disk bandwidth w_k is allocated.

The BQS module generates a S_{base} from I/O requests in RQ as with the YFQ algorithm. First, it selects an I/O request from the reservation queue of the minimum virtual finish time, say r_s^1 and RQ_s, respectively. Second, it advances $v(t) = S_s$, Third, it updates $S_s = F_s$ and $F_s = S_s + \frac{l_s^{next}}{w_s}$ if RQ_s has a next I/O request, where l_s^{next} is the size of the next I/O request of r_s^1. Otherwise, S_s and F_s remain unchanged. If a new request r_i^k arrives at the empty RQ_i, then $S_i = max\{v(t), F_i\}$ and $F_i = S_i + \frac{l_i^k}{w_i}$. The typical number of I/O requests selected by the BQS module is four. After choosing a number of I/O requests, the BQS module reorders them in a C-SCAN order, thus producing a final sequence of I/O requests called *a base I/O sequence*, S_{base}.

The DOR module generates $S_{expanded}$ by attempting to insert as many I/O requests as possible into S_{base} given by the BQS module when two conditions on the disk idle time and QoS guarantee are met. These two conditions are determined by two controlling parameters, M_{dmo} and M_{qos}. (These two parameters will be explained in the succeeding paragraph.) The DOR module consists of two core components: the Disk Head Movement Overhead Time Estimator (DME) and the Dynamic QoS Enforcer (DQE). The DME module determines whether an extra I/O request in RQ can be squeezed into a $S_{interim}$ by estimating the disk overhead time which exists in the sequence. $S_{interim}$ means an intermediate I/O sequence in transit from S_{base} to $S_{expanded}$. To control the insertion of I/O requests into the current I/O sequence, we devise a controlling parameter called an overhead margin M_{dmo}, which will eventually be added to the estimated disk overhead time between the two I/O requests in $S_{interim}$. As M_{dmo} becomes larger, a chance increases that an extra I/O request can be squeezed into the

sequence. The DQE module controls the degree of QoS enforcement against the I/O requests which have passed the DME module successfully. Aside from the start/finish virtual times at each queue, this module additionally maintains a separate set of virtual times for each RQ_k and a displacement between the maximum and minimum virtual times among the non-empty reservation queues. A virtual time can advance as long as the displacement is not greater than the QoS margin denoted with M_{qos}. We will define two groups of I/O requests: the BQS_IO and the DOR_IO. The BQS_IO represents a group of I/O requests which pass through the BQS module, and the DOR_IO means a group of I/O requests squeezed into a base I/O sequence S_{base} via the DOR module. Obviously, all I/O requests of the BQS_IO can commit a given QoS guarantee.

2.2 M_{dmo}, M_{qos} Control and Algorithm Description

We will explore in detail the behavior of the DME and DQE modules with the two controlling parameters: M_{dmo} and M_{qos}. M_{dmo} determines a margin for a disk overhead time which exists between two I/O requests in $S_{interim}$, and M_{qos} decides how rigorously a given QoS feature is to be applied to the I/O requests which have been chosen by the DME module. The size and I/O characteristics of a $S_{expanded}$ depend heavily on the M_{dmo} and M_{qos} values.

To take control of inserting I/O requests to the current I/O sequence, we introduce M_{dmo} to be added to the estimated disk overhead time between the two I/O requests in $S_{interim}$. Property 1 clarifies the meaning of the marginal overhead time M_{dmo}. $T_{ov}(r_i, r_j)$ means the disk overhead time from request r_i to r_j and the disk $S_{interim}$ represents an I/O sequence in transit from S_{base} to $S_{expanded}$.

Property 1. r_k can be insertable between r_i and r_j in $S_{interim}$, if $T_{ov}(r_i, r_k) + T_{ov}(r_k, r_j) \leq T_{ov}(r_i, r_j) + M_{dmo}$.

The $M_{dmo} = p$ implies that its overhead margin is equal to the p percentage of the full seek time of an underlying disk. Even though the disk overhead between two I/O requests can be defined in several ways, we simply use a seek overhead between r_i and r_j. The information on the seek times of an underlying disk can be extracted from host [5]. According to Property 1, we expect that a length of $S_{expanded}$ will be longer in proportion to the value of M_{dmo}. However, recall that a length of $S_{expanded}$ is also affected by M_{qos}.

Next, we maintain another QoS enforcement scheme that consists of a set of virtual times for each RQ_k and a current virtual time displacement, in order to enforce a given QoS feature on I/O requests which have been successfully passed by the DME module. The purpose of enforcing a given QoS is to keep a given displacement controlled by the M_{qos} parameter for the virtual times of the non-empty reservation queues. Denote the additional virtual time of RQ_k with V_k. Initially, $V_k = 0$. If an I/O request arrives at an empty RQ_k, $V_k = max\{V_k, min_{\{j,j \neq k\}}\{V_j\}\}$, for $RQ_j \neq \emptyset$. The displacement among different virtual times is written as $\delta_V = max_i\{V_i\} - min_j\{V_j\}$ for the non-empty reservation queues.

Property 2. For a $r_k^i \in RQ_k$, r_k^i can preserve a given QoS feature if it meets: $\delta_V \leq M_{qos}$ when $V_k = V_k + \frac{l_k^i}{w_k}$.

$M_{qos} = q$ implies that a set of virtual times will advance with the following amount of QoS deterioration, such that the resulting I/O throughput of each RQ_k will deviate from a given QoS feature on the average by $\frac{w_k}{l_k^{avg}} q$ I/O requests, where l_k^{avg} is an average block size of an I/O request from RQ_k. According to Property 2, no I/O requests can pass the DOR module with $M_{qos} = 0$, since no displacement is allowed. It implies that a given QoS feature is preserved strictly with the BQS module.

Algorithm Description: Algorithm 1 describes how the DOR module operates. Recall that the BQS module strictly enforces a given QoS feature in exactly the same manner as the YFQ algorithm.

Algorithm 1: Reducing Disk Overhead

```
Data   : S_base = {r_1, r_2, ...r_n}
Result : S_expanded = S_base + DOR_IO requests
begin
    S_interim ← S_base ;
    r_cur ← r_head ← r_1; r_tail ← r_n ;
    while r_cur ≠ r_tail do
        for each r_k^i in RQ_k ∈ RQ do
            if r_k^i meets Property 1 and 2 then
                insert r_k^i between r_cur & r_next in S_interim ;
                V_k = V_k + l_k^i / w_k;
                break ;
            end
        end
        r_cur ← next I/O request of r_cur in S_interim ;
    end
end
```

3 Performance Evaluations

This section investigates the performance of the proposed algorithm by discovering two desirable values of M_{dmo} and M_{qos} which not only improve disk I/O performance, but also preserve a given QoS feature with negligible deterioration.

3.1 Simulation Environments

We implemented the proposed algorithm as one of driver-specific disk scheduling schedulers within the DiskSim simulator [6]. Two competing I/O workloads are

synthetically generated. The sizes of the I/O requests are distributed normally with a mean of 8 blocks. The ratio of reads to writes is set to two. An I/O workload level becomes heavier by increasing the number of outstanding I/O requests denoted with $|IO|$. A start block address of each I/O request is distributed randomly over the entire area of a IBM DNES 309170W SCSI disk which serves arriving I/O requests in a FIFO manner with a maximum of four outstanding concurrent I/O requests. It is assumed that two reservation queues exist, which are RQ_1 and RQ_2. $w_1 = 80$ and $w_2 = 20$. With the QoS requirements, the BQS module selects four I/O requests from the head of the reservation queues and then generates S_{base} where the I/O requests are reordered in a C-SCAN order. Next, the DOR module generates $S_{expanded}$ by expanding a given S_{base} with a group of squeezing I/O requests. As noted, two controlling parameters exist, which are an overhead margin of M_{dmo} for the DME module and a QoS margin of M_{qos} for the DQE module. A combination of these two parameters determines the length of a $S_{expanded}$ which will eventually characterize the behavior of the DOR module.

3.2 Performance Results

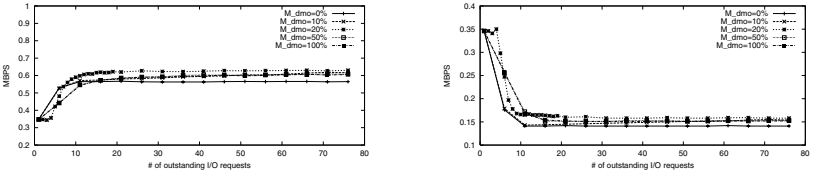
To locate two desirable controlling parameters for a given disk, we take the following two steps. First, while strictly maintaining a given QoS feature, we seek a M_{dmo}^{best} which gives the most proper adjustment to the disk overhead so that it can maximize disk I/O performance. Second, by relaxing the degree of QoS enforcement with the M_{dmo}^{best}, we attempt to obtain an M_{qos}^{best} which provides a better disk I/O performance with acceptable QoS deterioration. Finally, the proposed algorithm with M_{dmo}^{best} and M_{qos}^{best} will be compared with the YFQ algorithm in terms of the disk I/O performance and QoS guarantee.

Finding a M_{dmo}^{best} controlling parameter: Recall that the M_{dmo} was invented to control the slack time for overhead associated with disk head movement between two I/O requests. Given a fixed M_{qos}, a larger M_{dmo} will increase the number of I/O requests squeezed into S_{base} to exploit the estimated overhead of disk head movements. While no I/O requests are allowed to pass through the DOR module with $M_{dmo} = 0$, most I/O requests within RQ bypass the DOR module with $M_{dmo} = 100$ whose margin is equal to a disk full seek time. Figure 2 shows the I/O throughput of the proposed algorithm with a different M_{dmo} under various I/O workload levels where $M_{qos} = 0.5$[1]. We expect that the given QoS feature will be preserved throughout all I/O workload levels because of $M_{qos} = 0.5$, which corresponds to a strong QoS enforcement. Based on the performance results in Figure 2, the best I/O throughput can be obtained when $M_{dmo} = 20$.

To validate the obtained M_{dmo}, we investigate the variation of I/O throughputs as a function of M_{dmo} under a fixed I/O workload level, as shown in Figure 3.

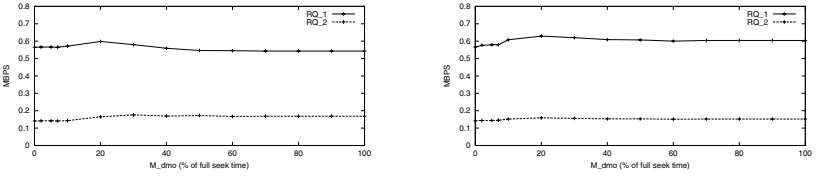
[1] By definition in Property 2, $M_{qos} = 0.5$ corresponds to a marginal virtual time in the DOR module for either five(5) I/O requests from RQ_1 or about a single I/O requests from RQ_2 on the average.

As with the previous results, the maximum I/O throughput can be achieved when $M_{dmo} = 20$. Observe that a large M_{dmo} close to 100 cannot improve the I/O throughput due to improper overhead estimations. Conversely, a small M_{dmo} decreases the probability that the remaining I/O requests within RQ can utilize the existing disk overhead time. A length of $S_{expanded}$ is saturated to about 8



(a) I/O throughput of RQ_1 (b) I/O throughput of RQ_2

Fig. 2. Variations of I/O throughputs of RQ_1 and RQ_2 as a function of an I/O workload level by varying the number of outstanding I/O requests ($|IO|$) under different values of M_{dmo}, where $M_{qos} = 0.5$



(a) $|IO| = 11$ (b) $|IO| = 61$

Fig. 3. Variations of I/O throughputs as a function of M_{dmo} when $|IO| = 11$ and $|IO| = 61$, where $M_{qos} = 0.5$

when M_{dmo} becomes 20. Finally, we conclude the first performance evaluation with the observation that M_{dmo}^{best} of 20 maximizes the I/O throughput under the condition of a strict QoS enforcement.

Finding a M_{qos}^{best} controlling parameter: We can achieve an additional performance enhancement by relaxing the degree of QoS enforcement. Figure 4 depicts the variations of I/O throughputs under different M_{qos} values, where the M_{dmo} is set to 20. The resulting performance of $M_{qos} \geq 5$ is higher than that of $M_{qos} = 0.5$ by over 10% in terms of the aggregate I/O throughputs of RQ_1 and RQ_2. When considering a further improvement in I/O throughputs by increasing M_{qos}, we have to consider the corresponding QoS deterioration. We will examine the variation of I/O throughput at each reservation queue as a function of M_{qos} under fixed I/O workload levels, $|IO| = 11$ and $|IO| = 61$, as shown

in Figures 5–6. Figure 5 shows that the I/O throughput of the RQ_1 decreases, whereas the I/O throughput of its competing I/O workload RQ_2 improves by increasing M_{qos}.

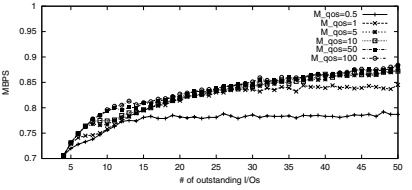
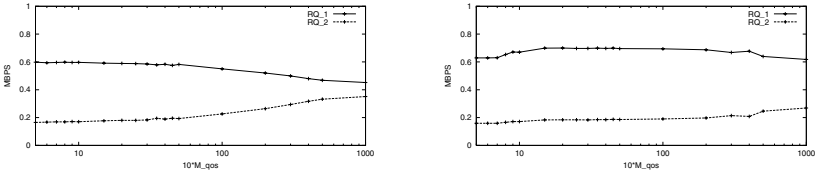


Fig. 4. Variations of I/O throughputs as a function of an I/O workload level under different M_{qos} values

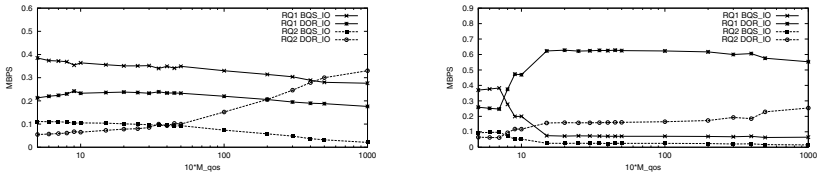


(a) $|IO| = 11$ (b) $|IO| = 61$

Fig. 5. Variations of I/O throughputs as a function of M_{qos} under fixed I/O workload levels, $|IO| = 11$ and $|IO| = 61$, where $M_{dmo} = 20$

Figure 6(a)–(b) present the BQS_IO and DOR_IO throughputs which constitute the aggregate I/O throughputs in Figure 5. As M_{qos} increases with $M_{qos} = 20$, the resulting I/O throughput of each reservation queue becomes dominated by the DOR_IO, not the BQS_IO. Note that the BQS_IO consistently preserves the given QoS feature at any level of the I/O workload. However, the resulting I/O throughput of the BQS_IO is diminished as the DOR_IO becomes dominant.

Figure 7 presents that M_{qos} of 0.5 strongly enforces the given QoS feature to its DOR_IO requests, so that it can provision disk bandwidth with a given ratio of 4:1 between RQ_1 and RQ_2. However, as M_{qos} becomes larger, the desirable ratio deteriorates, *i.e.*, the I/O throughput of RQ_1 is mainly affected by RQ_2, which uses more disk bandwidth with a relaxed QoS enforcement. As a result, the given QoS feature deteriorates by 1% with $M_{qos} = 0.5$, by 2% with $M_{qos} = 1$, and by 5% with $M_{qos} = 5$. It is difficult to precisely determine the metric for the level of QoS satisfaction. In this paper, however, we simply define a ratio of reservation weights as our QoS metric. Using this QoS metric, we have to establish an acceptable range of the QoS satisfaction for a given QoS feature. We believe that satisfying a given QoS feature with 98-100% accuracy is reasonable



(a) $|IO| = 11$ (b) $|IO| = 61$

Fig. 6. Variations of BQS_IO and DOR_IO throughputs as a function of M_{qos} under fixed I/O workloads, $|IO| = 11$ and $|IO| = 61$, where $M_{dmo} = 20$

enough. In that sense, we can say that the QoS margin of $0.5 \leq M_{qos}^{best} \leq 1$ falls into such a reasonable range.

Analyzing the degree of QoS guarantee and I/O throughput with the M_{dmo}^{best} and M_{qos}^{best}: We have found $M_{dmo}^{best} = 20$ and $0.5 \leq M_{qos}^{best} \leq 1$ maximize the disk I/O performance with negligible QoS deterioration. Hereafter, we compare the proposed algorithm with the baseline algorithm (YFQ) assuring QoS guarantee in terms of the degree of QoS enforcement and I/O throughput. Figure 8(a) shows the degree of QoS enforcement for the given QoS ratio of 80:20 as a function of the number of outstanding I/Os. As expected, the proposed algorithm with $M_{qos} = 0.5$ and $M_{qos} = 1$ maintained the given QoS ratio with more than 98% accuracy. While the baseline algorithm having $M_{qos} = 0$ provides the given 4:1 ratio properly, the $M_{qos} = 0.5$ and $M_{qos} = 1$ end up with 3.99:1 and 3.96:1, respectively. Figure 8(b) compares the maximum I/O throughput of the proposed algorithm with the M_{dmo}^{best} and M_{qos}^{best} with the case of $M_{qos} = 0$ (YFQ). Observe that the proposed algorithm with $M_{qos} = 0.5$ and $M_{qos} = 1.0$ improves the I/O throughputs by 11–19% with only 1–2% QoS deterioration.

4 Conclusion and Future Work

We proposed a new proportional-share disk scheduling algorithm that considers both disk characteristics and QoS guarantees in an integrated manner. With extensive simulations, we have discovered two desirable controlling parameters for a given QoS feature. First, given a strong QoS enforcement with $M_{qos} = 0.5$, we observed that the proposed algorithm with $M_{dmo} = 20$ achieved the best performance. Second, given the $M_{dmo} = 20$, we learned that a better disk I/O performance is achievable with less than 2% QoS deterioration by increasing $M_{qos} \leq 1$. Obviously, a serious QoS deterioration is observed as M_{qos} increases continuously. Finally, comparing the proposed algorithm with the baseline algorithm (YFQ) strictly providing QoS guarantee with $M_{dmo} = 0$ and $M_{qos} = 0$, revealed that the proposed algorithm with $M_{dmo} = 20\%$ of the disk full seek time and $0.5 \leq M_{qos} \leq 1$ improved I/O throughput by 11–19% with only 1–2% QoS

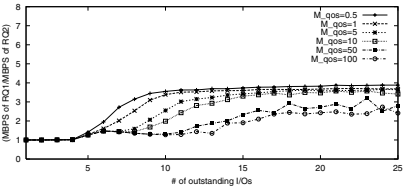
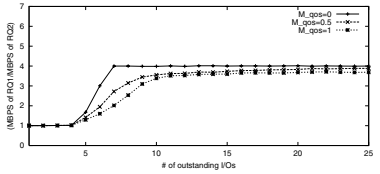
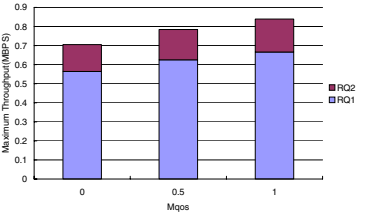


Fig. 7. Variations of QoS guarantees as a function of M_{qos}



(a) QoS guarantees



(b) Maximum throughput

Fig. 8. Variation of QoS guarantees and the maximum I/O throughput according to M_{qos} with $M_{dmo} = 20$: $M_{qos} = 0$(YFQ) and $M_{qos}^{best} = \{0.5, 1.0\}$

deterioration. In future work, we need to define a QoS metric which can measure the degree of QoS enforcement and deterioration more accurately. Besides, we need to devise a technique to adjust the two controlling parameters to maximize underlying disk I/O throughputs with tolerable QoS deterioration for any given I/O workload including traced real I/O workloads.

References

1. Shenoy, P., Vin, H.: Cello: A disk scheduling framework for next-generation operating systems. In: Proceedings of ACM SIGMETRICS. (1998)
2. Bruno, J., *et. al.*: Disk scheduling with quality of service guarantees. In: Proceedings of the IEEE International Conference on Multimedia Computing and Systems. (1999)
3. Parekh, A., Gallager, R.: A generalized processor sharing approach to flow control in integrated services networks: The single-node case. IEEE/ACM Trans. on Networking **1** (1993) 344–357
4. Goyal, P., Vin, H., Cheng, H.: Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. IEEE Transactions on Networking **5** (1997) 690–704
5. Schindler, J., Ganger, G.: Automated disk drive characterization. Technical Report CMU-CS-99-176, CMU (1999)
6. Ganger, G., Worthington, B., Patt, Y.: The DiskSim Simulation Environment Version 2.0 Reference Manual. CMU. (1999)

Computational Aspects of Chemical Data Assimilation into Atmospheric Models

Gregory R. Carmichael[1], Dacian N. Daescu[2], Adrian Sandu[3], and Tianfeng Chai[1]

[1] Center for Global and Regional Environmental Research, 204 IATL,The University of Iowa, Iowa City, IA 52242-1297. gcarmich@cgrer.uiowa.edu
Phone: 319-335-3333. Fax: 319-335-3337.
[2] Institute for Mathematics and its Applications, University of Minnesota, 400 Lind Hall 207 Church Street S.E. Minneapolis, MN 55455. daescu@ima.umn.edu
[3] Department of Computer Science, Michigan Technological University, 1400 Townsend Drive, Houghton, MI 49931. asandu@mtu.edu

Abstract. The task of providing an optimal analysis of the state of the atmosphere requires the development of novel computational tools that facilitate an efficient integration of observational data into models. In this paper we discuss some of the computational tools developed for the assimilation of chemical data into atmospheric models. We perform a theoretical analysis of discrete and continuous adjoints for stiff differential equation solvers. Software tools particularly tailored for direct and adjoint sensitivity analysis of chemical systems are presented. The adjoint of the state-of-the-art model STEM-III is discussed, together with ozone assimilation results for a realistic test problem.
Keywords: Data assimilation, Chemical transport models, Adjoint modeling.

1 Introduction

Our ability to anticipate and manage changes in the atmospheric pollutant concentrations relies on an accurate representation of the chemical state of the atmosphere. As our fundamental understanding of atmospheric chemistry has significantly advanced, novel computational tools are needed to integrate observational data and models with the goal of providing an optimal analysis state of the atmosphere. By optimal analysis state we mean an intimate and close integration of modeled and measured quantities, with the two merged together to provide the best estimate, physically consistent, of the evolving chemical state of the atmosphere. The analysis state better defines the spatial and temporal fields of key chemical components in relation to their sources and sinks. This information is critical in designing cost-effective emission control strategies for improved air quality, for the interpretation of observational data such as those obtained during intensive field campaigns, and to the execution of air-quality forecasting.

In this paper we present some of the computational tools developed for chemical data assimilation into atmospheric transport models. The paper is organized

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 269–278, 2003.
© Springer-Verlag Berlin Heidelberg 2003

as follows. In Section 2 we review the adjoint sensitivity analysis technique and in Section 3 we discuss the derivation of the adjoint model associated with one step stiff numerical integrators. Software tools that enable an efficient implementation are presented in Section 4. Results from a 3-dimensional assimilation test problem are shown in Section 5, and in Section 6 we draw conclusions and pinpoint future work.

2 Adjoint Sensitivity Analysis

Consider the adjoint sensitivity analysis of stiff nonlinear ordinary differential equations arising in the mathematical formulation of the chemical reactions mechanisms

$$\frac{dy}{dt} = f(t,y) \ , \qquad y\left(t^0\right) = y^0 \ , \qquad t^0 \le t \le t^{\mathrm{F}} \ . \tag{1}$$

where $y(t) \in \Re^n$ represents the vector of concentrations of the chemical species in the model. To simplify the presentation, in this paper we consider as parameters the initial state y^0 of the model; it is known that this does not restrict the generality of the formulation. We will refer to the dynamical model (1) as a forward (direct) model; its solution $y = y(t, y^0)$ is uniquely determined once the vector of model parameters y^0 is specified.

Data assimilation applications require the sensitivities (gradient) of a scalar response function $g = g\big(y(t^{\mathrm{F}})\big)$ with respect to the model parameters $y_1^0 \ldots y_n^0$. When the number of parameters is large, the adjoint method provides an efficient alternative to the direct decoupled method [7] for evaluating these sensitivities. Mathematical foundations of the adjoint sensitivity for nonlinear dynamical systems are presented by Cacuci [2,3] and Marchuk et al. [13,14]. Specific applications of adjoint modeling to atmospheric chemical data assimilation are described by Menut et al. [12], Vautard et al. [16], Wang et al. [20], and Elbern et. al. [8].

In the adjoint sensitivity analysis one distinguishes between the continuous and the discrete adjoint modeling, see Sirkes [19].

2.1 Continuous Adjoint Sensitivity

The continuous adjoint model is obtained from the forward model using the linearization technique [2]. Define the "adjoint variable" $\lambda(t) \in \Re^n$ as the solution of the adjoint problem

$$\frac{d\lambda}{dt} = -J^T(t,y)\lambda \ , \quad \lambda(t^{\mathrm{F}}) = \nabla_y g(y(t^{\mathrm{F}})) \tag{2}$$

The continuous method consists in applying a numerical discretization scheme to integrate (2) backwards in time; the sensitivity values are

$$\nabla_{y^0} g = \lambda(t^0) \ . \tag{3}$$

2.2 Discrete Adjoint Sensitivity

The discretization of the system (1) with a selected numerical method results in the discrete forward model, which obtains a numerical approximation $y^N \approx y(t^{\mathrm{F}})$ through a sequence of N intermediate states

$$y^{i+1} = F^i(y^i) \ , \quad i = 0, \ldots, N-1 \ , \tag{4}$$

where F^i represents a one-step numerical integration formula which advances the solution from t^i to t^{i+1}. To obtain the adjoint sensitivity of the numerical solution we evaluate the adjoint variables at $t^i, i = N-1, \ldots, 1, 0$ using the recursive relations

$$\lambda^N = \nabla_y g(y^N) \ , \quad \lambda^i = F_y^i(y^i, p^i)^T \lambda^{i+1} \ . \tag{5}$$

The discrete functional sensitivities are

$$\nabla_{y^0} g(y^N) = \lambda^0 \ . \tag{6}$$

3 Stiff Numerical Methods and Their Adjoints

Here we consider one-step numerical methods for the integration of stiff ordinary differential equations and discuss the corresponding discrete adjoints and their consistency with the continuous adjoint equation. This analysis is relevant for atmospheric data assimilation in the context of modeling stiff chemical processes.

3.1 Runge-Kutta Methods

Consider the s-stage Runge Kutta method [10, 11]

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j f(z_j) \ , \quad z_i = y_n + h \sum_{j=1}^{s} a_{ij} f(z_j) \ . \tag{7}$$

The Discrete Adjoint. In what follows we will use the notation

$$L_j = h \, J^T(t_n + c_j h, z_j) \approx \left. J(t, y(t)) \right|_{t=t_n + c_j h} \ , \quad j = 1 \cdots s \ ,$$

The discrete adjoint of the Runge-Kutta scheme reads [9]

$$\lambda_n = \lambda_{n+1} + \sum_{j=1}^{s} \theta_j \ , \quad \theta_i = L_i \left[b_i \lambda_{n+1} + \sum_{j=1}^{s} a_{j,i} \theta_j \right] \ , . \tag{8}$$

The (backwards) propagation of the discrete adjoints defines the discrete adjoint transfer function

$$\lambda_n = \left(\frac{\partial y_{n+1}}{\partial y_n} \right)^T \lambda_{n+1} = R_D \, \lambda_{n+1} \ . \tag{9}$$

The Continuous Adjoint. Consider the adjoint equation (2) and discretize it directly using a Runge-Kutta method (7) with coefficients $\tilde{a}_{ij}$, $\tilde{b}_i$, $\tilde{c}_j$. We denote

$$M_i = h\, J^T(t, y(t))\big|_{t=t_{n+1}-\tilde{c}_i h}$$

The method is applied backwards in time (from t^{n+1} to t^n)

$$\lambda_n = \lambda_{n+1} + \sum_{j=1}^{s} \tilde{b}_j M_j \cdot w_j \ , \quad w_i = \lambda_{n+1} + \sum_{j=1}^{s} \tilde{a}_{ij} M_j \cdot w_j \ . \tag{10}$$

This (backwards) propagation of the discretized adjoints defines the continuous adjoint transfer function

$$\lambda_n = R_C \ \lambda_{n+1} \tag{11}$$

Comparison of the two approaches. Hager [9] noticed that, if $L_i = M_i$ and $b_i \neq 0$, the discrete adjoint is equivalent to a discretization of the continuous adjoint equation by a Runge Kutta scheme with

$$\tilde{a}_{ij} = \frac{a_{s+1-j,s+1-i}\, b_{s+1-j}}{b_{s+1-i}} \ , \quad \tilde{b}_i = b_{s+1-i} \ , \quad \tilde{c}_i = 1 - c_{s+1-i} \ . \tag{12}$$

Order Conditions.

Theorem 1. *Consider a Runge-Kutta method (7) consistent of order p and its discrete adjoint (9). The relation (9) is a numerical discretization of the continuous adjoint equation (2), which is consistent of order p.*

Proof. The proof is given in [18]. Based on the order condition theory for Runge Kutta methods [10, Section II.2] one expresses the entries of the discrete and the continuous transfer functions in terms of elementary differentials, and establishes that the order conditions for the discrete adjoint are the same as for the original Runge Kutta method.

Control Problems. Control Problems use a formulation where the function f depends on the adjoint $f(y, \lambda)$ (due to its dependency of the control variable). In [18] we analyze the consistency of the discrete adjoint (9) with (2) and generalize the findings of Hager [9].

If all $b_i \neq 0$ we can rewrite the discrete adjoint relation as (see [18])

$$\lambda_{n+1} = \lambda_n + h \sum_{j=1}^{s} b_j \ell_j \ , \quad \ell_i = -J^T(z_i) \left[\lambda_n + h \sum_{j=1}^{s} \bar{a}_{i,j} \ell_j \right] \ . \tag{13}$$

where $\bar{a}_{i,j} = b_j - (a_{j,i} b_j)/b_i$.

Consider now the forward model (1) in tandem with its adjoint (2) as a partitioned ordinary differential equation and solve it using the partitioned RK method given by (7) and (13).

To obtain the order of the partitioned method we follow the P-tree theory of [10, p. 309]. We associate meagre vertices with y and fat vertices with λ. For control problems we recover the extra conditions of Hager [9]. For example, the extra condition for order 3 corresponds to the following P-tree $\bullet - - \circ - - \bullet$; this P-tree has a vanishing differential if f does not depend on the adjoint, $f = f(y)$. Obtaining higher order conditions of consistency is a tedious but straightforward process.

Singular Perturbation Analysis. In [18] we analyze the behavior of adjoints in the stiff case by considering the singularly perturbed problem

$$y' = f(y,z) \ , \quad \epsilon z' = h(y,z) \ , \quad t^0 \le t \le t^F \tag{14}$$

with the sub-Jacobian h_z assumed nonsingular. This model allows us to distinguish between the derivatives of the functional $g(y(t^F), z(t^F))$ with respect to non-stiff and stiff variables

$$\lambda(t) = \frac{\partial g(y(t^F), z(t^F))}{\partial y(t)} \ , \quad \mu(t) = \frac{\partial g(y(t^F), z(t^F))}{\partial z(t)} \ .$$

We consider ϵ-expansions of the adjoint variables $\lambda = \lambda^0 + \epsilon\lambda^1 + \ldots$ and $\mu = \mu^0 + \epsilon\mu^1 + \ldots$ The first terms of these expansions satisfy

$$(\lambda^0)' = \left(-f_y^T + h_y^T h_z^{-T} f_z^T\right) \lambda^0 \quad \text{and} \quad \mu^0 = 0 \ . \tag{15}$$

Consider a Runge Kutta method with invertible coefficient matrix A and with the stability function satisfying $R(\infty) = 0$. In [18] it is shown that, if the cost function depends only on the non-stiff variable y (i.e. we initialize $\mu_N^0 = 0$) and $R(\infty) = 0$ then $\mu_n^0 = 0$ for all n. In this case the discrete adjoint is a consistent discretization of the reduced equation (15), and the values of λ are solved with the same accuracy as the original method, within $\mathcal{O}(\epsilon)$. A similar conclusion holds for continuous Runge Kutta adjoints.

3.2 Rosenbrock Methods

Consider an autonomous Rosenbrock method in the implementation-friendly form

$$y_{n+1} = y_n + \sum_{j=1}^{s} m_j k_j \ , \quad Y_i = y_n + \sum_{j=1}^{i-1} a_{i,j} k_j \ , \tag{16}$$

$$\left(\frac{1}{h\gamma} - J(y_n)\right) k_i = f\left(Y_i\right) + \sum_{j=1}^{i-1} \frac{c_{i,j}}{h} k_j \ , \quad i = 1, \cdots, s \ . \tag{17}$$

The Discrete Adjoint In [18] we show that the discrete adjoint of the Rosenbrock method (17) reads

$$\left(\frac{1}{h\gamma} - J^T(y_n)\right) u_i = m_i\lambda_{n+1} + \sum_{j=i+1}^{s}\left(a_{j,i}v_j + \frac{c_{j,i}}{h}u_j\right) , \tag{18}$$

$$v_i = J^T(Y_i)u_i , \quad i = s, s-1, \cdots, 1 , \tag{19}$$

$$\lambda_n = \lambda_{n+1} + \sum_{i=1}^{s}\left(H(y_n) \times k_i\right)^T \cdot u_i + \sum_{i=1}^{s} v_i \tag{20}$$

Here J denotes the Jacobian and H (a 3-tensor) is the Hessian of the derivative function f. The formulation can be easily extended to non-autonomous systems.

The Continuous Adjoint The continuous adjoint is obtained by solving (2) with the Rosenbrock method (17).

4 Software Tools for Adjoint Sensitivity of Chemical Kinetic Systems

In this section we present the Kinetic PreProcessor KPP software tools that are useful in derivative computations. A detailed discussion of the basic KPP capabilities can be found in our previous work [6]. Here we focus on the new features introduced in the release 1.2 [17, 5] that allow a quick and efficient implementation of the code for sensitivity analysis of chemical kinetic systems.

KPP builds Fortran or C simulation code for chemical systems with chemical concentrations changing in time according to the law of mass action kinetics. KPP generates the following building blocks:

1. *FunVar*: the time derivative of concentrations;
2. *JacVar, JacVar_SP*: Jacobian of FunVar in full or in sparse format;
3. *KppDecomp*: sparse LU decomposition for the Jacobian;
4. *KppSolve, KppSolveTR*: solve sparse system with the Jacobian matrix and its transpose;
5. *JacVar_SP_Vec, JacVarTR_SP_Vec*: sparse Jacobian (transposed or not) times vector;
6. The stoichiometric matrix STOICM;
7. *ReactantProd*: vector of reaction rates;
8. *JacVarReactantProd*: the Jacobian of the above;
9. *dFunVar_dRcoeff*: derivatives of FunVar with respect to reaction coefficients (in sparse format);
10. *dJacVar_dRcoeff*: derivatives of JacVar with respect to reaction coefficients times user vector;
11. *HessVar*: the Hessian of FunVar; this 3-tensor is represented in sparse format;

12. *HessVar_Vec, HessVarTR_Vec*: Hessian (or its transpose) times user vectors; same as the derivative of Jacobian (transposed) vector product times vector.

In [17, 5] we show how the KPP building blocks can be used to implement very efficiently code for direct and adjoint sensitivity analysis of chemical systems.

5 Numerical Results with Adjoint STEM-III

We now present assimilation results for a problem being analyzed in support of large field experiments conducted in East Asia (i.e. TraceP and AceAsia experiments). The simulated region is East Asia, the simulated interval is 6 hours starting at 0 GMT on March 1st, 2001. The meteorological fields are given by a dynamic meteorological model (RAMS), and the initial fields and boundary conditions correspond to Trace-P data campaign. The grid is $90 \times 60 \times 16$ points and has a horizontal resolution of $80Km \times 80Km$.

The adjoint STEM-III code has the following characteristics:

- The transport is computed using an upwind third order finite difference scheme for advection, and a second order centered scheme for diffusion, with Crank-Nicholson time integration. The transport operator is linear and the implementation of its adjoint straightforward;
- The gas phase chemical mechanism is SAPRC-99 [4] which considers the gas-phase atmospheric reactions of volatile organic (VOCs) and nitrogen oxides (NOx) in urban and regional settings. The forward time integration is done with the Rosenbrock numerical integrator Ros-2; the continuous adjoint model uses Ros-2 on the same sequence of steps as the forward chemical integration. Both the forward and the adjoint models are implemented using KPP.
- The forward and adjoint models are parallel and were run on a cluster of Linux workstations. Parallelization is based on our library PAQMSG [15].
- The checkpoints store the concentration fields every 15 minutes, i.e. for every operator split step. The checkpoints are distributed as seen in Figure 2, which means that each node stores local information on the local disk. This increases the storage capabilities and decreases the input/output overhead when the parallel computation runs on a cluster of workstations.
- On 16 nodes (Pentium, 2GHz, 1GB RAM) the cpu time for a forward run is about 2 minutes per hour of simulation; and the cpu time for a forward-backward run is about 4.5 minutes per hour of simulation.

The data assimilation procedure is set using the twin experiments method as follows:

Reference run: we start a model run at $t^s = 0{:}00GMT$ with the standard concentrations of all variable chemical species.

Initial guess run: the experiment is repeated with ozone initial concentrations increased by 20%.

Observations and assimilation window: We consider a 6 hours assimilation window. The observations for O_3 concentrations on all grid points are provided only at the end of the assimilation window $t^0 + 6h$.

Parameters: the control parameters are the ozone concentrations at t^0 .

Cost functional: Information to the assimilation process is provided only by the "observations" at the final time. To achieve a better scaling and to eliminate the positivity constraint the logarithmic form is used

$$\mathcal{J} = \frac{1}{2} \sum_{\text{gridpoints}} [(\ln O_3(t^F) - \ln O_3^{\text{obs}}(t^F)]^2 \ . \tag{21}$$

Optimization algorithm: Quasi-Newton limited memory L-BFGS [1]. The optimization proceeds until the cost functional is reduced to 0.001 of its initial value.

The results of the data assimilation are presented in Figure 1. We notice a sharp decrease in the cost function value during the first optimization iterations, followed by a slower decrease rate. The ozone concentration at gridpoint (40,30,1) is recovered within 6 accurate digits. The root mean square difference between the assimilated and the reference fields decreases to about 1e-3. Larger assimilation errors are noted near the top and bottom domain boundaries, which requires further algorithmic developments.

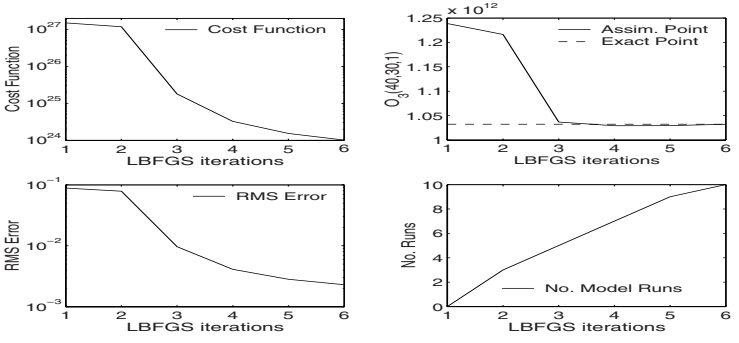


Fig. 1. Data assimilation results for ozone, for a 6h run.

6 Conclusions and Future Work

In this paper we presented some of the computational tools developed for chemical data assimilation into atmospheric transport models. The focus is on developing efficient adjoints for the stiff ordinary differential equations arising in the simulation of chemistry.

We recall the analysis of discrete Runge Kutta adjoints performed by Hager [9] and develop the discrete adjoints for Rosenbrock methods. We establish that

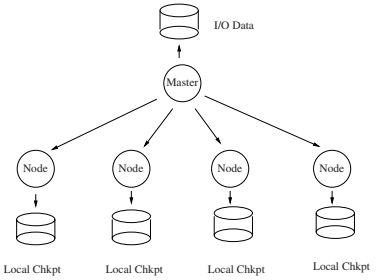


Fig. 2. The parallel adjoint STEM implements a distributed checkpointing scheme.

Runge Kutta discrete adjoints are consistent with the continuous adjoint equation with the same order as the underlying method. For control problems we propose an order analysis in the framework of partitioned Runge Kutta methods; this analysis enables an extension of the original analysis [9]. A singular perturbation analysis enables us to establish that L-stability is beneficial not only for the integration of the stiff forward model, but also for the backwards integration with both the discrete and the continuous adjoints.

The Kinetic PreProcessor KPP developed by the authors is a symbolic engine that translates a given chemical mechanism into Fortran or C kinetic simulation code. A comprehensive set of software tools for direct and adjoint sensitivity analysis were developed and implemented in the new release of the Kinetic PreProcessor (KPP-1.2). They include code generation for the Hessian and several other derivatives, as well as dedicated sparse linear algebra routines.

An adjoint of the state-of-the-science chemical transport model STEM-III was developed; it uses an efficient distributed checkpointing scheme; the chemical subsystem is generated with KPP. Results are shown for a 6 hours test case based on the TraceP scenario, where the perturbed initial concentrations of ozone are recovered from the known ozone concentrations at the end of the simulation interval.

Future work will focus on continuing the theoretical analysis of discrete adjoints for stiff solvers; on using the developed computational infrastructure to run more complex tests and to assimilate real measurements data; and on continuing to improve the computational infrastructure for data assimilation.

Acknowledgements.
The authors thank the National Science Foundation for supporting this work through the award NSF ITR AP&IM 0205198.

References

1. R. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Stat. Comput.*, 16(5):1190–1208, 1995.

2. D. G. Cacuci. Sensitivity theory for nonlinear systems. I. Nonlinear functional analysis approach. *J. Math. Phys.*, 22:2794–2802, 1981.
3. D. G. Cacuci. Sensitivity theory for nonlinear systems. II. Extensions to additional classes of responses. *J. Math. Phys.*, 22:2803–2812, 1981.
4. W.P.L. Carter. Implementation of the SAPRC-99 Chemical Mechanism into the Models-3 Framework. Technical report, Report to the United States Environmental Protection Agency, January 2000.
5. D. Daescu, A. Sandu, and G.R. Carmichael. Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems with KPP: II – Numerical Validation and Applications. *submitted to Atmospheric Environment*, 2002.
6. V. Damian, A. Sandu, M. Damian, F. Potra, and G.R. Carmichael. The kinetic preprocessor kpp – a software environment for solving chemical kinetics. *(Computers and Chemical Engineering)*, 26:1567–1579, 2002.
7. A. M. Dunker. The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. *Journal of Chemical Physics*, 81:2385, 1984.
8. H. Elbern and H. Schmidt. Ozone episode analysis by 4D-Var chemistry data assimilation. *Journal of Geophysical Research*, 106(D4):3569–3590, 2001.
9. William W. Hager. Runge-Kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282, 2000.
10. E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer-Verlag, Berlin, 1993.
11. E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer-Verlag, Berlin, 1991.
12. Menut L., Vautard R., Beekmann M., , and Honor C. Sensitivity of photochemical pollution using the adjoint of a simplified chemistry-transport model. *Journal of Geophysical Research - Atmospheres*, 105-D12(15):15,379–15,402, 2000.
13. G.I. Marchuk. *Adjoint Equations and Analysis of Complex Systems.* Kluwer Academic Publishers, 1995.
14. G.I. Marchuk, Agoshkov, and P.V. I.V., Shutyaev. *Adjoint Equations and Perturbation Algorithms in Nonlinear Problems.* CRC Press, 1996.
15. P. Miehe, A. Sandu, G.R. Carmichael, Y. Tang, and D. Daescu. A communication library for the parallelization of air quality models on structured grids. *(Atmospheric Environment)*, 36:3917–3930, 2002.
16. Vautard R., M.Beekmann, and L. Menut. Applications of adjoint modelling in atmospheric chemistry: sensitivity and inverse modelling. *Environmental Modeling and Sofware*, 15:703–709, 2000.
17. A. Sandu, D. Daescu, and G.R. Carmichael. Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems with KPP: I – Theory and Software Tools. *submitted to Atmospheric Environment*, 2002.
18. A. Sandu, D. daescu, and G.R. Carmichael. Discrete Adjoint for Stiff ODE Solvers. *In Preparation*, 2003.
19. Z. Sirkes and E. Tziperman. Finite difference of adjoint or adjoint of finite difference? *Mon. Weather Rev.*, 49:5–40, 1997.
20. K.Y. Wang, D.J. Lary, Shallcross, D.E., Hall aS.M., and Pyle J.A. A review on the use of the adjoint method in four-dimensional atmospheric-chemistry data assimilation. *Q.J.R. Meteorol. Soc.*, 127(576 (Part B)):2181–2204, 2001.

Virtual Telemetry for Dynamic Data-Driven Application Simulations

Craig C. Douglas[1,2], Yalchin Efendiev[3], Richard Ewing[3], Raytcho Lazarov[3], Martin J. Cole[4], Greg Jones[4], and Chris R. Johnson[4]

[1] University of Kentucky, Department of Computer Science, 325 McVey Hall, Lexington, KY 40506-0045, USA
{douglas}@ccs.uky.edu
[2] Yale University, Department of Computer Science, P.O. Box 208285 New Haven, CT 06520-8285, USA
douglas-craig@cs.yale.edu
[3] Texas A&M University, College Station, TX, USA
{efendiev,lazarov}@math.tamu.edu and richard-ewing@tamu.edu
[4] Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA
mjc@sci.utah.edu and crj@cs.utah.edu

Abstract. We describe a virtual telemetry system that allows us to devise and augment dynamic data-driven application simulations (DDDAS). Virtual telemetry has the advantage that it is inexpensive to produce from real time simulations and readily transmittable using open source streaming software. Real telemetry is usually expensive to receive (if it is even available long term), tends to be messy, comes in no particular order, and can be incomplete or erroneous due to transmission problems or sensor malfunction. We will generate multiple streams continuously for extended periods (e.g., months or years): clean data, somewhat error prone data, and quite lossy or inaccurate data. By studying all of the streams at once we will be able to devise DDDAS components useful in predictive contaminant modeling.

1 Introduction

Consider an extreme example of a disaster scenario in which a major waste spill occurs in a river flowing through the center of a major city. In short time, the waste will be on kilometers of the city's shoreline.

Sensors can now be dropped into an open water body to measure where the contamination is, where the contaminant is going to go, and to monitor the environmental impact of the spill. Whether or not the procedure to drop the sensors exists today is not relevant for the moment; only that it could be done. Scrambling aircraft to drop sensors in a river is no longer such a far fetched scenario.

A well designed DDDAS predictive contaminant tracking program will be able to determine where the flow will go, the neighborhoods that have to be

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 279–288, 2003.
© Springer-Verlag Berlin Heidelberg 2003

evacuated, and optimize a clean-up plan. A backward in time DDDAS program will offer help in determining where and when a contaminant entered the environment. For this to become reality data streaming tolerant algorithms with sensitivity analysis incorporated for data injection, feature extraction, and multiple scaling must be developed for this to scenario to become a reality.

There are several approaches to designing DDDAS systems for disaster management. One is to collect real data and replay it while developing new data dynamic algorithms. Another is to generate fictional data continuously over a several year period with disasters initiated at random times or by human intervention (possibly without informing the other researchers in advance).

In either case, data streaming is a useful delivery system. Like realistic telemetry, data arrives that is usually incomplete, not in any order, and occasionally wrong. In short, it is an appalling mess.

In Sect. 2, we define an application that is our first test case. In Sect. 3, we define the DDDAS components that are of interest in this paper. In Sect. 4, we define model reduction for our application. In Sect. 5, we describe the telemetry middleware that we are in the process of developing and what our choices are. In Sect. 6, we discuss the programming environment that we are using to generate useful prototypes quickly so that a final package, useful to others as well, can be built. In Sect. 8, we draw some conclusions.

2 An Example Application

As an example application we consider a single component contaminant transport in heterogeneous porous media taking into account convection and diffusion effects. This simple model will be further extended by incorporating additional physical effects as well as uncertainties. The mathematical formulation of the problem is given by coupled equations that describe pressure distribution and the transport equations, $\nabla \cdot \mathbf{k} \nabla p = f$, $S_t + \mathbf{v} \cdot \nabla S = \nabla \cdot D \nabla S$, $\mathbf{v} = -\mathbf{k} \nabla p$. We consider two different permeability field scenarios. For the first case we assume that a large horizontal permeability streak is located in the middle of the domain and there are two vertical low permeability zones. The background permeability is taken to be 1, the permeability of high streak region is taken to be 2, and vertical low permeability regions have permeability 0.01. The second permeability field is chosen as an unconditional realization of a fractal field whose semivariogram is given by $\gamma(h) = Ch^{0.5}$, where the semivariogram is defined as

$$\gamma(\mathbf{h}) = \frac{1}{2} E[(\xi(\mathbf{x} + \mathbf{h}) - \xi(\mathbf{x}))^2].$$

A horizontal high permeability streak is added at the center of the domain. To generate a realization of this field we use fractional Brownian motion theory developed in [1, 7, 15]. The realization of the field is generated from the generalized Weierstrass-Mandelbrot function with fractal co-dimension $\beta/2$ ($\beta = 0.5$)

$$\xi(\mathbf{x}) = \sum_{i=1}^{\infty} A_i \lambda^{-i\frac{\beta}{2}} \sin(\lambda^i \mathbf{x} \cdot \mathbf{e}_i + \phi_i),$$

where A_i are normally distributed random numbers with mean zero and variance 1, $\mathbf{e}_i$ are uniformly distributed on the unit sphere, ϕ_i are uniformly distributed over $[0, 2\pi]$, and $\lambda > 1$. This isotropic random field has an approximately power semivariogram, i.e., $0 < ch^\beta < \gamma(h) < Ch^\beta$. In all the examples the rectangular domain $[0, 2] \times [0, 1] \times [0, 1]$ is consider and the following initial and boundary conditions are imposed. We assume the pressure at the inlet $x = 0$ to be $p = 1$ and $p = 0$ at the outlet $x = 2$. For the other faces we assume no flow boundary conditions for the pressure equation. For the concentration field we assume that the concentration is $S = 1$ at the inlet $x = 0$ and $\partial S/\partial n = 0$ is imposed on the other faces. Furthermore, $D = 0.02$ is taken to be constant. The computations are implemented using 3-D finite element simulator developed at Texas A&M employing the mesh generator NETGEN [16].

The concentration fields cannot easily be rendered in a grayscale. We have decided to post the concentrations on the web at URL

$$http://www.dddas.org/itr-0219627/papers.html\#iccs2003.$$

3 DDDAS Components

In contaminant movement predictions, it is common to run simulations for a few hours or days as a batch process. Although the individual application simulation periods are a few wall clock hours, they are built to update the early time steps with real data as it becomes available.

Converting software from a data set oriented batch code to a data stream continuously running code requires significant, nontrivial changes. We are dissecting contaminant transport models and associated codes in order to understand how to convert these models into the DDDAS model. Underground situations (e.g., nuclear waste contamination from a leaky containment vessel) are different from an above ground situation (contaminants in a lake and/or river). We are investigating both of these situations as well as the combination of the two.

The addition of DDDAS features to a system requires that aspects related to the initial data must be identified. Additionally, the time-related updating procedures must investigated. As we approach these requirements we will focus on maintaining a solution that is generally useful to other application fields that are initial boundary value problem (IBVP) based. Almost all IBVP formulations preclude dynamic data or backward in time error checking, sometimes in quite subtle ways.

The use of continuous data streams instead of initial guess only data sets presents an additional challenge for data driven simulations since the results vary based on the sampling rate and the discretization scheme used. Dynamic data assimilation or interpolation might be necessary to provide a feedback to experimental design/control. DDDAS algorithms also need to dynamically assimilate new data at mid-simulation as the data arrives, necessitating "warm restart" capabilities.

Modifying application programs to incorporate new dynamically injected data is not a small change in the application program, particularly since the

incoming data tends to be quite messy. It requires a change in the application design, the underlying solution algorithms, and the way people think about the accuracy of the predictions.

Uncertainties in DDDAS applications emanate from several sources, namely uncertainty associated with the model, uncertainties in the input data (streamed), and the environment variables. Identifying the factors that have the greatest impact on the uncertainty output of the calculations is essential in order to control the overall processes within specific limits. Handling all output distributions to provide error bounds is, for most realistic problems, a computationally prohibitive task. Hence, using prior observations to guide the output distribution estimations presents a possible approach to incorporating uncertainty in control decisions.

Incorporating these statistical errors (estimations or experimental data uncertainties) into computations, particularly for coupled nonlinear systems, is difficult. This is compounded by the fact that tolerances may also change adaptively during a simulation. Error ranges for uncertainty in the data must be created and analyzed during simulations. Sensitivity analysis must be performed continuously during simulations with options in case of a statistical anomaly.

The common mathematical model in many DDDAS applications may be formulated as solving a time dependent, nonlinear problem of the form $F(x+Dx(t))=0$, by iteratively choosing a new approximate solution x based on the time dependent perturbation $Dx(t)$.

At each iterative step, the following three issues may need to be addressed. Incomplete solves of a sequence of related models must be understood. In addition the effects of perturbations, either in the data and/or the model, need to be resolved and kept within acceptable limits. Finally, nontraditional convergence issues have to be understood and resolved. Consequently, there will be a high premium on developing quick approximate direction choices, such as, lower rank updates and continuation methods. It will also be critical to understand the behavior of these chosen methods.

By generating telemetry in real time, we allow for the new, DDDAS code to determine how well it has predicted the past, assuming the DDDAS code runs much faster than real time. We run a simulation backwards in time to a point where we can verify that we have not introduced too great an error into a simulation at a future time. This is particularly important when deciding whether or not to introduce all or just part of a data stream. For example, if a data stream update causes a loss or addition of mass when it is conserved, the data stream update will lead to an abrupt loss of usefulness unless a filtering process is developed to maintain the conservation of mass. We are developing new filters for our applications that resolve the errors introduced by converting the applications to data streams.

4 Model Reduction and Multiscale Computations

Model reduction will largely be accomplished with the use of upscaling techniques. Due to complicated interactions and many scales, as well as uncertainties, upscaling is desirable. The upscaling is in general nontrivial because heterogeneities at all scales have a significant effect, and these effects must be captured in the coarsened subsurface description. Most approaches for upscaling are designed to generate a coarse grid description of the process which is nearly equivalent (for purposes of flow simulation) to an underlying fine grid description. We will employ both static and dynamic upscaling techniques. Static upscaling techniques will generally be used in coarsening the media properties, while the dynamic upscaling will be employed to understand the effect of the small scale dynamics on the larger scales. One of the important aspects of our upscaling approach is the use of carefully selected dynamic coarse scale variables in a multiscale framework.

To demonstrate the main idea of our static upscaling procedures we consider our application model, the pressure equation $\nabla \cdot \mathbf{k} \nabla p = f$. Within this approach the heterogeneities of the media are incorporated into the finite element (or finite volume element) base functions that are further coupled through the global formulation of the problem We seek the solution of this equation in a coarse space whose base elements $\phi_i(\mathbf{x})$ contain the local heterogeneity information, $V_h = span(\phi_i)$. The numerical solution u_h is found from

$$\int_D \mathbf{k} \nabla u_h \nabla v_h d\mathbf{x} = \int_D f v_h d\mathbf{x}, \ \forall v_h \in V_h.$$

This approach allows us to solve the saturation equation on the on the fine scale grid as well as on the coarse scale grid since the fine scale features of the velocity field can be recovered from the base functions. Consequently, we can adjust the permeability at the fine scale directly. One of the advantages of these approaches is that the local changes of the permeability field will only affect few base functions. Consequently, we will only need to re-compute few base functions and solve the coarse problem. This methodology will be useful in backward integration for finding sources of inconsistencies within our DDDAS application. We will also employ traditional approaches based on upscaling of permeability field [2] and use mainly the upscaling of permeability field with oversampling techniques [19]. The main idea of this approach is to use larger domain (larger than the coarse block) in order to reduce the boundary effects. To reduce the grid effects grid based upscaling techniques will be employed.

For the time dependent transport problems we will employ coarsening techniques that are based on dynamic upscaling. For these approaches the dynamic quantities (e.g., quantities that depend on concentration) are coarsened and their functionality is determined through analytical considerations. Determining the form of coarse scale equations is important for multiscale modeling. Our previous approaches on this issue were mainly in two directions. First approach that is based on perturbation techniques models subgrid effects as a nonlocal macrodispersion term [5]. This approach takes into account the long range interaction in

the form of diffusion term that grows in time. Our second approach based on homogenization techniques [6] computes the dynamic upscaled quantities using local problems [4, 3].

5 Telemetry Middleware

Real telemetry is usually expensive to receive (if it is even available on a long term basis), tends to be messy, comes in no particular order, and can be incomplete or erroneous due to transmission problems or sensor malfunction. For predictive contaminant telemetry, there are added problems that due to pesky legal reasons (corporation X does not want it known that it is poisoning the well water of a town), the actual data streams are not available to researchers, even ones providing the simulation software that will do the tracking.

Virtual telemetry has the advantage that it is inexpensive to produce from real time simulations. The fake telemetry can easily be transmitted using open source streaming software.

We will generate multiple streams continuously for extended periods (e.g., months or years): clean data, somewhat error prone data, and quite lossy or inaccurate data. By studying all of the streams at once we will be able to devise DDDAS components useful in predictive contaminant modeling.

Real telemetry used in predictive contaminant monitoring comes in small packets from sensors in wells or placed in an open body of water. There may be a few sensors or many. With virtual telemetry, we can vary the amount of telemetry that we broadcast and its frequency.

There are a number of issues that are being resolved in the course of our project.

1. Should the telemetry data be broadcast as an audio stream?
2. Should the telemetry data be broadcast as a movie stream?
3. Should a complete 3D visualization be broadcast (and in what form)?
4. Should only sparse data from discrete points in a domain be broadcast?
5. At what rate can the virtual telemetry be broadcast so that network administrators do not cut off the data stream?

We are building middleware to answer all of the questions above.

There are a number of open source projects for doing reliable data streaming. Palantir [8] is a former commercial product that has been re-released as open source. It supports audio and video streaming as well as general data streaming. Gini [9] is an audio and video streamer that is still in beta, but is fairly active in development. GStreamer [10] is more of a streaming framework developed by graduate students in Washington. It supports audio, video, and general data streaming. VideoLAN [18] is a video streamer developed by students at the École Centrale Paris. QuickTime Streaming Server is an audio and video streaming server that Apple offers. Of the five, GStreamer is the clear first choice to concentrate on.

Broadcasting the telemetry as audio has the advantage that there are many programs to choose from to generate the data streams and to "listen" to them on the receiving end.

Broadcasting the telemetry as a movie stream or a full 3D visualization has the advantage that it can be trivially visualized on the receiving end. This is particularly attractive when studying incomplete and/or erroneous data streams. However, there is a potential of transmitting too much data and overwhelming the network.

Broadcasting only sparse data from discrete points in any form has pluses and minuses. Almost any Internet protocol can be used. However, do we really want to tie ourselves to one Internet protocol?

Avoiding the attention of network administrators is a serious concern. We must balance adequate data streams with not having any serious impact on a network. This is highly dependent on where we are running the virtual telemetry from and to and cannot easily be determined *a priori*. However, it is easily determined *a posteriori*, which will be part of a well behaved, adaptive broadcast system.

6 Program Construction

Current interactive scientific visualization and computational steering implementations require low latency and high bandwidth computation in the form of model generation, solvers, and visualization. Latency is particularly a problem when analyzing large data sets, constructing and rendering three-dimensional models and meshes, and allowing a scientist to alter the parameters of the computation interactively. However, large-scale computational models often exceed the system resources of a single machine, motivating closer investigation of meeting these same needs with a distributed computational environment.

To achieve execution speeds needed for interactive three-dimensional problem solving and visualization, we have developed the SCIRun problem solving environment and computational steering system [12, 11]. SCIRun allows the interactive construction, debugging, and steering of large scale scientific computations. SCIRun can be conceptualized as a computational workbench, in which a scientist can design via a dataflow programming model and modify simulations interactively. SCIRun enables scientists to interactively modify geometric models, change numerical parameters and boundary conditions, and adaptively modify the meshes, while simultaneously visualizing intermediate and final simulation results.

When the user changes a parameter in any of the module user interfaces, the module is re-executed, and all changes are automatically propagated to all downstream modules. The user is freed from worrying about details of data dependencies and data file formats. The user can make changes without stopping the computation, thus steering the computational process. In a typical batch simulation mode, the scientist manually sets input parameters, computes results, assesses the results with a combination of separate analytical and visualization

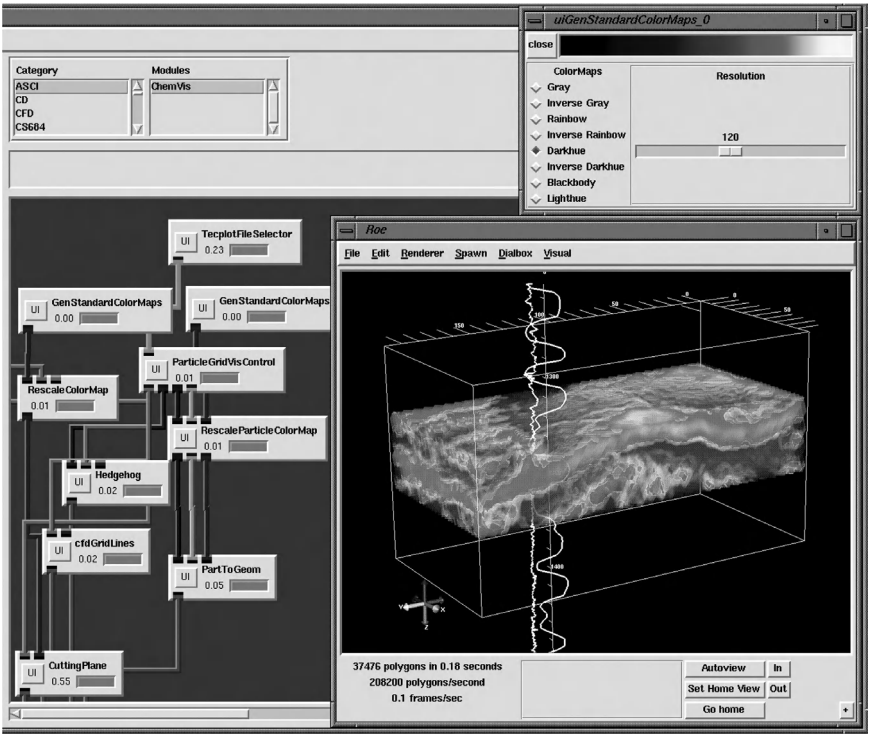


Fig. 1. SCIRun is open source software, and is freely downloadable.

packages, then iterates this procedure. SCIRun closes the loop and allows interactive steering of the design, computation, visualization, and analysis phases of a simulation[17].

It is important to have the ability to develop and manipulate the telemetry simulation interactively. An example of this need is with sensors. It can be imagined that the typical telemetry collection will combine many types of sensors reporting different data types and at different rates. The ability to quickly and seamlessly swap sensor types and dataflow types is critical to a DDDAS application. SCIRun provides the ability to develop rapid prototypes. The modular design of SCIRun allows the simulation to be built from a set of components, in such a system components or modules may be assigned to each sensor type making it easy to combine or change the sets of sensors used in a simulation. This modularity also enables a change in the scale or type of a problem being simulated to be done very quickly and directly.

Testing of the system and its robustness with simulated data will require the ability to manipulate the data stream either by adding data losses or abnormalities. Using SCIRun, the running telemetry stream can be manipulated by hand essentially as a data steering problem.

As the testbed is expanded it will almost certainly be a requirement that the application perform in a distributed environment. It will also be likely that the application be required to work, seamlessly, with other software packages or libraries, not directly incorporated in the application. There are numerous examples of the ease of bridging third party software with SCIRun, and of SCIRun operating in a distributed environment [14, 13] . This will enable the utilization of existing free codes for streaming the dynamic data needed to drive this simulation.

7 Conclusions

We have described issues in constructing a virtual telemetry system. Our target is enabling DDDAS components in a predictive contaminant model. We expect to have useful open source middleware readily available on the Internet soon. Once the middleware is completed, we can explore many interesting features of DDDAS.

Acknowledgements.
This work was supported in part by a National Science Foundation collaborative research grant (EIA-0219627, EIA-0218721, and EIA-0218229).

References

1. Chu, J., and Journel, A. In *Geostatistics for the next Century*, R. Dimitrakopoulos, Ed. Kluwer Academic Publishers, 1994, pp. 407–412.
2. Durlofsky, L. J. Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media. *Water Resour. Res. 27* (1991), 699–708.
3. Efendiev, Y., and Durlofsky, L. Accurate subgrid models for two-phase flow in heterogeneous reservoirs. paper SPE 79680 presented at the 2003 SPE Symposium on Reservoir Simulation, Houston, TX.
4. Efendiev, Y., and Durlofsky, L. A generalized convection diffusion model for subgrid transport in porous media. submitted to SIAM on Multiscale Modeling and Simulation.
5. Efendiev, Y. R., Durlofsky, L. J., and Lee, S. H. Modeling of subgrid effects in coarse scale simulations of transport in heterogeneous porous media. *Water Resour. Res. 36* (2000), 2031–2041.
6. Efendiev, Y. R., and Popov, B. On homogenization of nonlinear hyperbolic equations. submitted to SIAM J. Appl. Math. (available at http://www.math.tamu.edu/∼yalchin.efendiev/submit.html).
7. Falconer, K. *Fractal Geometry: Mathematical Foundations and Applications.* John Wiley & Sons, 1990.
8. FastPath Research. Palantir. http://www.fastpath.it/products/palantir.
9. Gini. Gini. http://gini.sourceforge.net.
10. GStreamer. Gstreamer. http://www.gstreamer.net.

11. Johnson, C., and Parker, S. The scirun parallel scientific computing problem solving environment. In *Ninth SIAM Conference on Parallel Processing forScientific Computing* (1999).
12. Johnson, C., Parker, S., and Weinstein, D. Large-scale computational science applications using the scirun problem solving environment. In *Supercomputer 2000* (2000).
13. Miller, M., Hansen, C., and Johnson, C. Simulation steering with scirun in a distributed environment. In *Applied Parallel Computing, 4th InternationalWorkshop, PARA'98*, E. E. B. Kagstrom, J. Dongarra and J. Wasniewski, Eds., vol. 1541 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998, pp. 366–376.
14. Miller, M., Hansen, C., Parker, S., and Johnson, C. Simulation steering with scirun in a distributed memory environment. In *Seventh IEEE International Symposium on HighPerformance Distributed Computing (HPDC-7)* (jul 1998).
15. Oh, W. *Random field simulation and an application of kriging to image thresholding*. PhD thesis, State Univeristy of New York at Stony Brook, 1998.
16. Schoeberl, J. NETGEN-An advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science 1* (1997), 41–52. http://www.sfb013.uni-linz.ac.at/ joachim/netgen/.
17. SCIRun: A Scientific Computing Problem Solving Environment. Scientific Computing and Imaging Institute (SCI), http://software.sci.utah.edu/scirun.html, 2002.
18. VideoLAN. Videolan. http://www.videolan.org.
19. Wu, X. H., Efendiev, Y. R., and Hou, T. Y. Analysis of upscaling absolute permeability. *Discrete and Continuous Dynamical Systems, Series B 2* (2002), 185–204.

Tornado Detection with Support Vector Machines

Theodore B. Trafalis[1], Huseyin Ince[1], and Michael B. Richman[2]

[1] School of Industrial Engineering, University of Oklahoma,
Norman OK 73019, USA,
ttrafalis@ou.edu,
http://www.lois.ou.edu
[2] School of Meteorology, University of Oklahoma,
Norman OK 73019, USA,
mrichman@ou.edu

Abstract. The National Weather Service (NWS) Mesocyclone Detection Algorithms (MDA) use empirical rules to process velocity data from the Weather Surveillance Radar 1988 Doppler (WSR-88D). In this study Support Vector Machines (SVM) are applied to mesocyclone detection. Comparison with other classification methods like neural networks and radial basis function networks show that SVM are more effective in mesocyclone/tornado detection.

1 Introduction

The National Weather Service (NWS)uses several severe weather detection algorithms. One of them is the Mesocyclone Detection Algorithm(MDA). It is based on empirical rule based algorithms and works on the WSR-88D. The skill of the MDA algorithm is rather low. For example, the percentage of observed mesocyclones that were correctly forecast is below fifty percent. Additionally, there are a large number of mesocyclones forecast that do not occur. Moreover, modeling of a complex dynamical system with a closed mathematical expression is not an easy task. Owing to these two factors, there is a need to develop or use new techniques to address the problem.

One of the techniques that does not rely on assumptions about the underlying probability distribution governing the input data is Support Vector Machines(SVM). This is in contrast to other types of models that assume the data follow the normal distribution, like those based on the linear discriminant analysis (DA) method. Furthermore, the SVM classification method is more robust than other techniques such as neural networks (NN) and radial basis networks [14].

The MDA algorithm and SVM methods will be briefly explained in the next two sections. Then a comparison with other techniques is discussed in section 4. Section 5 concludes the paper.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 289–298, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 The Mesocyclone Detection Algorithm (MDA)

In this section we present one of the detection algorithms developed by the National Severe Storms Laboratory (NSSL), the MDA (Mesocyclone Detection Algorithm). The data used in this study are intermediate data computed by this algorithm. The NSSL has developed the NSSL MDA for the WSR-88D system to automatically detect and diagnose the Doppler radar radial velocity patterns associated with all storm-scale (1-10km diameter) vortices in thunderstorms, rather than defining the strength thresholds at the very first analysis step used by 88D B9MDA. The first step of the algorithm is to preprocess the Doppler velocity data. Noisy data, such as velocities whose reflectivity values are below a preset threshold (typically 0 - 20 dBZ), are deleted. Next, the NSSL MDA's automated vortex detection techniques set the initial strength thresholds to be much lower, and classifications and diagnosis are performed on the properties of the four-dimensional detections. The algorithm first processes data at the one-dimensional (1D) level; shear segments of cyclonic azimuthal shear are detected. Next, the shear segments are horizontally associated to form two-dimensional (2D) features. The NSSL MDA then uses vertical association to create three-dimensional (3D) detections at the end of each volume scan. Finally, time association and tracking are employed to complete the process. More information about how MDA works can be found in [5,13].

3 Support Vector Machines: A Brief Review

SVM is a learning machine developed by Vapnik [3,14] based on statistical learning theory [15]. In the case of classification [1,3,6,7], we try to find an optimal hyperplane that separates two classes of data points D_ℓ as shown in Figure 1. The objective is to establish an equation of the hyperplane that divides D_ℓ into two sets S_1 and S_2, leaving all the points of the same class on the same side while maximizing the minimum distance between either of the two classes and the resulting hyperplane [3,1]. In order to find the hyperplane that has the maximum margin between the two classes while at the same time minimizes the misclassification error on D_ℓ, one has to solve the following Quadratic Programming (QP) optimization problem:

$$\begin{aligned} \min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{Subject to} \\ y_i \cdot (w \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, 2, ..., l \\ \xi_i \geq 0, \end{aligned} \tag{1}$$

where $w \in \Re^d$ is the vector normal to the separating hyperplane, $b \in \Re$ is the offset with respect to the origin, ξ_i are the slack variables that measure the empirical misclassification error, $y = \pm 1$ and C is the regularization parameter [4,

16,11,15] defining the trade off between margin and empirical error. By assigning a Lagrangian multiplier α_j to each constraint and by introducing the variables $\alpha = (\alpha_1, \alpha_2, ..., \alpha_l)$, the matrix $\mathbf{K}_{ij} = (y_i y_j \mathbf{x}_i \mathbf{x}_j)$, and vectors $e = (1, 1, ..., 1)$, and $y = (y_1, y_2, ..., y_l)$, the dual problem can be formulated in closed form as:

$$\begin{aligned} \max \ & -\alpha^T e + \frac{1}{2}\alpha^T \cdot \mathbf{K} \cdot \alpha \\ & \text{Subject to} \\ & \alpha^T \cdot y = 0 \\ 0 \leq \alpha_j \leq C \quad & \forall j = 1, ..., l. \end{aligned} \tag{2}$$

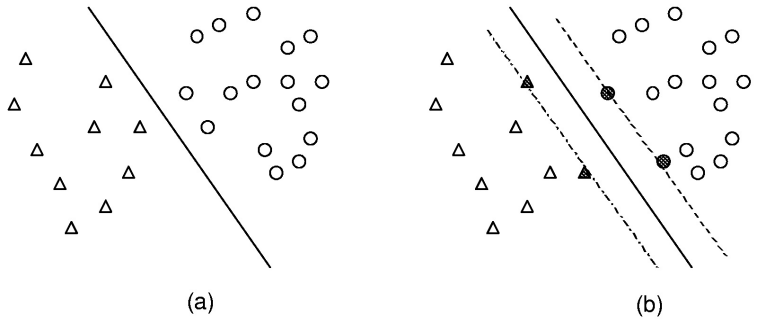


Fig. 1. Separating hyperplane and optimal separating hyperplane. Both solid lines in (a) and (b) separate the two identical sets described by circles and triangles. But the solid line in (b) leaves the closest points (filled circles and triangles) at the maximum distance. The distance between dashed lines in (b) gives the maximum margin.

The above approach can be generalized in the case of nonlinear separating surfaces mapping the input data $\{x_i\}_{i=1}^{l}$ into a higher dimensional feature space through the use of a feature map $\phi : \Re^d \to F$. Then a separating hyperplane (if it exists) can be found in that space.

Our objective is to determine a discriminant function

$$f(x) = sign(w \cdot \phi(x) + b) = \left\{ \begin{array}{l} +1 \text{ if } x \in S_1 \\ -1 \text{ if } x \in S_2 \end{array} \right\}, \tag{3}$$

where $\phi : \Re^d \to F$ is a map of $\Re^d$ into the feature space F.

Specifically, the corresponding SVM optimization problem is as follows:

$$\begin{aligned} \min \ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} \xi_i \\ & \text{Subject to} \\ y_i \cdot (w \cdot \phi(x_i) + b) \geq 1 - \xi_i, & i = 1, 2, ..., l \\ \xi_i \geq 0. & \end{aligned} \tag{4}$$

The solution of the above problem can be expressed as a linear combination of the ϕ-images of the data points [9], i.e.

$$w = \sum_{i=1}^{l} \alpha_i \cdot y_i \cdot \phi(x_i). \tag{5}$$

Therefore, if we define a dot product in the feature space [9] as $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, then problem 4 can be expressed as in 2 but with $K_{ij} = y_i \cdot y_j \cdot k(x_i, x_j)$.

Combining equation (3) and (5), $f(x)$ can be expressed as follows

$$f(x) = sign\left(\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot \phi(x_i) \cdot \phi(x) + b\right) \tag{6}$$

or equivalently, using the kernel function as

$$f(x) = sign\left(\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot k(x_i, x) + b\right) \tag{7}$$

The important data points are the ones for which $\overline{\alpha_i} > 0$ where $\overline{\alpha}$ denotes the optimal solution of the above problem 1. Those points are called support vectors and provide a sparse representation of the discriminant function. Usually, the number of support vectors is smaller than the number of data points. There are several kernel function to choose, e.g. radial basis function, polynomial function, etc. [16].

To solve this quadratic programming problem, several decomposition algorithm have been proposed [12,2,8]. In this paper, an SVM learning decomposition algorithm, SVMTorch [2] is applied to mesocyclone detection since the number of data is large (3768 data points).

4 Experiments

The circulation database used for SVM method was 'truthed' to determine which circulations were associated with reports of tornado events. This was the first step to obtain the actual target value for each observation. Then, the missing observations were removed from the dataset. 23 variables were chosen as inputs. These variables are intermediate outputs of WSR-88D and NSSL mesocyclone algorithms. They are used by the National Weather Service to diagnose mesocyclones during the severe storm warning operations.

In a recent paper [13],the reduction of input variables is suggested by using statistical methods or principal component analysis. On the other hand, some authors suggest that these techniques do not improve the prediction of the tornados [5]. Because of this, all variables will be used in this paper. The definition of the variables and more information can be found in [13,5].

Table 1. Confusion Matrix

		Observed		
		Yes	No	Total
Forecast	Yes	Hits (a)	False Alarm (b)	Forecast Yes
	No	Misses (b)	Correct Negatives (d)	Forecast No
Total		Observed Yes	Observed No	Total

The performance of SVM methods is evaluated by using a suite of forecast evaluation indices based on a contingency table (otherwise also known as a "confusion matrix"). More information about some of these measures can be found in [13,5,10,17]. The confusion matrix is shown in Table 1.

In this definition of the confusion matrix, the Probability of Detection, POD, can be defined as

$$POD = \frac{a}{(a+c)}. \tag{8}$$

POD measures the fraction of observed events that were correctly forecast.Its range is 0 to 1 and a perfect score is 1 (or 100%). Note that POD is sensitive to hits, good for rare events. POD ignores false alarms and can be improved artificially by issuing more "yes" forecasts to increase the number of hits.

False Alarm Rate, FAR, can be defined as

$$FAR = \frac{b}{(a+b)}. \tag{9}$$

FAR measures the fraction of "yes" forecasts in which the event did not occur. Its range is 0 to 1 and 0 is a perfect rate. FAR is sensitive to false alarms and it ignores misses. It can be improved artificially by issuing more "no" forecasts to reduce the number of false alarms.

The Critical Success Index, CSI, is defined as

$$CSI = \frac{a}{(a+c+b)}. \tag{10}$$

CSI measures the fraction of observed and/or forecast events that were correctly forecast. Its range is 0 to 1 with a perfect score being 1. CSI is sensitive to hits, penalizes both misses and false alarms. It does not distinguish the source of forecast error and it depends on the climatological frequency of events (worse scores for rarer events) since some hits can occur purely due to random chance.

Accuracy is defined as

$$Accuracy = \frac{(a+d)}{total}. \tag{11}$$

Accuracy measures the fraction of all forecasts that were correct, which makes it a seemingly intuitive measure. The range is 0 to 1 with 1 being best. However, it can be misleading since it is heavily influenced by the most common category, usually "no event" in the case of severe weather.

Bias is defined as

$$Bias = \frac{(a+b)}{(a+c)}. \tag{12}$$

Bias measures the ratio of the frequency of forecast events to the frequency of observed events. The range is from 0 to infinity. A perfect score is 1. Bias indicates whether the forecast system has a tendency to underforecast (bias $<$ 1) or overforecast (bias $>$1) events. It does not measure how well the forecast corresponds to the observations; it measures only relative frequencies.

Probability of False Detection, POFD, is defined as

$$POFD = \frac{b}{(b+d)}. \tag{13}$$

POFD measures the ratio of false alarms to the total number of no observations. The probability of false detection is a measure of inaccuracy with respect to the observations and provides a measure of the extent to which the forecasts provide a false warning for the occurrence of an event. POFD varies from 0 to 1. A perfect score is zero.

Hanssen and Kuipers discriminant (true skill statistic), H-K Skill,is a measure of the improvement of a forecast over some reference forecast (e.g., random forecast).

$$H - KSkill = \frac{a}{(a+c)} - \frac{b}{(b+d)}. \tag{14}$$

H-K Skill measures the ability of the forecast to separate the "yes" cases from the "no" cases. It can also be interpreted as accuracy(events) + accuracy(non-events) - 1. H-K Skill ranges from -1 to 1 with 0 indicating no skill. A perfect statistic value is 1. The advantage of H-K Skill is that it uses all elements in contingency table. It does not depend on climatological event frequency. For rare events, H-K Skill is weighted heavily toward the first term (same as POD).

Odds Ratio, OR, is a newer statistic for forecast evaluation.

$$OR = \frac{(a*d)}{(c*b)}. \tag{15}$$

OR measures the ratio of the probability of making a hit to the probability of making a miss or false alarm. For the OR, the range is 0 to infinity, 1 indicates no skill. A perfect score is infinity. OR gives better scores for rarer events.

Comparison of SVM, NN, DA and the rule-based MDA based on these indices are given in Tables 3 and 4. The trade-off value(C), kernel functions and kernel related parameter have to be determined. The trade-off value is set to 1000. The radial basis kernel function is used that is shown in equation (16).

$$k(x, y) = exp(-\frac{1}{2\sigma^2}\|x - y\|^2) \tag{16}$$

We have used different σ's to find the best fit. Table 2 shows the results for different parameters. As it can be seen from the Table 2, the tornado dataset verification indices are very sensitive to this parameter. The results show that accuracy increases up to a σ of 7.25 and then decreases slightly. A similar behavior can be seen for CSI. The POD is relatively insensitive to σ over the range tested, though the highest values occur between σ values of 2.5 and 7.25. The Bias indicates that all of the values lead to underforecasting, though it is minimized at the two lowest values of σ. The odds ratio shows a clear advantage at the two largest σ. Both the FAR and POFD should be lowest for the best results. These measures are lowest for the two largest values of σ. The skill score is high over the σ range 2.5 to 15, with a maximum of 57.13 percent at a σ of 7.5.

Table 2. The Accuracy, CSI, POD, Bias, ODDS, FAR, POFD and H-K Skill for different kernel parameters

σ	ACCURACY (%)	CSI (%)	POD (%)	Bias (%)	ODDS (Ratio)	FAR (%)	POFD (%)	H-K Skill (%)
1.25	83.60	44.65	60.13	94.76	14.08	36.43	9.77	50.36
2.5	85.54	48.70	62.38	90.48	19.54	30.99	7.93	54.45
7.25	88.27	53.46	61.24	75.80	37.44	19.19	4.11	57.13
15	87.98	51.75	58.71	72.15	36.53	18.61	3.79	54.92

Tables 3 and 4 show that the SVM method outperforms the NN, DA and MDA rule based algorithms. For comparison, 10 different training and validation sets are used. The average CSI, POD and FAR are provided.

Since the MDA employs a rule based algorithm, it does not explain the relation between the tornado and the input variables. DA is a classification method with several assumptions such as normality and the homoscedasticity of the distribution. Most of the time those assumptions are violated. The theory behind the NN does not make any assumption about the distribution of the empirical data. Because of this, it outperforms the MDA rule based algorithm and DA. Despite this, the error function for NN is not convex and most of the time, the solution is a local optimum. In contrast, the SVM approach provides the most optimal solution of the methods tested because the SVM training

Table 3. The validation CSI for SVM, NN, MDA and DA for ten different training and validation sets. The values for MDA, DA and NN are taken from [5].

Seed	CSI_{MDA}	CSI_{DA}	CSI_{NN}	CSI_{SVM}
1	26.9	31	36.9	47.78
2	24	29.2	35.7	47.49
3	24.7	28.1	38.3	50.2
4	28.7	27.7	33.6	49.38
5	27.4	30	34.2	47.5
6	28	28.8	32.5	44.93
7	29.9	26.1	33.1	51.62
8	21.3	28.7	29.1	48.19
9	27.7	30.6	37.8	50.5
10	21.5	26.5	31.7	49.41
Average	26.01	28.67	34.29	48.7

Table 4. The POD and the FAR are also shown for SVM and NN. The values for MDA, DA and NN are taken from [5].

Seed	POD_{NN}	FAR_{NN}	POD_{SVM}	FAR_{SVM}
1	51.2	43.10	62.53	33.06
2	50	44.4	60	30.5
3	55	44.3	64.05	30.11
4	58.8	56.1	60.51	27.13
5	50	48.1	60.25	30.81
6	47.5	49.3	59.49	35.26
7	52.5	52.8	68.61	32.42
8	46.2	56	60.51	29.71
9	60	49.5	64.56	30.14
10	47.5	51.3	63.29	30.75
Average	51.87	49.49	62.38	30.99

algorithm is convex. Hence, it outperforms the other three methods. The details of the comparison between these techniques are based on the forecast verification statistics. The results indicate that both the CSI and POD are considerably larger for SVM, whereas the FAR is reduced dramatically to less than half of value found for the currently deployed NN algorithm (Table 2, σ =7.25).

5 Conclusions

We have applied a novel approach to mesocyclone and tornado detection. In the forecasting of tornadoes, accuracy or detection, length of lead time and a low false

alarm rate are crucial elements for success. If mesocyclones are predicted with an algorithm containing these attributes, that will help to minimize the loss of life. Comparison of the four methods (MDA, DA, NN, SVM) has been performed. Currently, the WSR-88D Doppler radar uses the MDA rule based algorithm and NN. These existing algorithms have moderate detection probabilities and moderate false alarm rates. The moderate FAR is particularly insidious, as it tends to lull the public into a false sense of complacency concerning tornado warnings. The SVM algorithm is the most accurate algorithm in terms of the highest values of CSI, POD and the lowest of FAR (less than half the FAR of the NN technique for one SVM model tested). In order to improve the capability of WSR-88D Doppler radar to detect mesocyclones, this work has established that the SVM algorithm can be used successfully. Accordingly, SVM should be tested more fully to determine if these results generalize well in other situations and geographical locations. We are currently investigating refining SVM modeling to improve these results with data being assimilated from an array of radars.

Acknowledgment. This research is partially funded by NSF EIA-0205628. The authors would like to thank Greg Stumpf for providing the datasets and Alexander Malyscheff for helping with the final manuscript.

References

1. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Classification. Data Mining and Knowledge Discovery **2**(2) (1998) 121–167
2. Collobert, R., Bengio, S.: SVMTorch: Support vector machines for large-scale regression problems. Journal of Machine Learning Research **1** (2001) 143–160
3. Cortes, C., Vapnik, V.: Support Vector Networks. Machine Learning **20** (1995) 273–297
4. Girosi, F.: An Equivalence Between Sparse Approximation and Support Vector Machines. Neural Computation **10**(6) (1998) 1455–1480
5. Marzban, C. , Stumpf, G.J.: A Neural Network for Tornado Prediction Based on Doppler Radar-Derived Attributes. Journal of Applied Meteorology **35**(5) (1996) 617–626
6. Osuna, E., Freund, R., Girosi, F.: Training Support Vector Machines: An Application to Face Detection. Proc. Computer Vision and Pattern Recognition '97 (1997) 130–136
7. Pontil, M., Verri, A.: Properties of Support Vector Machines. Technical Report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory (1997)
8. Rifkin, R.: SvmFu a Support Vector Machine Package. http://five-percent-nation.mit.edu/PersonalPages/rif/SvmFu/index.html (2000)
9. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press, Cambridge, Massachusetts (2002)
10. Stephenson, D.B.: Use of the "Odds Ratio" for Diagnosing Forecast Skill. Weather and Forecasting **15**(4) (2000) 221–232
11. Evgeniou, T., Pontil, M., Poggio, T.: Regularization Networks and Support Vector Machines. Advances in Computational Mathematics **13** (2000) 1–50

12. Trafalis, T.B., Evgeniou, T., Ince, H.: Hierarchical Methods for Training Support Vector Machines with Very Large Datasets. In: Proceedings of the 30th International Conference on Computers and Industrial Engineering, Tinos Island, Greece (2002)
13. Trafalis, T.B., White, A., Fras, A.: Data Mining Techniques for Tornadic Pattern Recognition. In: C.H. Dagli, A.L. Buczak, J. Ghosh, M. Embrechts, O.Ersoy, and S. Kercel, editors, Intelligent Engineering Systems Through Artificial Neural Networks **10** ASME (2000) 455–460
14. Vapnik, V.: Estimation of Dependencies Based on Empirical Data. Springer Verlag (1982)
15. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
16. Wahba, G.: Splines Models for Observational Data. Series in Applied Mathematics **59** SIAM (1990)
17. Wilks, D.S.: Statistical methods in the Atmospheric Sciences. Academic Press (1995)

Computational Science Simulations Based on Web Services

Paul Chew[1], Nikos Chrisochoides[5], S. Gopalsamy[4], Gerd Heber[6], Tony Ingraffea[2], Edward Luke[3], Joaquim Neto[2], Keshav Pingali[1], Alan Shih[4], Bharat Soni[4], Paul Stodghill[1], David Thompson[3], Steve Vavasis[1], and Paul Wawrzynek[2]

[1] Department of Computer Science, Cornell University
[2] Department of Civil Engineering, Cornell University
[3] Engineering Research Center, Mississippi State University
[4] School of Engineering, University of Alabama at Birmingham
[5] College of William and Mary
[6] Cornell Theory Center

Abstract. We describe the software architecture of a system for doing multi-physics simulation of a coupled fluid, thermal, and mechanical fracture problem. The system is organized as a collection of geographically-distributed software components in which each component provides a web service, and uses standard web-service protocols to interact with other components. The resulting system incorporates many features such as componentization and geographical distribution which we believe are vital to adaptive and dynamic data-driven application systems (DDDAS).

1 Introduction

Dynamic Data Driven Application Systems (DDDAS) ([10]) are systems in which computational simulation is coupled with real-world experimental data. One example of a DDDAS system is a weather forecasting simulation that periodically uses field observations for corrections. To build a true DDDAS system, we must be able to build systems that adapt continuously to input from sensors and observers, and to changes in computational requirements as the simulation progresses.

One of the goals of the Adaptive Software Project (ASP)[7] is to lay a foundation for building such adaptive systems. In our work, we have identified three levels at which adaptivity occurs in computational science simulations.

Application-level A number of mathematical models (discrete, continuum, etc.) may be available to describe the science of a given problem. A simulation code may find it advantageous to switch adaptively between such models to trade off accuracy for computational time and resources.

Algorithm-level There may be many algorithms for implementing a desired functionality (*e.g.*, direct and iterative solvers for linear systems), and it may be advantageous to switch between algorithms.

[7] Additional information about the ASP project can be found at `http://www.asp.cornell.edu/`.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 299–308, 2003.
© Springer-Verlag Berlin Heidelberg 2003

System-level Changes in the computational environment such as hardware failures may require a simulation system to be adaptive at the system-level.

In this paper, we describe the ASP system which exhibits these three types of adaptivity. In Section 2, we introduce a multi-physics fracture problem from the aerospace domain that we use as a challenge problem. In Section 3, we describe our simulation system, which is organized as a collection of components that interact using web-services implemented on top of standard web-service protocols. We also present the pros and cons of such an architecture. In Section 4, we present some preliminary performance measurements that show that the performance overhead of using web-services is relatively small. We discuss related work in Section 5, and conclude in Section 6.

2 Overview of the Problem

The applications engineers on our research team work in computational fluid mechanics and solid mechanics, so we decided to tackle a problem involving high Reynolds number, chemically reacting gas flow coupled with linear elastic fracture mechanics.

The geometry shown in Figure 1 represents an idealized segment of a rocket engine modeled after actual NASA experimental spacecraft hardware. The object is a curved, cooled pipe segment that transmits a chemically reacting, high-pressure, high-velocity gas through the inner, large diameter passage, and cooling fluid through the outer array of smaller diameter passages. The curve in the pipe segment causes a non-uniform flow field that creates steady-state but non-uniform temperature and pressure distributions on the inner passage surface. These temperature and pressure distributions couple with non-uniform thermomechanical stress and deformation fields within the pipe segment. In turn, the thermomechanical fields act on an initial crack-like defect in the pipe wall (see Figure 2), causing this defect to propagate.

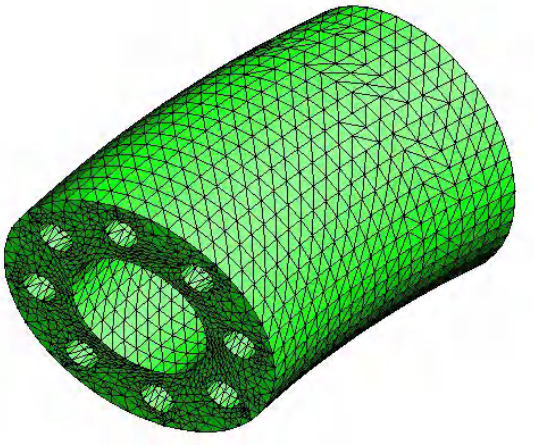


Fig. 1. The Pipe

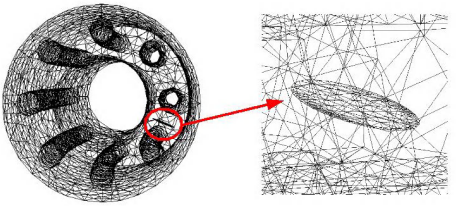


Fig. 2. The Pipe with Crack

2.1 The Workflow

The workflow of a single time step of the simulation is shown in Figure 3. In this figure, the components of our system appear like this, the intermediate data sets appear like *this*, and the "human in the loop" is appear like this. In our current workflow, the only data that is passed from one timestep to the next is the geometric model of the pipe, which is updated in each time step as the defect is inserted and grown.

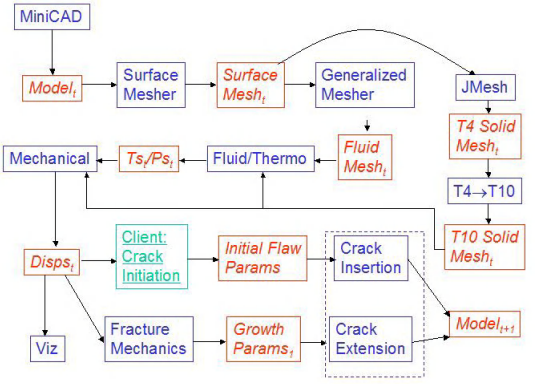


Fig. 3. Workflow for the Pipe problem

Crack initiation is an active area of research in Fracture Mechanics, and, at present, is not understood well enough to do automatically. Hence, in our present system, we require a knowledgeable user to manually determine the Initial Flaw Parameters by studying the displacement field at the end of time step $t = 0$. This is shown as a component labeled `Client: Crack Initiation` in Figure 3. In subsequent timesteps,

t=1,2,..., state of the art fracture mechanics techniques are used to predict growth parameters that determine how the crack defect will grow. At present, a complete simulation of the Pipe Problem consists of three time steps because at the end of the third step, we find that the crack reaches the surface of the component, possibly leading to failure.

2.2 Components

We now describe the components used in the simulation. Many of these components were written as part of the ASP project.

MiniCAD MiniCAD is an integrated environment for creating geometry and topology. It uses Nonuniform Rational B-Splines (NURBS) to represent geometry surfaces. MiniCAD offers several advantages over traditional CAD systems, including being able to guarantee the "watertightness" of the models that it produces. MiniCAD's output is a geometric model in an XML-based representation developed by our project [8].

Surface Mesher Once the geometry model is produced, it is passed to the surface mesher component which produces triangular meshes for each of the model's geometric surfaces. This component produces surface meshes with certain quality guarantees [7].

Generalized Mesher Meshes and grids employed for simulating viscous fluid flows require highly anisotropic elements in regions near no-slip boundaries, *i.e.* boundary layers. For such problems, topological adaptivity - using cell types that are locally appropriate for the region being discretized - offers an attractive alternative to traditional structured multi-block grids or unstructured tetrahedral meshes. Chalasani *et al* [6,5] have exploited this idea to generate high quality meshes consisting of extruded triangular prisms, tetrahedral elements, and generalized prisms.

Jmesh Jmesh [4] generates unstructured tetrahedral meshes for arbitrarily shaped three-dimensional regions, and was designed to handle the unique geometric problems that occur in fracture mechanics. The input for Jmesh is a triangular surface mesh, which describes the domain to be meshed, and the output is a tetrahedral mesh for the solid part of the pipe.

T4 to T10 The "T4 to T10" component converts the volume meshes produces by Jmesh, which use four-noded tetrahedra, into equivalent meshes of ten-noded tetrahedrons.

Fluid/Thermal Simulation Loci [15,14] is a framework for intra-application coordination of fine-grained numerical kernels and methods. The CHEM code [15,16] is a library of Loci rules (fine-grained components) and a front end that generates a component that simulates 3-D chemically reacting flows of thermally perfect, calorically imperfect gases.

Mechanical Simulation The mechanical solver solves the equations of linear elasticity to determine the deformation of the pipe due to different loading conditions (e.g. pressure on the inner pipe) and thermal expansion.

Fracture Mechanics The fracture mechanics component takes as input the volume mesh for the solid part of the pipe and the the nodal displacements computed by the mechanical simulation. It computes the new crack front. This component, as well as a number of other components, uses GGTK [1], a library for manipulating geometric models and for performing geometric operations.

Visualization We have developed an innovative real-time visualization tool for interactive exploration of large-scale 3D solid models and underlying engineering data. Users guide dynamic data extraction by manipulating visual probes and selectors. These choices are automatically translated by the visualization system into SQL queries, which are sent to a parallel database server cluster and which return new features to the user's display.

3 Using Web Services for Simulation

In this section, we describe how the components in our system communicate with each other. Briefly, each component provides web services, implemented using standard web-service protocols, and these services are invoked by any other component that needs to interact with that component.

3.1 Why Components?

Components and component frameworks are critical for adaptive and DDDAS systems. Consider, for example, instances of algorithmic adaptivity in which the application switches from one technique to another. If the two implementations of these techniques did not have clearly defined interfaces or did not use similar parameter (or data) types, then it would be impossible to switch between these implementations dynamically.

3.2 Why Distributed?

It is likely that many instruments and visualization tools that would be used in a production DDDAS system are geographically distributed. Another reason is that in a multidisciplinary, multi-institutional project like ours, there are likely to be many different architectures and operating systems in use. It would be a tremendous burden if every developer had to port their code to every other platform. Ideally, a component would be implemented on just one platform, and it would be invoked as needed by project partners. In other words, components should be *write once, run* from *anywhere*.

This has a number of advantages. For the component developer, intellectual property issues become less critical because source code does not have to be released. For component users, the advantage is that they do not have to download source code and install components, nor do they have to find sufficient computational resources to run the components.

3.3 Distributed Components Lead to Web Services

The W3C Web Services Architecture Working Group defines the term "web service" as ([12]),

> A web service is a software system identified by a URI [RFC 2396], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

We have found that frameworks for building and deploying web services are suitable for developing our adaptive simulation systems. In particular, the current system for solving the pipe problem has been developed and deployed using the following web services frameworks.

Microsoft .NET On our Windows platforms, we use Microsoft .NET ([9]), which provides a "holistic" approach to distributed applications.

SOAP::Clean On our UNIX and Linux platforms, we use SOAP::Clean ([20]), a Perl module for exposing legacy applications as web services. Compared with .NET, SOAP::Clean provides a "minimalistic" approach to distributed applications. It is designed to allow existing command-line applications to be exposed as web services after writing a handful of lines of "glue" code, . It also provides client tools that make remote web services appear as local command-line programs.

4 Performance

We have implemented the workflow shown in Figure 3 by deploying each component as a web service on a number of different computers at Cornell Computer Science (CU CS), the Cornell Theory Center (CTC), the Engineering Research Center at Mississippi State University (MSU ERC) and the College of William and Mary (CW&M), as shown in Table 1. The entries that are boxed denote the specific instances of components that were used in these experiments.

4.1 Results

Table 2 shows the individual and total execution times taken by running each of the components using a number of different methods. All times are given in minutes. The columns denote the following execution methods,

Local, non-XML Some of our components consist of application programs that read and write non-standard ASCII file formats. In order to make them interoperable, we "wrapped" them with code that converted between these non-standard formats and our standard XML-based formats. This column contains the execution times of the original application programs running directly on a single machine (i.e., no web services involved). These times represent the expected performance of these modules within a traditional monolithic simulation system.

Table 1. Component deployment

Component	CU CS	CTC	MSU	ERC CW&M
Surface Mesher	yes	no	no	yes
Jmesh	yes	no	no	yes
T4_to_T10	yes	no	no	no
Generalized	yes	no	yes	no
Fluid/Thermal	yes	no	no	no
Mechanical	no	yes	no	no
Fracture Mechanics	yes	no	no	no
Crack Growth	yes	no	no	no

Table 2. Execution times for the Pipe problem

	Local, non-XML	Local, XML	Local, WS	Polling freq	Intra-campus	Interstate
Surface Mesher	1.10	1.10	1.15	0.17	1.33	1.43
Jmesh	17.08	16.92	16.98	5.00	20.27	20.67
T4_to_T10	n.a.	0.70	0.77	0.17	1.02	2.22
Generalized(*)	n.a.	0.57	0.57	0.17	0.67	1.35
Fluid/Thermal	23.00	24.13	28.05	5.00	25.42	28.92
Mechanical(*)	n.a.	16.87	17.45	1.00	18.75	n.a.
Fracture Mechanics	n.a.	0.65	0.72	0.17	1.05	n.a.
Crack Growth	n.a.	0.00	0.08	0.00	0.15	n.a.
Total Execution	-	60.97	65.82	-	68.73	-
Overhead	-	0%	8%	-	13%	-

Local, XML This column contains the execution times of all of the component programs, which all use our standard XML-based file formats. Again, these times are from directly running the component programs on a single machine without using web services. This column is used for the base times when computing overheads.

LOCAL, WS This column contains the execution times for invoking the components using web services. For all but two of the components, the web services client and server were run on the same machine, so it was possible for the two to communicate without having to go through the HTTP server. It was not possible to run the Generalized Mesher and Mechanical Simulation in this way, so these two were run according to the “Intra-campus” column.

Intra-campus This column contains the execution times obtained by putting the client on a different machine, within the same campus (Cornell) network, than the server. In this case, all of the communication between the client and the server has to go over a LAN and through an HTTP server. In order to prevent network connections from timing out, the client uses polling to determine when the server has completed execution of the component. The column marked "Polling freq" contains the polling frequency. The Generalized Mesher component resided at MSU ERC, so its time includes a slightly longer network delay.

Interstate This column contains executions times obtained by putting the client on a machine in a different state (Alabama) from all of the servers (Mississippi and New York). The same polling frequencies were used as in the "Intra-campus" cases. These times are incomplete, because we have not completed the experiments, but it should be clear from the "Jmesh" and "Fluid/Thermal" times that they are comparable to the "Intra-campus" times.

There are several observations that we can make from these results. First, we would argue that the total overhead of 13% in the "Intra-campus" case and similar overheads in the "Interstate" case are not excessive. In particular, paying the price of this overhead gives us all of the advantages of web services discussed earlier. Second, the bulk of this overhead is from using the web services frameworks (8%) and not Internet communication (5%) . This is very encouraging, because there are many places in our system (in SOAP::Clean in particular) where there is room for aggressive optimization. Third, we expect that as we scale our experiments to solve larger problems, this overhead should go down.

5 Related Work

A number of frameworks and standards have been proposed for developing component-based systems. Perhaps the best known are CORBA ([19]) and COM ([18]). We investigated using these frameworks, but they did not meet our needs. Using these frameworks required major reengineering of our existing applications to incorporate them into the frameworks. We also found that the existing frameworks were primarily designed for deploying applications within a single machine. DCOM ([17]) is one exception to this. It is also interesting to note that existing component frameworks are evolving towards interoperability with web services (witness .NET subsuming COM and DCOM, and the OMG's adoption of a specification on CORBA-WSDL/SOAP Interworking).

Perhaps the most widely know paradigm for distributed scientific computing is Grid Computing [11] and the associated Globus Toolkit [2]. We did not follow this approach for a number of reasons. The most fundamental reason was that none of the Grid software provided server-side functionality under Microsoft Windows. Given that the Cornell Theory Center has many hundreds of cluster nodes running Windows, this was a show-stopper[8]. Just as importantly, standards for Grid frameworks are only now be-

ing developed whereas the web services community has already adopted a number of standards (e.g., XML, SOAP, WSDL) that enable frameworks from different vendors to interoperate.

This is not to say that we will not use Grid computing frameworks at some point in the future. The Core Grid Functionality ([13]) contains many features that we have not implemented within our web services frameworks, and leveraging them seems prudent. Furthermore, the Grid standards being proposed leverage web services functionality such as WSDL. We see Grid and web services converging in some way in the future, although in what form we are not sure.

6 Conclusions

We have argued that component design and distributed computing are fundamental to adaptive and DDDAS simulations, and we have argued that web services provide a natural means of achieving these characteristics. We have described a multi-physics, adaptive simulation system developed along these lines. Preliminary performance results indicate that the overhead of using web services for distributed simulation is not unacceptably high, and certainly worth the benefits and flexibility they give us.

References

1. GGTK home page. Available at "`http://www.erc.msstate.edu/ccs/research/GGTK/`". Accessed February 13, 2003.
2. The Globus Project. Available at "`http://www.globus.org`". Accessed February 13, 2003.
3. Globus Project, Microsoft expand partnership. Available at "`http://www.anl.gov/OPA/whatsnew/020315globus.htm`". Accessed February 13, 2003.
4. J.B. Cavalcante-Neto, P.A. Wawrzynek, M.T.M. Carvalho, L.F. Martha, and A.R. Ingraffea. An algorithm for three-dimensional mesh generation for arbitrary regions with cracks. *Engineering with Computers*, 17:75–91, 2001.
5. S. Chalasani and D. Thompson. Quality improvements in extruded meshes using topologically adaptive generalized elements. *International Journal for Numerical Methods in Engineering*, (submitted).
6. S. Chalasani, D. Thompson, and B. Soni. Topological adaptivity for mesh quality improvement. In *Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, Honolulu, HI, June 2002.
7. L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the Ninth Symposium on Computational Geometry*, pages 274–280. ACM Press, 1993.
8. L. Paul Chew, Stephen Vavasis, S. Gopalsamy, TzuYi Yu, and Bharat Soni. A concise representation of geometry suitable for mesh generation. In *Proceedings, 11th International Meshing Roundtable*, pages pp.275–284, Ithaca, New York, USA, September 15-18 2002. Available at "`http://www.imr.sandia.gov/papers/imr11/chew.pdf`".

[8] This issue is being addressed[3]

9. Microsoft Corporation. Microsoft .NET. Available at "http://microsoft.com/net/". Accessed February 11, 2003.
10. Craig Douglas, Abhi Deshmukh, et al. Report from the March 8-10, 2000 NSF sponsored workshop on Dynamic Data Driven Application Systems. Available at "http://www.cise.nsf.gov/eia/dddas/dddas-workshop-report.htm". Accessed February 8, 2003.
11. Global Grid Forum. Global Grid Forum home page. Available at "http://www.ggf.org/". Accessed February 13, 2003.
12. W3C Web Services Architecture Working Group. Web services glossary W3C working draft 14 November 2002. Available at "http://www.w3.org/TR/ws-gloss/". Allen Brown and Hugo Haas, eds. Accessed February 8, 2003.
13. W. Johnston and J. Brooke. Core Grid Functions: A minimal Architecture for Grids. GGF Draft Document. Available at "http://www.ggf.org/meetings/ggf6/ggf6\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}wg\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}papers/CoreGridFunctions.v3.pdf". Accessed February 13, 2003.
14. E. Luke. Loci: A deductive framework for graph-based algorithms. In S. Matsuoka, R. Oldehoeft, and M. Tholburn, editors, *Third International Symposium on Computing in Object-Oriented Parallel Environments*, number 1732 in Lecture Notes in Computer Science, pages 142–153. Springer-Verlag, December 1999.
15. E. A. Luke. *A Rule-Based Specification System for Computational Fluid Dynamics*. PhD thesis, Mississippi State University, 1999.
16. E. A. Luke, X.L. Tong, J. Wu, L. Tang, and P. Cinnella. A step towards "shape-shifting" algorithms: Reacting flow simulations using generalized grids. In *Proceedings of the 39th AIAA Aerospace Sciences Meeting and Exhibit*. AIAA, 2001. AIAA-2001-0897.
17. Microsoft, Inc. Distributed component object model (DCOM). Available at "http://www.microsoft.com/com/tech/dcom.asp". Accessed February 13, 2003.
18. Microsoft, Inc. Microsoft COM technologies. Available at "http://www.microsoft.com/com/". Accessed February 13, 2003.
19. Object Management Group, Inc. Welcome to the OMG's CORBA website. Available at "http://www.corba.org/". Accessed February 13, 2003.
20. Paul Stodghill. SOAP::Clean, a Perl module for exposing legacy applications as web services. Available at "http://www.asp.cornell.edu/SOAP-Clean/". Accessed February 11, 2003.

DDEMA: A Data Driven Environment for Multiphysics Applications

John Michopoulos[1], Panagiota Tsompanopoulou[2], Elias Houstis[3,2], John Rice[3], Charbel Farhat[4], Michel Lesoinne[4], and Frederic Lechenault[4]

[1] U.S. Naval Research Laboratory, Special Projects Group, Code 6303
Washington, DC 20375, U.S.A.
john.michopoulos@nrl.navy.mil
[2] University of Thessaly, Dept. of Comp. Eng. and Telecommunications
38221 Volos, Greece
yota@inf.uth.gr
[3] Purdue University, Computer Sciences Department
W. Lafayette, IN 47906, U.S.A.
{enh, jrr}@cs.purdue.edu
[4] University of Colorado at Boulder, Dept. of Aerospace Engineering Sciences
Boulder, CO 80309-0429, U.S.A.
{farhat, michel, lechenault}@colorado.edu

Abstract. In this paper we present the design of a multidisciplined problem solving environment (MPSE) for supporting an efficient prediction capability for the response of multiscale interdisciplinary continuous interacting systems. This design takes into consideration information technologies, coupled multiphysics sciences, and data-driveness to steer adaptive modelling and simulation of the underlying systemic behavior. The paper describes the design objectives and software architecture of DDEMA in the context of two multidisciplinary applications related to material/structure design of supersonic platforms and fire/material/environment interaction monitoring, assessment and management.

1 Introduction

Continuous interacting systems under the influence of coupled multifield loading can exhibit static and dynamic behavior that is highly variable in space and time, spans multiple scales, and sometimes has intense nonlinear interaction and response. This is true especially for material/structural systems embedded in host continua as in the case of structures interacting with surrounding fluids under generalized loading conditions. The development of the multiphysics science and technology that allows predicting behavioral characteristics of such interacting systems is of utmost importance when it comes to their applications in contexts driven by human needs. Examples of such applications are: a) Energy storing and dissipating composite materials for supersonic and hypersonic multi-mission conventional or uninhabited air vehicles for civil and defense applications. b) Materials and structures for the civil and defense infrastructure life extension and

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 309–318, 2003.
© Springer-Verlag Berlin Heidelberg 2003

maintainability, such as smart materials and systems that can self-adjust when a scheduled or unexpected restriction of their state space occurs. An example would be high temperature structural composites that can change their mechanical properties when exposed to sudden temperature gradients due to explosion or mechanical damage.
Specific vertical industries must comply with consumer driven quality standards and/or demands that introduce implicit requirements for reducing uncertainty of simulated behavior. This has increased the crucial aspect of the validation and verification of predictions, rendered by the computational environments used for this purpose, up to the point of becoming one of the main areas if intensive work. The main objective of our project is to develop a framework for applying specific Multidisciplinary Problem Solving Environments (MPSEs) to multidisciplinary applications. An instantiation of this framework will be a particular MPSE, which will provide simulation based behavior prediction of fully identified systems as a consequence of applying inverse approaches along with massive automated data acquisition during tests of continuous systems where the tests are exhaustive of multidimensional stimulus-response spaces. Such a data-driven framework will allow the experimental capturing of the multiphysics behavioral characteristics of the continuous system, thus intrinsically ensuring the verifiability and validity of simulations based on the testing. To develop and validate, within the framework provided by this objective, the proposed data driven distributed MPSE will employ appropriate methodologies that will demonstrate two main features. The first feature is a software encapsulation of multiphysics data-driven constitutive characterization and model selection and encapsulation. The second feature is heavy- and light-weight computational solution implementations and comparison for solving of multi domain coupling (i.e. aero-structural interaction) problems. We will be referring to this data-driven MPSE as a Data Driven Environment for Multiphysics Applications (DDEMA) thereafter.

2 MPSE Overview and DDEMA Specifications

Progress towards the solution of coupled multiphysics and multi-continua interaction both from the constitutive and the field equation perspectives has been scarce. There is significant progress however, when focus has been given to computational implementations for a small number (mostly three) of interacting fields and a number of active projects exist. Among them one can include our development of the GasTurbLab [1], [2] multidisciplinary problem solving environment (MPSE) that addresses the interactions between a stator, rotor, and combustor in a gas turbine engine that has been implemented on a mobile agent platform, utilizing a collaborating partial differential equations (PDEs) methodology [3], [4]. In the area of fluid-structure-control interaction, we can mention the advanced AERO computational platform. This platform comprises the AERO-F, AERO-S, AERO-C and MATCHER suite of codes, [5],[6], developed at the University of Colorado for the solution on nonlinear transient aeroelastic problems. They are portable, and run on a large variety of comput-

ing platforms ranging from Unix workstations to shared as well as distributed memory massively parallel computers. A generalization of AERO to an aerothermoelastic implementation was recently achieved [7]. The case of aerostructural coupling with an energy dissipative non-linear material response under a virtual wind tunnel environment has also been initiated recently [8], [9]. The DDEMA system will provide high confidence prediction capability through its data-driven system modeling capability.

The DDEMA system will use network based distributed computing infrastructures and data and will provide a web-accessible user's environment. The envisioned computing infrastructure will have the following properties: (1) Computation will be partitioned into coarse-grain and fine-grain components and will be executed in a loosely coupled fashion for the lightweight implementation, and in tightly coupled fashion for the heavyweight implementation. (2) The system will distribute dynamically the various components in order to adapt according to resource availability and performance variations. (3) To enable automatic adaptive modelling, the system will be minimizing the difference between stored behavioral datasets and predicted behavioral datasets, in a design optimization fashion, where the objective function and the potential constraints will be user or system definable. (4) To enable optimal user customization and history maintenance of user-system interaction, a visual problem definition editor (VPDE) will be developed and its user-invoked states will be archiveable and unarchiveable (or serializable and deserializable) upon request. (5) To enable collective and dynamic knowledge transfer among distributed points of entry and achieve auto enhancement of simulation due to historical collective user experience, abstractions of successful simulations and/or problem solutions as well as their least upper bound (as a representation of the non conflicting historic common asserted knowledge) will be maintained for all users with appropriate authorization/authentication. (6) The system will deal with machine and programming language heterogeneity problems while it will preserve platform independent component communication by leveraging efficient and ubiquitous data exchange formats and methodologies.

The proposed work will borrow design concepts from or build on top of: (1) existing agent systems (e.g. [10]), (2) communications tools that achieve interoperability between different platforms (e.g. Java-RMI-IIOP [11]), (3) symbolic algebra systems and sub systems for automating the field equation generation process (e.g. MathTensor [12]), (4) flexible APIs for visual editing of user controlled problem definition as well as 2D and 3D visualization (e.g. PTOLEMY-II [13]), (5) high performance Java implementations for parallel computation over distributed and shared memory architectures focused on intra-paradigm compatibility (e.g. ProactivePDC [14]) or message passing efficiency (e.g. Hyperion [15]). Many of these technologies have already been proven to be appropriate for rapid prototyping, involving traditional multiphysics and multi-scale models developed from different disciplines and organizations. What is crucially important for this project is the clean, simple and flexible programming model of mobile or static component systems, which is expected to facilitate the development of the

overall envisioned system. Besides making necessary adjustments/extensions to such systems, our IT development work will also focus on transforming proven but largely monolithic legacy codes into a set of well integrated, interoperable components (via wrappers based on the Java Native Interface (JNI) [16]) that can be distributed over the network.

3 Software Architecture of DDEMA Based on Data-Driveness Time Modalities

The application domain for the validation of utility for DDEMA will be that a) of composite material structures in their linear and nonlinear behavioral regimes when they interact with surrounding fluids under multiphysics loading conditions primarily for material/structural design of supersonic platform and b) fire-material-environment interaction, monitoring, assessment and management system.

The two application scenarios considered can be characterized by the following data-driveness time modalities which influence the software architecture of DDEMA. We have identified the following two time modalities:

- *Real time* (unpredictable) data driveness where data, originating from sensor networks, effect systemic operational assessment of integrity and utility measures (i.e. fire effects on environment and structural/material health).
- *Prior time* (operator controllable) data driveness where massive data from automated experimentation are used to select, calibrate and construct behavioral models for materials/structures and fluids related to design, qualification and prediction scenarios (i.e. design of a nonlinear supersonic platform). This can also have a real time component as well especially if model selection is left to be determined by incoming sensor data both on laboratory and on-board platform (ships, submarines etc) situations.

Throughout, we will use these modalities to describe the two proposed DDEMA application architectures.

3.1 Real-Time System

In this system configuration, we assume that the on-board user has the many operational objectives some of which are:

- Monitor the health state of the area (e.g. entire platform, substructure, component) of interest by observing continuous field distributions guided by incoming sensor datastreams originating from a distributed sensor network that can be monitored from multiple points of entry.
- Simulate potential future states ("what-if" scenarios) of the system given certain hypothetical battle or situational escalation scenarios. This requires auto conversion of actual operating conditions to corresponding boundary/initial conditions, exploitation pre-computed field states to synthesize potential field evolutions quickly for "what-if" scenarios.

– Based on input from the previous two items, the on-board users will be able to make decisions and implement them by activating actuation mechanisms. These require a dynamic analysis of simulated scenarios based on controllability of the systems,an impact analysis via a system that implements heuristic knowledge management techniques, and an archival capability for future re-usage and recording evaluative experiences of implemented decisions.

A high level schematic view of the so called real-time system architecture is depicted in Fig. 1, where the coarse grain functionality modules have been shown from a data association perspective only. This diagram should be implementable within the VPDE. Space limitations require that we do not expand on the definition of each one of the modules presented.

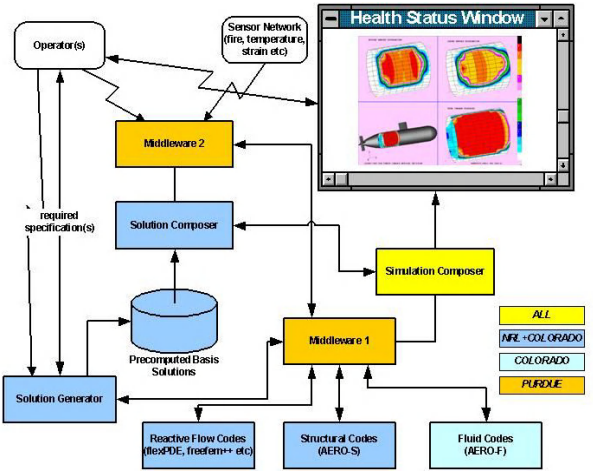


Fig. 1. A real-time system architecture.

3.2 Prior-Time System

In this system configuration, we assume that the laboratory or off-board user, who can be a structural designer or system qualifier who needs accurate systemic state behavior prediction, has similar operational objectives to that of the real-time modality. The only difference will be that the real-time component will not be present. The role of the incoming data will be played by the multidimensional material testing data-streams for identifying constitutive behavior of materials that may also guide model selection and implementation.
This requirements specification represents the structural system design and qualification utilization of the system. The high level schematic view of the so called prior-time system architecture is depicted by appropriate adjustments of the architecture depicted in Fig. 1.

4 The Design of Middleware and User Interface

The plethora of available choices on computational infrastructure (hardware, software, networking), see Sec. 2, along with specific domain of application and user specific backgrounds and needs have introduced a large number of research issues associated with the computational implementation of any MPSE. The major of these issues are: (1) ability and optimization for computation over distributed resources, (2) ability for dynamic migrating component distribution, (3) adaptive modeling capability, (4) user dependent customization, (5) collaborative knowledge representation and interactivity, (6) dealing with heterogeneity of legacy and new code, and finally (7) ability to sustain component fault tolerance. To address several of the above issues, in the context of the data-driveness and the two time modalities scenarios considered, the following middleware design is proposed.

4.1 DDEMA Middleware

The design of the middleware is based on our experience from previous projects implementation, such as GasTurbinLab [1]. It is presented in the context of an agent-based middleware. New packages (e.g. Ptolemy II, [13]) are based on a combination of computational domain "director" and "actor" dual concepts. Actors can be thought as metamorphic agents with a visual incarnation in a GUI, and therefore include the concept of the agent and accordingly we promote their use in DDEMA.

Based on the issues, requirements, and data associations between functional abstractions as they have been described above, we have decided on proposing the following set of actors. A Director (or General Control Actor (GCA)), a Finite Element Tear and Interconnect Actor (FETIA), a Communication Actor (CA), Domain Actors (DA), Database Actor (DBA), and many I/O Actors (e.g. Display Actor, Printer Actor, Sensor Actor, User Actor), see Fig. 2.

Each Actor has specific tasks and interactions with some of the other Actors. However, we will not discuss here the individual responsibilities of each actor/agent due to the lack of space and we will allow the reader to infer some of this functionality through the associations depicted in Fig. 2

In many cases the actors to be developed will have to be constructed such as they encapsulate of existing codes like the AERO suite [5],[6] or their subcomponents.

4.2 DDEMA Realization and User Interface/VPDE

DDEMA's realization will be based on two main operational modes: The applications design and the application use modes.

In the application design mode the designer will utilize the VPDE for designing the actual application architecture in terms of data flow and message passing diagram. In particular, a visual representation of the linguistic components available for composition through their visual incarnations within the VPDE will be used. This is exactly why we plan to utilize an appropriately modified version

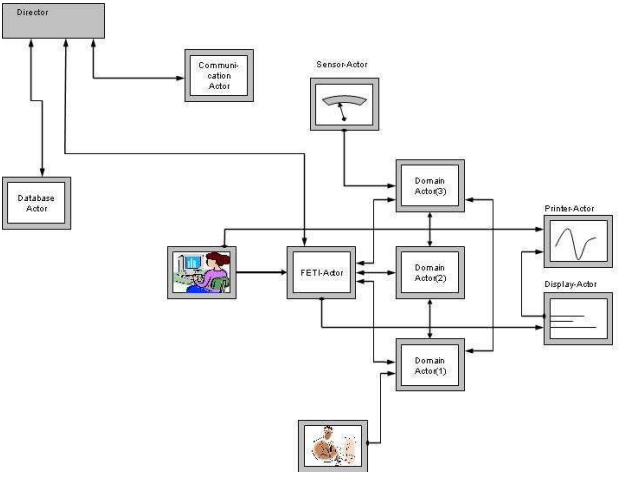


Fig. 2. Middleware design in Ptolemy's Vergil used as a VPDE.

of the "Vergil" visual editor paradigm provided by the Ptolemy II ([13]), system which already contains a well documented specification for creating new actors and directors. In this mode of usage DDEMA's GUI will be that of the VPDE based on Ptolemy-II.

In the application use mode, the user will take action to initiate the automatic generation of java source code or bytecode that implements the application functionality intended. Upon execution of the bytecode, the user will be presented with a stand alone custom made application that is the result of the previous mode to perform the activities associated with the corresponding usage requirements. During this phase secondary process will be spawned on the appropriate lightweight and heavyweight computational infrastructure available at hand. In order to address issues of parallelism, distribution, concurrency, security and process mobility, we will first attempt to introduce an intermediate layer between the actual mission functional and actor view layers, that provides a unified set of resources for this goal such as Inria's ProActive library [14]. In this case the GUI will be the one defined by the user in terms of the previous stage.

5 Case Studies

5.1 Coupled Multiphysics of Continua: Material/Structural Design of Supersonic Platform

Our experience with continuous multiphysics field theories [17], [18] suggests that the design process of a structure often requires the usage of a tool such as DDEMA for establishing optimal material characteristics like fiber-matrix properties, fiber orientation, and laminate layup properties, and the shape tailoring for an aircraft - or an aircraft component such as a fuselage, a wing, or a control

surface - under high temperature and mechanical loading conditions inflicted from a supersonic mission requirement.
Intrinsic to the validity and confidence of any aero-structural simulation are the utilized continuous models of material behavior. It is exactly this area of where NRLs robotic testing capability along with historical exhaustive data pertaining to identification of material behavior, will be used as the springboard to launch an effort for automating model selection, implementation and verification methodologies. This will be done by considering all intrinsic and extrinsic, wanted and unwanted factors (uncontrolled biasing, noise, repeatability, history dependence etc) effecting data quality for the intended usage of model formation. Information theoretic, statistical and certainly deterministic techniques for model selection and/or generation in the context of continuous system identification and inverse approaches will be encapsulated in DDEMA. This will exercise the prior time data-driveness aspect of the system.
We have already demonstrated that it is possible to deal with most of these issues from a single physics rapid modeling perspective [8], [9]. Currently, the University of Colorado members of our team are part of a research consortium investigating the development of a supersonic aircraft shaping technology for reducing the initial shock pressure rise characterizing the ground signature. However, such a Quiet Supersonic Platform (QSP)requires a lightweight airframe, and will exhibit an aero-structural behavior. Its feasibility depends not only on the ability to design a shape that generates a low-boom ground signature, but most importantly on the ability to build a lightweight high-stiffness and damage tolerant structure that can withstand the expected aerodynamic and thermal loadings for long-range missions at a sustained rate. Therefore, the final design of a QSP will have to rely on a multidisciplinary aero-thermo-structural optimization. The DDEMA provides a unique opportunity to complement our current QSP effort. From the real time modality perspective, an on board version of a particular runtime version of DDEMA can be utilized to exploit sensor originated datastreams for helping responsible personnel to take decisions about restrictions or extensions of the platform operational envelope based on data driven simulation of potential outcomes of structural health metrics such as dissipated energy density. More details of how this expression of DDEMA can evolve is further expanded below for the case of our second application.

5.2 Monitoring, Assessment, and Management of Fire-Material-Environment Interaction

The ability to deal with the consequences of fire, extend over a wide range of application areas that have a direct effect on the survivability, ability to repair, maintainability, life extension and mission goals attainment of the environments and structural platforms effected by it. Some of these application areas can involve time critical situations that when they arise, demand decision-making support for an accurate monitoring capability supplemented by fire damage assessment and management and control countermeasures capabilities. A point in case are navy vessels build with a great variety of materials under extremely

demanding threat conditions such as possible catastrophic events due to fuel tanks proximity, limited oxygen supply (i.e. submarines), fire byproducts toxicity, structural and atmospheric damage.
To demonstrate the value of the core ideas behind the development of DDEMA, we plan develop a proof of concept instantiation of DDEMA within the context of managing all required activities of an accidental fire induced crisis scenario. More specifically, this application of DDEMA should be able to provide assistance to the users by employing the following features: (1) Ability to receive real time datastreams from multiple redundant distributed sensor networks that allow capturing of multimodal point or continuous distributions of various field quantities such as area/volume of live flames, temperature, chemical byproducts concentrations etc; (2)ability for multi-point-of-entry monitoring where this can be accomplished in multiple locations synchronously or asynchronously; (3) copresentation of reactive flow and reactive phase transformation simulation capability with multiphysics fluid-structure interaction simulation capability that allows "what-if" prediction exploration in order to be able to evaluate validity of decisions and alternative countermeasures; (4) a decision support subsystem, in order to combine sensor inputs, simulation results analysis and user choices based on experience and knowledge and form a course of action countermeasures that can also be simulated; (5) an interface to a control system of an existing countermeasure distributed actuation network, in order to implement the decided management strategy.
A system with these capabilities allows for portability and wide applicability.

Acknowledgement. The authors acknowledge the support by the National Science Foundation under grants EIA-0205663 and EIA-0203958.

References

1. Fleeter, S., Houstis, E.N., Rice, J.R., Zhou, C., Catlin, A.: GasTurbnLab: A Problem Solving Environment for Simulating Gas Turbines. Proc. 16th IMACS World Congress, (2000) No 104-5.
2. Houstis, E.N., Catlin, A.C., Tsompanopoulou, P., Gottfried, D., Balakrishnan, G., Su, K., Rice, J.R.: GasTurbnLab: A Multidisciplinary Problem Solving Environment for Gas Turbine Engine Design on a Network of Non-Homogeneous Machines. To appear in the Proceedings of 15th Toyota Conference, October 2001 and special issue on the J. of Computational Engineering and Mathematics.
3. Rice, J.R., Tsompanopoulou, P., Vavalis, E.A.: Interface Relaxation Methods for Elliptic Differential Equations. Applied Numerical Mathematics **32** (1999) 219–245.
4. Tsompanopoulou, P.: Collaborative PDEs Solvers: Theory and Practice. PhD thesis, Mathematics Department, University of Crete, Greece (2000).
5. Farhat, C., Lesoinne, M.: Two Efficient Staggered Procedures for the Serial and Parallel Solution of Three-Dimensional Nonlinear Transient Aeroelastic Problems. Computer Methods in Applied Mechanics and Engineering, **182**, (2000) 499-516.
6. Farhat, C., Lesoinne, M., Stern, P., Lanteri, S.: High Performance Solution of Three-Dimensional Nonlinear Aeroelastic Problems Via Parallel Partitioned Algorithms: Methodology and Preliminary Results. Advances in Engineering Software, **28** (1997) 43-61.

7. Tran, H., Farhat, C.: An Integrated Platform for the Simulation of Fluid-Structure-Thermal Interaction Problems, AIAA J. (in press.)
8. Michopoulos, J., Badaliance, R., Chwastyk, T., Gause, L., Mast, P., Farhat C., Lessoine, M.: Coupled Multiphysics Simulation of Composite Material Softening in a Virtual Wind Tunnel Environment, invited paper in Proceedings of Sixth U.S. National Congress on Computational Mechanics, U.S. Association for Computational Mechanics, Dearborn MI, (2001) pp. 521.
9. Michopoulos, J., Farhat, C., Lesoinne, M., Mast, P., Badaliance, R., Chwastyk, T., Gause, L.: Material Softening Issues in a Multiphysics Virtual Wind Tunnel Environment, AIAA Paper 2002-1095, 40th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, (2002).
10. The Grasshopper Agent Platform, IKV++ GmbH, Kurfurstendamm 173-174, D-10707 Berlin, Germany. http://www.ikv.de
11. Java-RMI-IIOP: http://java.sun.com/products/rmi-iiop/
12. Parker, L., Christensen, S.M.: MathTensor: A System for Doing Tensor Analysis by Computer, by Addison-Wesley, (1994)
13. Davis II, J., Hylands, C., Kienhuis, B., Lee, E.A., Liu, J., Liu, X., Muliadi, L., Neuendorffer, S., Tsay, J., Vogel, B., Xiong, Y.: Heterogeneous Concurrent Modeling and Design in Java, Memorandum UCB/ERL M01/12, EECS, University of California, Berkeley, CA USA 94720 March 15, 2001, http://ptolemy.eecs.berkeley.edu/ptolemyII/
14. Caromel, D., Klauser, W., Vayssiere, J.: Towards Seamless Computing and Metacomputing in Java, pp. 1043–1061 in Concurrency Practice and Experience, September-November 1998, 10(11–13). Editor Geoffrey C. Fox, Published by Wiley & Sons, Ltd., http://www-sop.inria.fr/oasis/proactive/
15. Antoniu, G.L., Bouge, l., Hatcher, P., MacBeth, M., McGuigan, K., Namyst, R.: Compiling multithreaded Java bytecode for distributed execution. In Euro-Par 2000: Parallel Processing. Lecture Notes in Comp. Science, Vol. 1900. Springer-Verlag, Munchen, Germany, (2000) 1039–1052.
16. Java Native Interface Specification. http://web2.java.sun.com/products/jdk/1.1/docs/guide/jni.
17. Sih, G.C., Michopoulos, J.G., Chou S.C.: Hygrothermoelasticity. Martinus Nijhoff Publishers (now Kluwer Academic) (1986).
18. Michopoulos, J.G., Sih, G.C.: Coupled Theory of Temperature Moisture Deformation and Electromagnetic Fields. Institute of Fracture and Solid Mechanics report IFSM-84-123, Lehigh University, (1984).

Discrete Event Solution of Gas Dynamics within the DEVS Framework

J. Nutaro, B. P. Zeigler, R. Jammalamadaka, and S. Akerkar

Arizona Center for Integrative Modeling and Simulation
University of Arizona
Tucson, AZ
{nutaro,zeigler,rajani,salila}@ece.arizona.edu

Abstract The DEVS (Discrete Event Systems Specification) formalism has been applied to continuous and discrete phenomena. The use of discrete events, rather than time steps, as a basis for simulation has been shown to reduce computation time by orders of magnitude in many applications. However, the application of DEVS to partial differential equation (pde) simulation has only recently been investigated. Here, in an application to a shockwave problem, we show that the time to solution is significantly reduced when a discrete event integration scheme is employed compared to a representative conventional approach. Recent theory suggests that speed advantages are to be expected for pdes that are characterized by heterogeneity in their time and space behavior. The implications for use of DEVS as a basis for adaptive control of large scale distributed simulations are discussed.

1 Introduction

The Discrete Event System Specification (DEVS) formalism provides a means of specifying a mathematical object called a system [7]. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given current states and inputs. Discrete event systems represent certain constellations of such parameters just as continuous systems do. For example, the inputs in discrete event systems occur at arbitrarily spaced moments, while those in continuous systems are piecewise continuous functions of time. The insight provided by the DEVS formalism is in the simple way that it characterizes how discrete event simulation languages specify discrete event system parameters. Having this abstraction, it is possible to design new simulation languages with sound semantics that easier are to understand. The DEVJAVA environment [3,8] is an implementation of the DEVS formalism in Java that enables the modeler to specify models directly in its terms. In this paper, we employ the adevs implementation in C++ [1]. A brief review of the DEVS concepts and formalism is provided in the Appendix.

The DEVS formalism has been applied to a number of continuous as well as discrete phenomena (e.g., see [2], [5]). The use of discrete events, rather than time steps, as a basis for simulation has been shown to reduce computation time by orders of magnitude in many applications. However, the application of DEVS to partial

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 319–328, 2003.
© Springer-Verlag Berlin Heidelberg 2003

differential equation (pde) simulation has only recently been investigated. In pdes, there is an interaction between time and space that is more intimate (through for example, the Courant condition) than in lumped parameter continuous models. Our research has been investigating DEVS solutions to relatively simple pdes while at the same time seeking indicators and supporting theory to predict where significant advantages may be gained.

2 Formulation of the DEVS Model

This paper describes a discrete event solution to a one dimensional gas dynamics problem. The goal is to solve a conservation law with state variable **u** in one spatial dimension *x* whose form is

$$\mathbf{u}_t + [f(\mathbf{u})]_x = 0.$$

A solution can be approximated by dividing the space *x* into discrete cells (grid points) and determining the flux of each state variable across a cell boundary (through a grid point). The motion of the solution is then computed by integrating over the net flux through a cell. To find the flux requires solving the Riemann problem with a particular solution being dependent on the physics of the model. To set up the discrete event approach we started with a conventional discrete time solution to the conservation law and the associated exact, iterative Riemann solver provided by Andrei A. Chernousov (source code available from http://www.geocities.com/andrei_chernousov). As in the latter approach, our discrete event approach employs Chernousov's Riemann solver. However, the essential difference between the two lies in the encompassing integration technique. Figure 1 shows the structure of the space-discretized form of the conservation law.

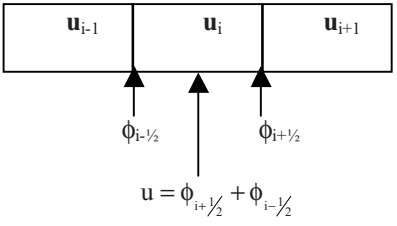


Fig. 1. Spatial discretization of the conservation law.

This system can be modeled by a coupled DEVS [7] that consists of three basic models. The first basic model computes the integral at each cell. It is based on the quantized integrator presented in [4], extended to integrate an N dimensional vector using a volume in N-space as the quantum description. The second basic model receives the outputs of the neighboring quantized integrators and computes the flux at a cell boundary using the exact, iterative Riemann solver. The third is a simple adder that computes the flux across a cell.

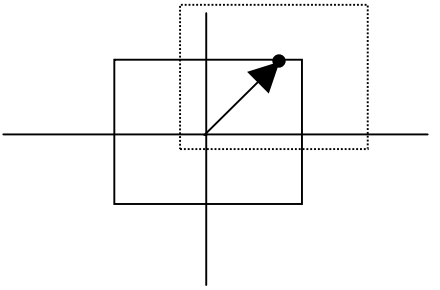


Fig. 2. A quantized integrator with rectangular quantum regions.

The quantized integrator works by drawing a box around the initial point **u**(0) whose dimensions are $D_1 \times D_2 \times \ldots D_N$, where D_i is the quantum size for the i^{th} element of **u**. Figure 2 depicts such an integrator operating in a two-dimensional state space. An output is produced *only* when **u**(t) reaches a boundary of the box. When this occurs, a new box is drawn around the point **u**(t) and the time advance is set to the time required to reach the boundary of this new box. The time advance is used to schedule the boundary crossing. A boundary crossing is represented by an internal event. It is important to note that no computation internal to cell occurs until the internal event is executed – unless an external event is received. An external event occurs when the output of a neighboring cell is received. The time advance after an external event takes into account the updated state as well as the remaining quantum to be consumed.

The discrete event model for this integrator can be formally described with the DEVS formalism as follows:

The state variables for the DEVS model are

q , the current value of the state variables,

$\mathbf{q}_l$, the value of the state variables at the last boundary crossing,

$\dot{\mathbf{q}}$, the first derivatives of the elements of **q**,

σ, the time until the next boundary is crossed.

The model is parameterized by the size of the state vector, denoted by N, and the vector **D** whose elements are the dimensions of the box. The input and output for the integrator are vectors of length N. Before describing the dynamics of the integrator, we need the function T(x,y,z,d) that computes the time to the next internal event in any one of the state dimensions. It is defined by:

$T(x,y,z,d) = \infty$ if $z = 0$

$T(x,y,z,d) = (d - |x - y|) / |z|$ if $z \neq 0$.

When x = y = 0, the situation prevailing after a just completed internal event, T returns the predicted time to reach the next box limit using derivative z. Following an external event, the arguments x and y represent the updated current state and the last

boundary crossing, respectively. Thus, $d - |x - y|$ is the quantum remaining to be consumed, and T returns the predicted time to consume this remaining quantity.

We also define a function $\Gamma(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{d})$ which returns the smallest time to the next event over all state dimensions. $\Gamma(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{d}) = \min \{ T(x_i , y_i, z_i, d_i) \}$ where **x**, **y**, **z**, and **d** are vectors with N elements and the index i ranges over N. With these definitions we have:

The time advance function is given by

$$ta(\mathbf{q}, \dot{\mathbf{q}}_l, \mathbf{q} , \sigma) = \sigma.$$

The state transition function is described by

$$\delta_{int}(\mathbf{q}, \dot{\mathbf{q}}_l, \mathbf{q} , \sigma) = (\mathbf{q} + \dot{\sigma}\mathbf{q} , \mathbf{q} + \dot{\sigma}\mathbf{q}, \mathbf{q} , \Gamma(\mathbf{0},\dot{\mathbf{0}},\mathbf{q} ,\mathbf{D})),$$

$$\delta_{ext}((\mathbf{q}, \dot{\mathbf{q}}_l, \mathbf{q} , \sigma),e,\mathbf{x}) = (\mathbf{q} + \dot{e}\mathbf{q} , \mathbf{q}_l, \mathbf{x}, \Gamma(\mathbf{q} + \dot{e}\mathbf{q} ,\mathbf{q}_l,\mathbf{x},\mathbf{D})), \text{ and}$$

$$\delta_{con}((\mathbf{q}, \dot{\mathbf{q}}_l, \mathbf{q} , \sigma), \mathbf{x}) = (\mathbf{q} + \dot{\sigma}\mathbf{q} , \mathbf{q} + \dot{\sigma}\mathbf{q} , \mathbf{x}, \Gamma(\mathbf{0},\mathbf{0},\mathbf{x},\mathbf{D})).$$

Finally, the output function is given by

$$\lambda(\mathbf{q}, \dot{\mathbf{q}}_l, \mathbf{q} , \sigma) = \mathbf{q} + \dot{\sigma}\mathbf{q} .$$

The boundary flux model has two state variables $\mathbf{q}_{left}$ and $\mathbf{q}_{right}$ that are the estimates of the state variables in the left and right cells, relative to the boundary. When the boundary flux model receives an input, it updates these values, computes the flux across the boundary, and then immediately outputs the new flux. The summers operate similarly by remembering two values (in this case the left and right fluxes) and responding to input by updating the two state variables and outputting the sum.

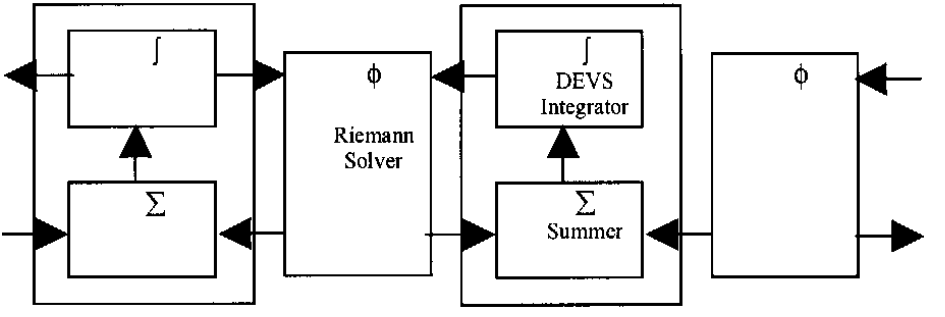


Fig. 3. A DEVS coupled model that approximates the conservation law.

Figure 3 depicts the coupled model that realizes the discrete event approximation based on the spatial discretization shown in Figure 1. The symbol ‡ denotes the quantized integrator, f is the summer, and ϕ is the Riemann boundary flux solver.

3 Implementation and Experimentation

The discrete event model and corresponding discrete time model were implemented in C++ and executed on a desktop computer with a 500 MHz AMD processor and 198M RAM running the Linux operating system. The discrete time model uses explicit Euler to integrate through time (see [4] for a comparison of explicit Euler and the first order explicit quantized integrator). As indicated, the same iterative Riemann solver was used for both implementations.

An initial value problem was solved using both techniques with identical discretizations in space and a fixed quantum size for the discrete event problem. The step size for the discrete time problem was taken to be the smallest value of the time advance function computed as a result of an internal or confluent event. This gives similar error bounds for both solution techniques. Figure 4 shows the initial conditions for the test problem.

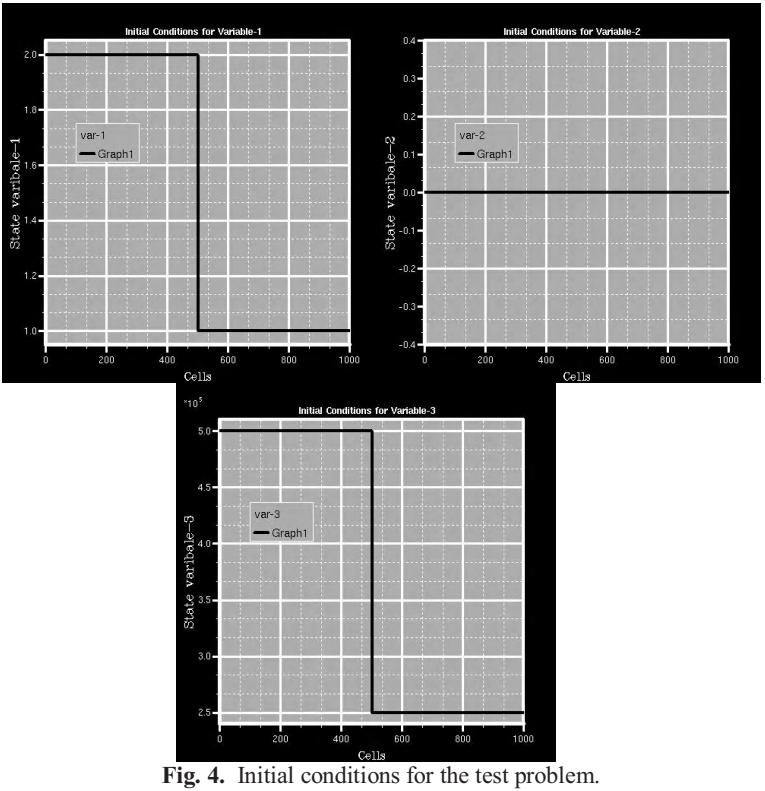


Fig. 4. Initial conditions for the test problem.

The simulation was run for 0.2 units of time. The solution generated by both techniques for a discretization in space using 1000 cells is shown in figure 5.

The allocation of computational effort for the discrete time and discrete event solution for the same 1000 cell run is shown in figure 6. It can be seen that the discrete event solution dynamically focuses computational effort on the highly active

portions of the solution as the shockwave moves outward (left figure). When all computations are accumulated, each active cell receives similar attention but this number is orders of magnitude less than it receives in the discrete time approach (right figure). Note that at the edges are cells to which activity has not propagated and which have therefore not been updated at all.

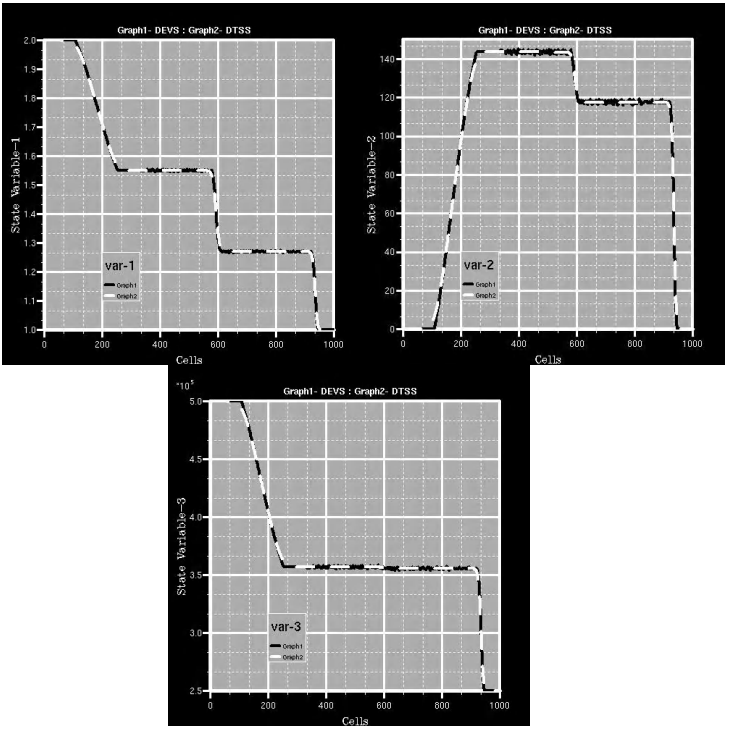


Fig. 5. Solution to the gas dynamics problem at t = 0.2.

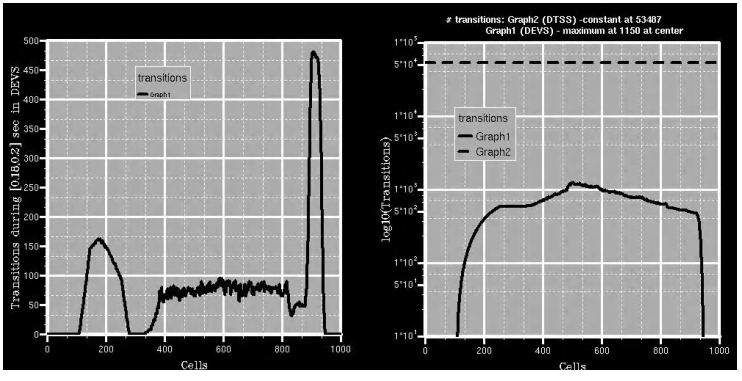


Fig. 6. The number of state changes computed by the DEVS simulator during the time interval [0.18,0.2] (left) and [0.0,0.2] (right).

Table 1 shows the parameters used for each choice of discretization in space and the resulting time to solution. It can be seen that the speed advantage of the DEVS solution, relative to the discrete time solution, increases as the number of cells grows from 200 to 1600. At 1600 cells, the speedup appears to level off at a value of about 34. The combination of focused computational effort and significantly reduced number of computations per cell has produced more than an order of magnitude improvement in the time to solution.

Table 1. Solution parameters and speedup for DEVS and discrete time (DTSS) techniques.

Cells	Quantum size (U1,U2,U3)	Time step	DEVS soln. time (seconds)	DTSS soln. time (seconds)	Relative speedup of DEVS
200	(0.1,1.0,1000.0)	1.87E-005	2.23	19.01	8.52
1000	"	3.74E-006	13.37	380.76	28.5
1600	"	2.34E-006	26.95	941.73	34.9
2000	"	1.87E-006	41.19	1429.42	34.7
4000	"	9.35E-007	165.73	5482	32.8
6000	"	6.23E-007	339.55	10534.5	31.0
8000	'	4.67E-007	622.94	21585	34.7

4 Conclusions

We are developing a theory that suggests that the potential speed advantages with DEVS are to be expected for pdes that are characterized by heterogeneity in their time and space behavior. In such cases, as exemplified by the example discussed above, discrete events are a natural way to focus attention on the portions of the solution that are exhibiting high activity levels at the moment. In fact, theory suggests a way to characterize the activity of solutions over time and space independently of the solution technique that might be employed. This activity measure, when divided by a quantum size, predicts the number of boundary crossings (computations) required by the DEVS simulator for the accuracy afforded by that quantum size. Where significant heterogeneity of activity exists, the number of discrete event computations may be orders of magnitude lower than that required by a uniform allocation of computational resources across both space and time. To realize these potential gains, the choice of data structures used to implement the discrete event simulation engine is of critical importance. A key feature of the discrete event simulator is the use of data structures whose relevant operations have a time complexity of at most $O(\log_2 N)$, where N is the total number of models currently in the simuation (e.g., cells and Riemann boundary flux solvers in the above example).

The work reported here, and research in progress, suggests that DEVS can offer significant performance advantages for simulation of continuous phenomena

characterized by spatiotemporal heterogeneity. Since the DEVS hierarchical, modular framework accommodates coupled models containing both discrete and continuous components, it offers a scalable, efficient framework for very large scale distributed simulation. An important avenue to explore is the incorporation of spatial disaggregation techniques (such as adaptive mesh refinement) within the variable structure capabilities of DEVS modeling and simulation.

Acknowledgement.
This research has been supported in part by NSF Grant No. DMI-0122227, "Discrete Event System Specification (DEVS) as a Formal Modeling and Simulation Framework for Scaleable Enterprise Design" and in part by the Scientific Discovery through Advanced Computing (SciDAC) program of the DOE, grant number DE-FC02-01ER41184.

References

1. Adevs software, http://www.ece.arizona.edu/~nutaro
2. J. Ameghino, A. Tróccoli, G. Wainer. "Models of Complex Physical Systems using Cell-DEVS", Proceedings of the Annual Simulation Symposium, Seattle, Washington, 2001.
3. DEVSJAVA software, http://www.acims.arizona.edu
4. Kofman, E.. "Quantization Based Simulation of Differential Algebraic Equation Systems", Technical Report LSD0203, LSD, Universidad Nacional de Rosario, 2002.
5. Alexandre Muzy, Eric Innocenti, Antoine Aiello, Jean-Francois Santucci, and Gabriel Wainer. "Cell-DEVS Quantization Techniques in a Fire Spreading Application", Winter Simulation Conference, San Diego, California, 2002.
6. Zeigler, B.P., (2002). "The brain-machine disanalogy revisited", *BioSystems,* Vol. 64, pp. 127-140.
7. Zeigler, B.P., T.G. Kim, et al. Theory of Modeling and Simulation. New York, NY, Academic Press, 2000.
8. Zeigler, B.P., H.S. Sarjoughian. "Introduction to DEVS Modeling and Simulation with JAVA: A Simplified Approach to HLA-Compliant Distributed Simulations", http://www.acims.arizona.edu, 2001.

Appendix: A Brief Review of DEVS Concepts

The structure of a model may be expressed in a mathematical language called a *formalism.* The formalism defines how to generate new values for variables and the times the new values should take effect. The discrete event formalism focuses on the changes of variable values and generates time segments that are piecewise constant. Thus an event is a change in a variable value that occurs instantaneously. An important aspect of the DEVS formalism is that the time intervals between event occurrences are variable (in contrast to discrete time where the time step is generally a constant number).

To specify modular discrete event models requires that we adopt a different view than that fostered by traditional simulation languages. As with modular specification

in general, we must view a model as possessing input and output ports through which all interaction with the environment is mediated. In the discrete event case, events determine the values appearing on such ports. More specifically, when external events, arising outside the model, are received on its input ports, the model description must determine how it responds to them. Also, internal events, arising within the model, change its state, as well as manifesting themselves as events on the output ports, which in turn are to be transmitted to other model components.

A *basic model* contains the following information:

- the set of input ports through which external events are received,
- the set of output ports through which external events are sent,
- the set of state variables and parameters: two state variables are usually present, "phase" and "sigma" (in the absence of external events the system stays in the current "phase" for the time given by "sigma"),
- the time advance function which controls the timing of internal transitions – when the "sigma" state variable is present, this function just returns the value of "sigma",
- the internal transition function which specifies to which next state the system will transit after the time given by the time advance function has elapsed,
- the external transition function which specifies how the system changes state when an input is received – the effect is to place the system in a new "phase" and "sigma" thus scheduling it for a next internal transition; the next state is computed on the basis of the present state, the input port and value of the external event, and the time that has elapsed in the current state,
- the confluent transition function which is applied when an input is received at the same time that an internal transition is to occur, and
- the output function which generates an external output just before an internal transition takes place.

A *Discrete Event System Specification* (*DEVS*) is a structure

$$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$

where

X is the set of input values.
S is a set of states.
Y is the set of output values.
δ_{int}: $S \to S$ is the *internal transition* function.
δ_{ext}: $Q \times X^b \to S$ is the *external transition* function, where
$Q \in \{(s,e) \mid s \in S, 0 \le e \le ta(s)\}$ is the *total state* set,
e is the *time elapsed* since last transition,
X^b denotes the collection of bags over X (sets in which some elements may occur more than once).
δ_{con}: $S \times X^b \to S$ is the *confluent transition* function.
λ: $S \to Y^b$ is the output function.
ta: $S \to \mathbf{R}^+_{0,\infty}$ is the *time advance* function.

The interpretation of these elements is illustrated in figure 7. At any time the system is in some state, *s*. If no external event occurs the system will stay in state *s* for time *ta*(*s*). Notice that *ta*(*s*) could be a real number and it can also take on the values 0 and ∞. In the first case, the stay in state *s* is so short that no external events can intervene – we say that *s* is a *transitory* state. In the second case, the system will stay in *s* forever unless an external event interrupts its slumber. We say that *s* is *a passive* state in this case. When the resting time expires, i.e., when the elapsed time, $e = ta(s)$, the system outputs the value, $\lambda(s)$, and changes to state $\delta_{int}(s)$. Note that output is only possible just before internal transitions.

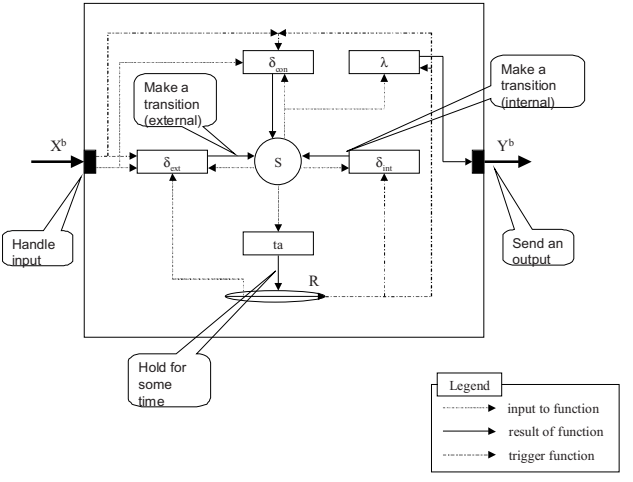


Fig. 7. Interpretation of the DEVS structure.

If an external event $x \in X^b$ occurs before this expiration time, i.e., when the system is in total state (*s*, *e*) with $e \leq ta(s)$, the system changes to state $\delta_{ext}(s,e,x)$. Thus the internal transition function dictates the system's new state when no events have occurred since the last transition. While the external transition function dictates the system's new state when an external event occurs – this state is determined by the input, *x*, the current state, *s*, and how long the system has been in this state, *e*, when the external event occurred. In both cases, the system is then is some new state s' with some new resting time, $ta(s')$ and the same story continues.

Note that an external event $x \in X^b$ is a bag of elements of X. This means that one or more elements can appear on input ports at the same time. This capability is needed since DEVS allows many components to generate output and send these to input ports all at the same instant of time.

Basic models may be coupled to form a *coupled model*. A coupled model tells how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to hierarchical construction.

Data Driven Design Optimization Methodology A Dynamic Data Driven Application System

Doyle Knight

Dept of Mechanical and Aerospace Engineering
Rutgers - The State University of New Jersey
New Brunswick, NJ 08903
knight@soemail.rutgers.edu

Abstract. Engineering design optimization using concurrent integrated experiment and simulation is a Dynamic Data Driven Application System (DDDAS) wherein remote experiment and simulation can be synergistically utilized in real-time to achieve better designs in less time than conventional methods. The paper describes the Data Driven Design Optimization Methodology (DDDOM) being developed for engineering design optimization.

1 Introduction

A Dynamic Data Driven Application System (DDDAS) is an application software system capable of accepting and effectively utilizing remote data in real time (*i.e.*, during the execution of the application software). Many software systems currently utilize *static* input data, *i.e.*, input data which is specified *a priori*. The key concept of DDDAS is the generalization of application software systems to *dynamically utilize* real-time data arising from remote experiment and simulation, and to *control* such remote experiment and simulation to improve the performance of the application software system. The concept is illustrated schematically in Fig. 1.

Dynamic Data Driven Application Systems have become a major subject of research interest due to the continuing rapid advances in technology. These include, for example, improvements in computer processor performance (*i.e.*, doubling every 18 to 24 months (Berkowitz 1996)), network bandwidth, Rapid Prototyping (RP) and real-time data acquisition and control (*e.g.*, using Micro-Electro Mechanical Systems (MEMS)). These rapid technological advances provide the means to revolutionize the use of existing and future application software systems. For example, the NSF TeraGrid[1] is a multi-year effort to deploy 20 TFlops of computing capacity at five sites connected through a 40 Gbps network. The TeraGrid will offer unprecedented remot computing capability which can be utilized in real-time by application software systems executing at users' sites. The incorporation of the NSF Teragrid into an application software system is therefore an example of a Dynamic Data Driven Application System.

[1] http://www.ncsa.uiuc.edu/About/TeraGrid/

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 329–336, 2003.
© Springer-Verlag Berlin Heidelberg 2003

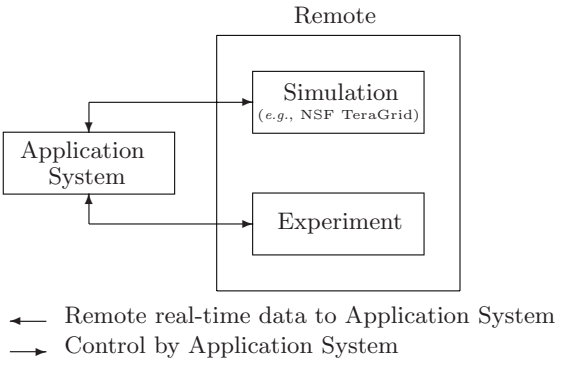


Fig. 1. Use of remote experiment and simulation in application software

The objective of this paper is to present a Dynamic Data Driven Application System for engineering design optimization. In the following sections, we describe the general engineering design optimization problem, the conventional approach to engineering design optimization, our Dynamic Data Driven Application System for engineering design optimization, and a demonstration example.

2 Design Optimization

The general engineering design optimization problem is defined as

$$\text{minimize } f_j(\boldsymbol{x}) \quad \text{for } j = 1, m \tag{1}$$

subject to

$$d_k \leq 0 \quad \text{for } k = 1, l \tag{2}$$

where

$$\boldsymbol{f} = (f_1, \ldots, f_m)^{\mathrm{T}} \tag{3}$$

is the vector of objective functions and

$$\boldsymbol{d} = (d_1, \ldots, d_l)^{\mathrm{T}} \tag{4}$$

is the vector of constraints and

$$\boldsymbol{x} = (x_1, \ldots, x_n)^{\mathrm{T}} \tag{5}$$

is the vector of design variables. The solution to the general engineering design problem is the set of non-dominated designs known as the Pareto Set (Deb 2001). Within this set of designs, improvement in one design criterion f_j can only be achieved at the expense of the degradation in the performance of one or more other design criteria.

3 Conventional Engineering Design Optimization

Engineering design is a potential Dynamic Data Driven Application System wherein remote real-time input from high performance computing resources (*e.g.*, NSF Teragrid) and experiment (*e.g.*, using capability for rapid fabrication of new experimental configurations) can produce a better design in shorter time with lower cost than the conventional approach. Fig. 2 illustrates the conventional approach to engineering design. An initial design is typically evaluated using simulation software executing on local computer resources. An example is Computational Fluid Dynamics (CFD) simulation using one of the several available commercial CFD software packages (*e.g.*, FLUENT, GASP, STAR-CD). The results of the simulation are reviewed by the designer and additional simulations performed as needed. An experiment is then planned and executed, and the results compared with the simulation. The process is iterated until a satisfactory design has been achieved. A major limitation of the conventional approach is the absence of synergistic real-time use of remot experimental data and high performance computer resources (*e.g.*, the NSF TeraGrid).

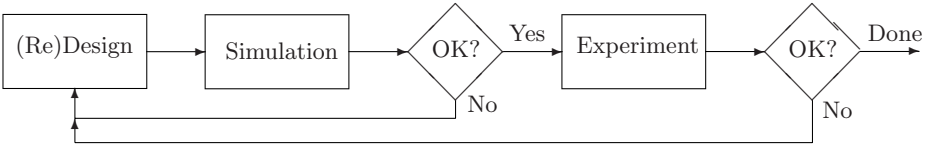


Fig. 2. Conventional approach to engineering design

4 Data Driven Design Optimization Methodology

We are developing an integrated software system, denoted Data Driven Design Optimization Methodology (DDDOM), to synergistically incorporate remote experiment and simulation into an automated design optimization methodology. The DDDOM is comprised of six elements as illustrated in Fig. 3 and are summarized below.

Controller The DDDOM Controller provides direction and control of the design optimization process. It is written in Perl (Wall *et al* 2000), a powerful systems programming language which enables robust program control on local and remote systems. The Controller utilizes the Globus software system (Foster and Kesselman 1997) to link to the NSF TeraGrid.

User Interface The User Interface provides for the initiation and monitoring of the design optimization. It also provides visualization of the intermediate results of the design optimization process. The interface is written in Perl/Tk (Lidie and Walsh 2002). Both Perl and Perl/Tk operate on virtually every platform and operating system.

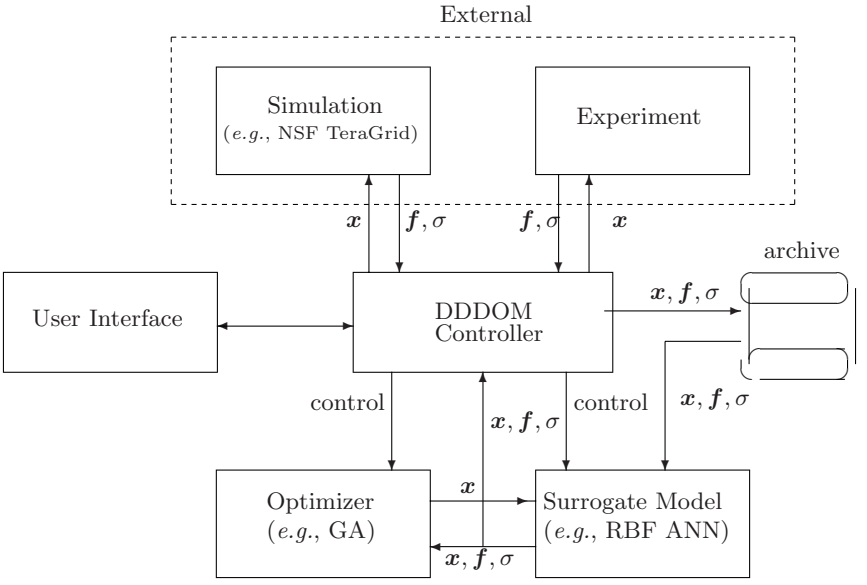


Fig. 3. Data Driven Design Optimization Methodology

Optimizer A Multi-Criteria Design Optimization (MDO) algorithm is used to search the design space to define the Pareto Set. Both local (*i.e.*, gradient-based) and global (*i.e.*, stochastic-based) algorithms are used. Examples are CFSQP (Lawrence *et al* 1994) and GADO (Rasheed and Hirsh 1999), respectively.
Surrogate Model Surrogate Models of the objective functions $\boldsymbol{f}$ are developed during the design optimization using experiment and simulation. These models are utilized by the optimizer to search the design space for the Pareto Set. We utilize Response Surfaces (Myers and Montgomery 1995) and Radial Basis Function Artificial Neural Networks (Hertz *et al* 1991). The Surrogate Model incorporates the measure of uncertainty σ in the results generated by the experiment and simulation.
Experiment Experiments are performed in real-time with the conditions of the experiment defined by the Controller. The interface with experiments is typically National Instruments LabView (Travis 2002). Rapid Prototyping is used to build new models as needed with minimal delay (*e.g.*, one day).
Simulation Simulations can be performed locally (*i.e.*, at the same site as the DDDOM is operated) or remotely (*e.g.*, on the NSF TeraGrid). The simulations are performed in real-time with the conditions of the simulation defined by the Controller.

5 Example

A five minute demonstration has been developed to illustrate the principle of operation of the DDDOM. The limited duration of the demonstration precludes

the actual use of RP which typically requires several hours for fabrication of the prototype for experiment. Therefore, we have selected a single experimental configuration. Nonetheless, the experiment provides data to the DDDOM Controller in real time, and the Controller directs the experiment, thereby illustrating the DDDOM concept.

The specific problem is defined as follows. Consider the flow of air in a converging-diverging nozzle attached to a stagnation chamber as illustrated in Fig. 4. The nozzle is designed to achieve Mach 2 at the exit assuming isentropic expansion throughout the nozzle. Depending upon the ratio of stagnation pressure p_{t_∞} to ambient pressure p_a, the flow in the diverging section of the nozzle can be 1) subsonic everywhere, 2) supersonic and subsonic with a normal shock at some location in the diverging section, and 3) supersonic everywhere (Anderson 2003). In the latter case, the theoretical exit Mach number is two. A one-dimensional simulation based upon inviscid gas dynamics (Liepmann and Roshko 1957) indicates that the exit pressure ratio p_e *vs* stagnation pressure p_{t_∞} behaves as shown by the curve denoted *Simulation* in Fig. 5. This behavior is idealized, however, due to the neglect of the viscous boundary layer on the nozzle walls and the effect of the interaction of the normal shock wave (if it exists) with the boundary layers. Consequently, the actual exit pressure p_e *vs* stagnation pressure p_{t_∞} curve differs substantially from the idealized shape.

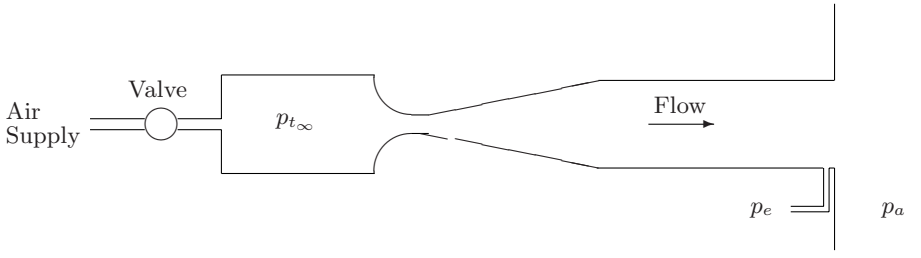


Fig. 4. Experimental configuration

The objective of the optimization is to determine the stagnation pressure p_{t_∞} which yields the minimum exit pressure p_e for a fixed ambient pressure p_a. Although this is a single design objective problem, it illustrates the fundamental concepts of DDDOM. Due to the physical assumptions inherent in the simulation code, the p_e *vs* p_{t_∞} function obtained from the simulation can be significantly in error. This is an intentional choice to force the DDDOM Controller to build and refine the Surrogate Model utilizing both simulation and experimental data to determine the optimum value of p_{t_∞} (*i.e.*, the value of p_{t_∞} which yields the minimum value of p_e). The dynamic updating of the Surrogate Model using real time experimental data is an example of a Dynamic Data Driven Application System.

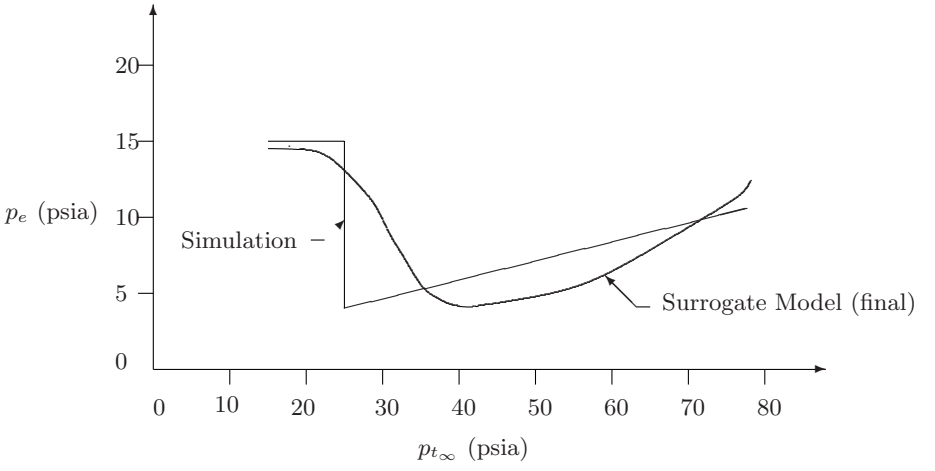


Fig. 5. Exit pressure p_e *vs* stagnation pressure p_{t_∞}

The experiment is located in a laboratory in the Department of Mechanical and Aerospace Engineering at Rutgers University and controlled by a workstation running Windows 98. The DDDOM is executed on a separate workstation in another building running Linux. The DDDOM and experiment workstation communicate via high speed Ethernet. A National Instruments LabView Virtual Instrument (VI) program (Travis 2002) executes on a workstation adjacent to the experiment. The VI continually monitors an input file on the workstation which contains a specified value of stagnation pressure p_{t_∞} and adjusts the stagnation pressure by means of a servo-controlled valve to the specified value. The VI thereafter writes the measured values of the stagnation pressure p'_{t_∞} (which differs from the specified p_{t_∞} due to "play" in the valve) and exit pressure p'_e on a file on the workstation.

The DDDOM Controller (Fig. 3) manages the optimization problem by the following sequence:

1. The Surrogate Model is initialized using the one-dimensional inviscid simulation code.
2. The optimizer (a Genetic Algorithm) determines the minimum of the p_{t_∞} *vs* p_e curve using the current Surrogate Model.
3. A value of p_{t_∞} is chosen for the experiment. In the early stages of the optimization, the value determined in step no. 2 is chosen with a probability of 0.5; otherwise, a random value is chosen within the overall range of experimentally possible values of p_{t_∞} (*i.e.*, 15 psi to 60 psi). This procedure avoids premature convergence to an exit pressure which is not the actual minimum.
4. A file is transferred to the workstation containing an identification number and the prescribed value of p_{t_∞}.
5. The VI on the workstation reads the file, changes the experimental total pressure to the specified value within experimental tolerances, and writes

the identification number, the actual experimental total pressure p'_{t_∞} and exit pressure p'_e to the file.

6. The DDDOM Controller accesses the file and reads the experimental values of p'_{t_∞} and p'_e. The database for the Surrogate Model is updated using this data and the nearest value to p_{t_∞} in the database which was generated by the one-dimensional simulation code is removed.
7. The convergence of the Surrogate Model is checked. If it is not converged, the procedure is repeated beginning at step no. 2.

6 Conclusions

Engineering design is an example of a Dynamic Data Driven Application System wherein real-time data from experiment and simulation can be effectively utilized to yield better designs in shorter time. A Data Driven Design Optimization Methodology (DDDOM) is being developed which incorporates experiment and simulation in a real-time, synergistic manner. The theory and organization of DDDOM is described. An example demonstration is described.

Acknowledgments.
The research is sponsored by the US National Science Foundation under grant CTS-0121058. The program managers are Drs. Frederica Darema, C. F. Chen and Michael Plesniak. The assistance of Profs. Greg Elliott and Madara Ogot in development and implementation of the supersonic nozzle experiment is gratefully acknowledged.

References

Anderson, J. Modern Compressible Flow with Historical Perspective. McGraw Hill (2003), New York.

Berkowitz, B. Information Age Intelligence. Foreign Policy **103** (1996) 35–50.

Deb, K. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons (2001), New York.

Foster, I. and Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications **11 (2)** (1997) 115–128.

Hertz, J., Krogh, A., and Palmer, R. Introduction to the Theory of Neural Computation. Lecture Notes Vol I, Santa Fe Institute Studies in the Sciences of Complexity. Perseus Publishing (1991), Cambridge, MA.

Knight, D., Elliott, G., Jaluria, Y., Langrana, N., and Rasheed, K. Automated Optimal Design Using Concurrent Integrated Experiment and Simulation. AIAA Paper No. 2002-5636, AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA (2002).

Lawrence, C., Zhou, J., and Tits, J. User's Guide for CFSQP Version 2.1. University of Maryland, Electrical Engineering Dept & Institute for Systems Research (1994).

Lidie, S., and Walsh, N. Mastering Perl/Tk. O'Reilly and Associates (2002), Sebastopol, CA.

Liepmann, H., and Roshko, A. Elements of Gas Dynamics. John Wiley & Sons (1957), New York.

Myers, R., and Montgomery, D. Response Surface Methodology - Process and Product Optimization Using Design Experiments. John Wiley & Sons (1995), New York.

Rasheed, K., and Hirsh, H. Learning to be Selective in Genetic-Algorithm-Based Design Optimization. Artificial Intelligence in Engineering, Design, Analysis and Manufacturing **13** (1999) 157–169.

Travis, J. LabVIEW for Everyone. Prentice Hall (2002), Upper Saddle River, NJ.

Wall, L., Christiansen, T., and Orwant, J. Programming Perl. O'Reilly and Associates (2000), Sebastopol, CA.

Derivation of Natural Stimulus Feature Set Using a Data-Driven Model

Alexander G. Dimitrov, Tomas Gedeon, Brendan Mumey, Ross Snider, Zane Aldworth, Albert E. Parker, and John P. Miller

Center for Computational Biology, Montana State University, Bozeman MT 59717
{Dimitrov, Gedeon, Mumey, Snider, Aldworth, Parker, Miller}@cns.montana.edu

Abstract. A formal approach for deciphering the information contained within nerve cell ensemble activity patterns is presented. Approximations of each nerve cell's coding scheme is derived by quantizing its neural responses into a small reproduction set, and minimizing an information-based distortion function. During an experiment, the sensory stimulus world presented to the animal is modified to contain a richer set of relevant features, as those features are discovered. A dictionary of equivalence classes is derived, in which classes of stimulus features correspond to classes of spike-pattern code words. We have tested the approach on a simple insect sensory system.

1 The Neural Encoding Problem

Two major goals facing neuroscientists are to understand how information is encoded in the activity patterns of neural ensembles and to understand how those activity patterns are decoded by cells at the subsequent processing stages. A formal, general approach toward achieving those goals has been presented in previous work [5]. An extension of the technique is presented here, along with a demonstration of the application of that approach to the analysis of neural coding in an insect sensory system. This approach is being refined through the development of a data-driven model of that sensory system.

1.1 A Dictionary for the Neural Code

Tools from information theory were used recently to characterize the neural coding scheme of a simple sensory system [3]. That work demonstrated that a coding scheme can be conceptualised as an almost-deterministic relation between clusters of stimulus-response classes, where each class consists of a set of stimuli and a synonymous set of the neural responses elicited by those stimuli. Each “entry” in the dictionary consists of one of these stimulus-responses classes: i.e., all of the stimuli in the stimulus class are treated as being equivalent (e.g. repeated but slightly variant strums of a guitar chord) and all neural responses in the class are also considered to

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 337–345, 2003.
© Springer-Verlag Berlin Heidelberg 2003

be equivalent (e.g. repeated but slightly variant hand-written notations for that chord). In the context of the sensory system used to illustrate that approach, the stimuli are short-duration (circa 20 msec.) segments of sensory input waveforms, and the neural responses are short-duration patterns of action potentials (1-3 APs within 20 ms windows following the stimuli).

A method was developed to find high quality approximations of such a coding scheme [4]. The technique involved the quantization of the neural responses to a small reproduction set, and used a minimization of an information-based distortion function to optimize the quantization. In cases involving complex, high-dimensional input stimuli, a model was derived for the stimulus-response relation. Several classes of models were used to provide upper bounds to the information distortion function used in the optimization problem [5]. In general, a smaller value of the cost function indicated a better model. All models were variants of Gaussian Mixture Models (GMM) [1]. The differences pertained to the number of parameters used: richer models provided a better bound to the cost function, but needed more data for robust estimates. The applicability of the models were demonstrated by investigating coding properties of several identified neurons in the cricket cercal sensory system [4,5].

1.2 Identification of Stimulus Features Encoded in Neural Activity Patterns

A typical initiation point for system-identification studies of this nature involve the presentation of Gaussian white noise stimuli. Many earlier studies have used GWN stimuli for the characterization of neural coding characteristics. However, recent results in the cricket cercal sensory system indicate that sensory interneurons show sensitivity to higher-order statistical features that occur very infrequently in GWN stimuli [13]. Any characterization of the encoding scheme of a neural system that does not encompass the relevant stimulus regime would be essentially meaningless.

Here, an approach is presented that enables discovery of the set of stimuli to which a cell or ensemble of cells is “tuned”, based on the assumption that those cells have become optimised over time to encode and transmit information about that natural stimulus set. This approach also enables a more consistent characterization of the stimulus/response properties of neurons to their natural, behaviourally-relevant stimulus regime. The techniques are introduced and demonstrated within the context of a simple test system: the cricket cercal sensory system.

2 Neurophysiological Testbed Preparation

The preparation used for these studies was the cercal sensory system of the cricket. This system mediates the detection and analysis of low velocity air currents in the cricket’s immediate environment. This sensory system is capable of detecting the direction and dynamic properties of air currents with great accuracy and precision [6-8,11,15-19] and can be thought of as a near-field, low-frequency extension of the animal’s auditory system.

The receptor organs for this modality are two antenna-like appendages called cerci at the rear of the abdomen. Each cercus is covered with approximately 1000 filiform mechanosensory hairs, like bristles on a bottle brush. Each hair is constrained to move along a single axis in the horizontal plane. As a result of this mechanical constraint, an air current of sufficient strength will deflect each hair from its rest position by an amount that is proportional to the cosine of the angle between the air current direction and the hairs movement axis. The 1000 hairs on each cercus are arrayed with their movement axes in diverse orientations within the horizontal plane, insuring that the relative movements of the ensemble of hairs will depend on the direction of the air current. The filiform hairs also display differential sensitivity to aspects of the dynamics of air displacements, including the frequency, velocity, and acceleration of air currents [12,14].

Each hair is innervated by a single spike-generating mechanosensory receptor neuron. These receptors display directional and dynamical sensitivities that are derived directly from the mechanical properties of the hairs[8-10,14-16]. The set of approximately 2000 receptors innervating these filiform hairs have frequency sensitivities spanning the range from about 5 Hz up to about 1000 Hz.

The sensory afferents synapse with a group of approximately thirty local interneurons and approximately twenty identified projecting interneurons that send their axons to motor centers in the thorax and integrative centers in the brain. It is a subset of these projecting interneurons that we study here. Like the afferents, these interneurons are also sensitive to the direction and dynamics of air current stimuli [11, 18,19]. Stimulus-evoked neural responses have been measured in several projecting and local interneurons, using several different classes of air current stimuli. Each of the interneurons studied so far has a unique set of directional and dynamic response characteristics. Previous studies have shown that these projecting interneurons encode a significant quantity of information about the direction and velocity of low frequency air current stimuli with a linear rate code [2,18,19]. More recent studies demonstrate that there is also substantial amount of information in the spike trains that cannot be accounted for by a simple linear encoding scheme [13]. Evidence suggests the implementation of an ensemble temporal encoding scheme in this system.

3 Methods

3.1 Experimental Approach

Stimulus-response properties of sensory interneurons were measured using intracellular and extracellular electrodes. Stimuli consisted of controlled air currents directed across the animals' bodies, and the responses consisted of the corresponding spike trains elicited by those air currents. The cricket preparations were mounted within a miniature wind tunnel, which generated laminar air currents having precisely controlled direction and velocity parameters. Details of the dissection, stimulus

generation, and electrophysiological recording procedures are presented in Dimitrov et al. (2001).

3.2 Derivation of the Stimulus-Response Equivalence Sets

Details of all analytical techniques, as well as discussions of our computational approaches, are presented in Dimitrov et al. [5]. A brief summary is as follows.

A model of neural processing. The *input signal X* to a neuron (or neural ensemble) may be a sensory stimulus or may be the activity of another set of (pre-synaptic) neurons. We considered the input signal to be produced by a source with a probability $p(x)$. The *output signal Y* generated by that neuron (or neural ensemble) in response to *Y* will be a spike train (or ensemble of spike trains.) We consider the encoding of *X* into *Y* to be a map from one stochastic signal to the other. This stochastic map is the *encoder* $q(y|x)$, which will model the operations of this neuronal layer. The *output signal Y* is induced by $q(y|x)$ by $p(y) = \sum_x q(y|x)p(x)$.

The neural code can thus be viewed as being probabilistic on a fine scale but deterministic on a large scale. This view emerges naturally in the context of Information Theory. The Noisy Channel Coding Theorem suggests that, in this context, relations between individual elements of the stimulus and response spaces are not the basic building elements of the system. Rather, the defining objects are relations between *classes* of stimulus-response pairs. There are about $2^{I(X;Y)}$ such equivalence classes (i.e., codeword classes). When restricted to codeword classes, the stimulus-response relation is almost bijective. That is, with probability close to 1, elements of *Y* are assigned to elements of *X* in the same codeword class. We decode an output *y* as any of the inputs that belong to the same codeword class. Similarly, we consider the neural representation of an input *x* to be any of the outputs in the same codeword class. Stimuli from the same equivalence class are considered indistinguishable from each other, as are responses from within the same class.

Finding the codebook. Given this model of neural function, our task is to recover the codebook. In this context, this equates to identifying the joint stimulus-response classes that define the coding relation. The approach we use is to quantize (*i.e.*, cluster) the response space *Y* to a small reproduction space of finitely many abstract classes, Y_N. This method allows us to study coarse (i.e., small *N*) but highly informative models of a coding scheme, and then to automatically refine them when more data becomes available. This refinement is done by simply increasing the size of the reproduction, *N*.

The mutual information $I(X;Y)$ tells us how many different states on the average can be distinguished in *X* by observing *Y*. If we quantize *Y* to Y_N (a reproduction with N elements), we can estimate $I(X;Y_N)$, which is the mutual information between *X* and the reproduction Y_N. Our information-preserving criterion will then require that we choose a quantizer that preserves as much of the mutual information as possible, *i.e.,* to choose the quantizer $q(Y_N|Y)$ which minimizes the difference

$$D_I(Y;Y_N) = I(X;Y) - I(X;Y_N) \quad \textbf{(1)}$$

Following examples from rate distortion theory, this problem of optimal quantization can then be formulated as a maximum entropy problem. The reason is that, among all quantizers that satisfy a given set of constraints, the maximum entropy quantizer does not implicitly introduce additional constraints in the problem. Within this framework, the minimum distortion problem is posed as a maximum quantization entropy problem with an appropriate information-theoretic distortion constraint. The optimal quantizer $q(y_N|y)$ induces a coding scheme from which is the most informative approximation of the original relation $p(x|y)$ for a fixed size N of the reproduction Y_N. Increasing N produces a refinement of the approximation, which is more informative (has lower distortion and thus preserves more of the original mutual information I(X;Y)). The elements of Y_N can be interpreted as the labels of the equivalence classes which we want to find. The quantizer $q(y_N|y)$ gives the probability of a response y belonging to an equivalence class y_N. Through this approach, we recover an almost complete reproduction of the coding scheme as a relation between stimulus-response equivalence classes. For each neuron, the characteristic stimulus features are represented as the mean voltage waveforms of the stimulus that drove the air currents immediately preceding the elicited spike pattern code words, and the response code words are represented as the spike patterns that corresponded to those stimulus features.

3.3 Refinement of the Stimulus Set

In previous studies, the encoding properties of nerve cells were studied using single-axis sinusoidal and band-passed gaussian noise stimuli. Here we characterized the INs responses to more complex stimuli, constructed to be more representative of the system's natural stimulus set. During each experiment, the sensory stimulus ensemble presented to the animal was modified to contain a richer set of relevant features, as those features were discovered through on-line analysis of the accumulating stimulus/response data.

During the course of the physiological recording, the system was stimulated initially with air currents drawn from a band-limited (5-500Hz) Gaussian white noise (GWN) source [19]. This broad, non-specific stimulus allowed us to explore a large portion of the input space, and provided sufficient stimulation for a coarse model (i.e., a low-dimensional quantization) of the system. This coarse analysis yielded the preliminary set of response-conditioned mean stimuli leading up to the most frequently observed neural spike-pattern code words (e.g. single spikes, short-interval spike doublets and triplets.)

After the initial model was in place, we modified the stimulus set in two ways. First, we added samples of the response-conditioned stimulus classes derived from the coarse analysis into the GWN stimulus waveforms, to increase the frequency of occurrence of the neural responses for that stimulus class. This allowed us to sample this part of stimulus space more finely, and refine the stimulus-response model, thus lowering the distortion and increasing the informativeness of the reproduction.

Second, we modified the variability of the stimuli along certain stimulus dimensions. Since most of the GMMs project parts of the stimulus to a smaller space,

reducing the variability in the discarded subspace does not significantly affect the model. Modifying the variability in the retained subspace does, however, modify the properties of the model. In this case, we modified it in a way that provided a tighter bound in the subsequent analysis, which indicated that we had built a better model. We also modified the variability in directions orthogonal to the ones already presented. The intent of this step was analogous to the initial GWN stimulation: to present parts of stimulus space which the sensory system had not yet perceived.

This refinement procedure was repeated, and the stimulus ensemble was thus iteratively refined from non-specific stimuli to the specific subset of stimuli which elicited a selection of the quantized response classes, which in turn enabled further refinement of the quantizer itself. The method produced a more refined dictionary of equivalence classes. In principle, this approach can be continued until the stimulus space is explored in sufficient detail that further refinements of the stimulus do not produce refinements of the response model.

4 Results

This analytical approach was used to characterize the encoding characteristics of single identified sensory interneurons in the cricket cercal system. The specific goal of the experiments and analyses were to discover (jointly) a) the dynamic stimulus waveform features encoded by the cells, and b) the spike train codeword patterns that encoded those features. Figure 1 demonstrates the application of this approach to one identified neuron in the cercal sensory system. In this figure, the stimulus features are represented as the mean voltage waveforms of the stimuli that drove the air currents immediately preceding the elicited spike pattern code words, and the response code words are represented as the actual spike patterns that corresponded to those stimulus features. For this cell, the stimulus-response space was quantized into four classes (i.e., N=4). The top panel shows four class-conditioned mean stimulus waveforms, corresponding to the four spike pattern code words derived through this successive quantization and stimulus refinement procedure. These waveforms correspond to the air-current velocity presented to the preparation. The four mean waveforms are each plotted in the gray scale of the corresponding class. The horizontal axis of this top plot denotes time, in ms, relative to the occurrence of the first spike in a class. That is, time 0 is the time at which the first spike on the codeword pattern occurred. The dashed lines denote 95% confidence intervals of the means, which depend on the reproduction size, N.

The lower right panel plots a representation of spike patterns associated with these stimulus waveforms. The spike patterns are plotted in a manner that is similar to a peristimulus time histogram. We call this representation a class conditioned time histogram (CCTH.) These are the classes of spike patterns that served as the basis for extracting the corresponding mean stimulus waveforms. Every class starts with a spike (line at 0ms). We plot the conditional probability of spike occurrence *vs.* time for each pattern on a logarithmic scale, with black indicating a probability of one for the occurrence of a spike, and a lighter shade of gray representing a lower probability. To be precise, we plot the expectation is $_y\, y_i p(y_i|y_N)$. The pattern y_i can be considered

as the conditional probability $p(t_j|y_i)$ = p(spike occurs at time t_j given that the observed pattern is y_i). This probability is 1 at times when a spike occurs and zero otherwise. In this case, this panel can be interpreted as showing $p(t_j|y_N)$ = $_y p(t_j|y_i)p(y_i|y_N)$ - the conditional probability of a spike at time t_i given class y_N. The similarity to a PSTH is that we present the distribution of spikes in time, conditioned on the occurrence of an event. For the PSTH, the event is a particular stimulus. For this representation, the event is a certain response class, hence the name CCTH. These patterns are aligned in time with the mean stimulus waveforms that elicited them, in the panel directly above. That is, the significant portions of the stimulus patterns that correspond to the observed spike patterns occur during the 20 ms preceding the spike code words.

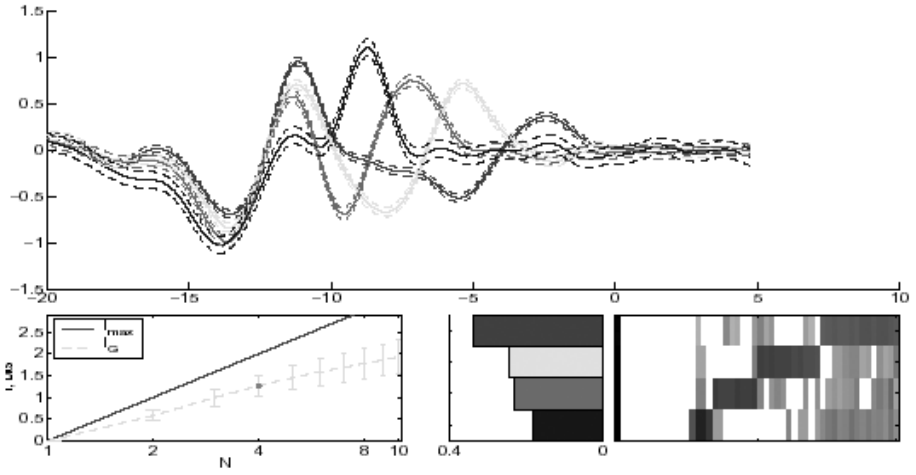


Fig. 1. A quantization with 4 classes. Top panel: the four class-conditioned mean stimulus waveforms. Bottom right panel: the CCTH of a spike at time T given the pattern in a certain class. Lower center panel: relative proportion of spike patterns belonging to the different classes, as GMM priors. Lower left panel: estimate of the lower bound to the mutual information *(gray dashed curve)* and the absolute upper bound for the same level of quantization *(dark solid curve, log_2)*

The lower center panel to the left of these CCTH plots show the relative proportion of spike patterns belonging to the different classes, as GMM priors (weights). These bars are gray-scale-coded to indicate the class-conditioned mean stimulus waveform to which the spike pattern to the right corresponds. This particular quantization grouped the spike patterns roughly according to interspike intervals: the top class consisted mostly of doublets with a second spike 7-10 ms after the initial spike (dark gray range to the right), and a few triplets (light gray bars in front), for which the third spike is in the same range. The bottom (black) class consists mostly of short doublets, with a second spike 2.5-3.3 ms after the first spike, and a range of triplets with a third spike 6-10ms after the first spike.

The lower left panel shows the estimate of the lower bound to the mutual information (gray dashed curve) and the absolute upper bound for the same level of

quantization (dark solid curve, $\log_2 N$). The error bars mark the uncertainty of the estimate, which depend on the reproduction size. The estimate for the 4-class quantization shown here is denoted with a dark plotted point on the mutual information curve at N=4.

For this cell, application of the iterative stimulus-refinement approach yielded a set of stimulus waveforms that differed significantly from the set obtained when the stimulus regime was limited to Gaussian white noise. Specifically, several of the characteristic stimulus features included multiple cycles of sinusoidal-like oscillations.

5 Conclusions

This analytical approach offered several significant advantages to our characterization of the neural encoding scheme for this cell than previous approaches. First, the approach enabled a more rapid convergence toward a more accurate and meaningful representation of stimulus–response equivalence classes than did our previous approach. A major reason for this is that the stimulus regime we crafted through the iterative process had a much higher content of waveform segments containing maintained, multiple cycles of sine waves than do Gaussian white noise signals. Such multi-cycle segments are rare occurrences in GWN, and use of GWN to achieve the same level of confidence had required much longer experimental recording sessions. In some other neurons we have studied, preliminary evidence suggests that GWN signals contain such a small fraction of "relevant" stimulus features that a conventional system identification approach would never be practical, given realistic experimental constraints.

The neurobiological results themselves are enlightening, in that the approach demonstrates that non-linear encoding schemes are being used to represent information. In the case shown in Fig. 1, spike multiplets carry a quantity of information about characteristic stimulus features that is greater than the amount that could be extracted by a mean-rate decoder operating on the same stimulus waveforms.

An electronic system to enable execution of this iterative stimulus refinement and quantization analysis in real-time is currently being developed. This test-bed will enable real-time decoding of ensemble neural activity patterns and real-time interactive modulation of those neural activity patterns. The hardware devices supporting these tasks are being developed with advanced Digital Signal Processing and Field Programmable Gate Array technologies.

Acknowledgements.
Research supported by an NSF EIA-BITS grant (JPM,AGD,RS,TG), and by grants DGE9972824 (ZA,AEP) and MRI9871191, and by NIH grants MH12159 (AGD) and MH57179 (JPM, ZA, AGD).

References

1. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York, New York (1998)
2. Clague, H., Theunissen, F., Miller, J.P.: The effects of adaptation on neural coding by primary sensor interneurons in the cricket cercal system. J. Neurophysiol. 77 (1997) 207–220
3. Dimitrov, A.G., Miller, J.P.: Neural coding and decoding: communication channels and quantization. Network: Computation in Neural Systems 12 (2001) 441-472
4. Dimitrov, A.G., Miller, J.P., Aldworth, Z., Parker, A.: Spike pattern-based coding schemes in the cricket cercal sensory system. Neurocomputing 44-46 (2002) 373-379
5. Dimitrov, A.G., Miller, J.P., Gedeon, T., Aldworth, Z., Parker, A.E.: Analysis of Neural Coding using Quantization with an information-based distortion measure. Network: Computation in Neural Systems 14 (2003) 151-176
6. Gnatzy, H., Heusslein, W.: Digger wasp against crickets. I. receptors involved in the antipredator strategies of the prey. Naturwissenschaften 73 (1986) 212–215
7. Heinzel, H.G., Dambach, M.: Traveling air vortex rings as potential communication signals in a cricket. J. Comp. Physiol. A. 160 (1987)79–88
8. Kamper, G., Kleindienst, H.-U.: Oscillation of cricket sensory hairs in a low frequency sound field. J. Comp. Physiol. A. 167 (1990)193–200
9. Landolfa, M.A., Jacobs, G.A.: Direction sensitivity of the filiform hair population of the cricket cercal system. J. Comp. Physiol. A 177 (1995)758–767
10. Landolfa, M.A., Miller, J. P.: Stimulus-response properties of cricket cercal filiform hair receptors. J. Comp. Physiol. A. 177 (1995) 749–757
11. Miller, J.P., Jacobs, G.A., Theunissen, F.E.: Representation of sensory information in the cricket cercal sensory system. I. Response properties of the primary interneurons. J. Neurophysiol. 66 (1991) 1680–1689
12. Osborne, L.C.: Biomechanical Properties Underlying Sensory Processing in Mechanosensory Hairs in the Cricket Cercal Sensory System. PhD thesis, University of California, Berkeley, CA. (1997)
13. Roddey, J.C., Girish, B., Miller, J.P.: Assessing the performance of neural encoding models in the presence of noise. J. Comp. Neurosci. 8 (2000) 95–112
14. Roddey, J.C., Jacobs, G.A.: Information theoretic analysis of dynamical encoding by filiform mechanoreceptors in the cricket cercal system. J. Neurophysiol. 75 (1996) 1365–1376
15. Shimozawa, T., Kanou, M.: The aerodynamics and sensory physiology of a range fractionation in the cercal filiform sensilla of the cricket gryllus bimaculatus. J. Comp. Physiol. A. 155 (1984a) 495–505
16. Shimozawa, T., Kanou, M.: Varieties of filiform hairs: range fractionation by sensory afferents and cercal interneurons of a cricket. J. Comp. Physiol. A. 155 (1984b) 485–493
17. Stout, J.F., DeHaan, C.H., McGhee, R.W.: Attractiveness of the male *acheta domesticus* calling song to females. I. Dependence on each of the calling song features. J. Comp. Physiol. 153 (1983) 509–521
18. Theunissen, F.E., Miller, J.P.: Representation of sensory information in the cricket cercal sensory system. II. Information theoretic calculation of system accuracy and optimal tuning curve width of four primary interneurons. J. Neurophysiol. 66 (1991)1690–1703
19. Theunissen, F., Roddey, J.C., Stufflebeam, S., Clague, H., Miller, J.P.: Information theoretic analysis of dynamical encoding by four primary interneurons in the cricket cercal system. J. Neurophysiol. 75 (1996)1345–1364

Performance Evaluation of Generalized Polynomial Chaos

Dongbin Xiu, Didier Lucor, C.-H. Su, and George Em Karniadakis[1]

Division of Applied Mathematics, Brown University, Providence, RI 02912, USA,
gk@dam.brown.edu

Abstract. In this paper we review some applications of generalized polynomial chaos expansion for uncertainty quantification. The mathematical framework is presented and the convergence of the method is demonstrated for model problems. In particular, we solve the first-order and second-order ordinary differential equations with random parameters, and examine the efficiency of generalized polynomial chaos compared to Monte Carlo simulations. It is shown that the generalized polynomial chaos can be orders of magnitude more efficient than Monte Carlo simulations when the dimensionality of random input is low, e.g. for correlated noise.

1 Introduction

The generalized polynomial chaos, also called the Wiener-Askey polynomial chaos, was first proposed in [6] for solving stochastic differential equations. It is a generalization of the classical Wiener's polynomial chaos, which is defined as the span of Hermite polynomial functionals of a Gaussian process [4]. The Hermite-chaos expansion converges to any L^2 functional in the L^2 sense, according to the Cameron-Martin theorem [1]. It has been a useful tool in the study of multiple Itô integrals. In recent years, the Wiener's Hermite-chaos has been applied to the stochastic modeling of engineering applications, including various problems in mechanics [2, 3]. The more general framework of polynomial chaos employs many classes of orthogonal polynomials, and includes the Hermite-chaos as a subset. The main advantage of the generalized polynomial chaos is that it can represent many non-Gaussian stochastic processes, including some discrete processes, more efficiently [5–8].

In this paper, we review the theory and application of generalized polynomial chaos, and evaluate its performance by solving stochastic ordinary differential equations.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 346–354, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 The Generalized Polynomial Chaos

The generalized polynomial chaos is a means of representing second-order stochastic processes $X(\omega)$, viewed as a function of ω (the random event)

$$\begin{aligned} X(\omega) = & a_0 \Psi_0 \\ & + \sum_{i_1=1}^{\infty} a_{i_1} \Psi_1(\xi_{i_1}(\omega)) \\ & + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} a_{i_1 i_2} \Psi_2(\xi_{i_1}(\omega), \xi_{i_2}(\omega)) \\ & + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3} \Psi_3(\xi_{i_1}(\omega), \xi_{i_2}(\omega), \xi_{i_3}(\omega)) \\ & + \cdots, \end{aligned} \tag{1}$$

where $\Psi_n(\xi_{i_1}, \ldots, \xi_{i_n})$ denotes the generalized polynomial chaos of order n in the variables $(\xi_{i_1}, \ldots, \xi_{i_n})$, and are orthogonal polynomials in terms of the multi-dimensional random variables $\boldsymbol{\xi} = (\xi_{i_1}, \ldots, \xi_{i_n})$. For notational convenience, one can re-arrange the terms in equation (1), according to some numbering scheme, and rewrite the expansion as

$$X(\omega) = \sum_{j=0}^{\infty} \hat{a}_j \Phi_j(\boldsymbol{\xi}), \tag{2}$$

where there is a one-to-one correspondence between the functions $\Psi_n(\xi_{i_1}, \ldots, \xi_{i_n})$ and $\Phi_j(\boldsymbol{\xi})$, and their corresponding coefficients $a_{i_1 i_2 i_3 \ldots}$ and $\hat{a}_j$. Again $\{\Phi_j(\boldsymbol{\xi})\}$ are the (multi-dimensional) orthogonal polynomials in terms of the multi-dimensional random vector $\boldsymbol{\xi}$, satisfying the orthogonality relation

$$\langle \Phi_i, \Phi_j \rangle = \langle \Phi_i^2 \rangle \delta_{ij}, \tag{3}$$

where δ_{ij} is the Kronecker delta and $\langle \cdot, \cdot \rangle$ denotes the ensemble average. This is the inner product in the Hilbert space determined by the support of the random variables

$$\langle f(\boldsymbol{\xi}), g(\boldsymbol{\xi}) \rangle = \int f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) w(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{4}$$

with $w(\boldsymbol{\xi})$ denoting the weighting function. In the discrete case, the above orthogonal relation takes the form

$$\langle f(\boldsymbol{\xi}), g(\boldsymbol{\xi}) \rangle = \sum_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) w(\boldsymbol{\xi}). \tag{5}$$

In (2), there is a one-to-one correspondence between the type of the orthogonal polynomials $\{\Phi\}$ and the type of the random variables $\boldsymbol{\xi}$. This is determined by choosing the type of orthogonal polynomials $\{\Phi\}$ in such a way that

their weighting function $w(\boldsymbol{\xi})$ in the orthogonality relation (4) has the same form as the probability distribution function of the underlying random variables $\boldsymbol{\xi}$. For example, the weighting function of Hermite orthogonal polynomials is $\frac{1}{\sqrt{(2\pi)^n}} \exp(-\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi})$, and is the same as the probability density function of the $n-$dimensional Gaussian random variables $\boldsymbol{\xi}$. Hence, the classical Wiener polynomial chaos is an expansion of Hermite polynomials in terms of Gaussian random variables. Some types of generalized polynomial chaos corresponding to the commonly known distributions are listed in table 1.

Table 1. Correspondence of the type of Wiener-Askey polynomial chaos and their underlying random variables ($N \geq 0$ is a finite integer).

	Random variables $\boldsymbol{\xi}$	Wiener-Askey chaos $\{\Phi(\boldsymbol{\xi})\}$	Support
Continuous	Gaussian	Hermite-chaos	$(-\infty, \infty)$
	gamma	Laguerre-chaos	$[0, \infty)$
	beta	Jacobi-chaos	$[a, b]$
	uniform	Legendre-chaos	$[a, b]$
Discrete	Poisson	Charlier-chaos	$\{0, 1, 2, \dots\}$
	binomial	Krawtchouk-chaos	$\{0, 1, \dots, N\}$
	negative binomial	Meixner-chaos	$\{0, 1, 2, \dots\}$
	hypergeometric	Hahn-chaos	$\{0, 1, \dots, N\}$

The expansion (1) (or (2)) resides in the infinite dimensional space determined by $\boldsymbol{\xi}$, and is an infinite summation. In practice, we have to restrict ourselves to the finite-term summation. This is achieved by reducing the expansion to the finite-dimensional space, i.e. expansion of finite-dimensional random variables $\boldsymbol{\xi}$, according to the nature of random inputs; we also set the highest order of the polynomials $\{\Phi\}$ according to accuracy requirement. The finite-term expansion takes the form

$$X(\omega) = \sum_{j=0}^{M} \hat{a}_j \Phi_j(\boldsymbol{\xi}), \tag{6}$$

where $\boldsymbol{\xi}$ is an $n-$dimensional random vector. If the highest order of polynomial $\{\Phi\}$ is m, then the total number of expansion terms $(M+1)$ is, $(M+1) = (n+m)!/(n!m!)$.

3 Applications to Stochastic ODEs

In this section we apply the generalized polynomial chaos to the solution of stochastic ordinary differential equations with random parameters. We first consider the first-order ODE and demonstrate in detail the solution procedure; we then consider a second-order ODE with multiple random parameters which results in multi-dimensional polynomial chaos expansion. Hereafter, we restrict our discussion to the continuous probability distributions.

3.1 First-Order ODE

We consider the ordinary differential equation

$$\frac{dy(t)}{dt} = -ky, \qquad y(0) = \hat{y}, \tag{7}$$

where the decay rate coefficient k is considered to be a random variable $k(\omega)$ with certain probability density function (PDF) $f(k)$, zero mean value and standard deviation σ_k.

By applying the Wiener-Askey polynomial chaos expansion (6) to the solution y and random input k

$$y(t) = \sum_{i=0}^{M} y_i(t)\Phi_i, \qquad k = \sum_{i=0}^{M} k_i\Phi_i \tag{8}$$

and substituting the expansions into the governing equation, we obtain

$$\sum_{i=0}^{M} \frac{dy_i(t)}{dt}\Phi_i = -\sum_{i=0}^{M}\sum_{j=0}^{M} \Phi_i\Phi_j k_i y_j(t). \tag{9}$$

We then project the above equation onto the random space spanned by the orthogonal polynomial basis $\{\Phi_i\}$ by taking the inner product of the equation with each basis. By taking $< ., \Phi_l >$ and utilizing the orthogonality condition (3), we obtain the following set of equations:

$$\frac{dy_l(t)}{dt} = -\frac{1}{\langle \Phi_l^2 \rangle}\sum_{i=0}^{M}\sum_{j=0}^{M} e_{ijl}k_i y_j(t), \qquad l = 0, 1, \ldots, M, \tag{10}$$

where $e_{ijl} = \langle \Phi_i\Phi_j\Phi_l \rangle$. Note that the coefficients are smooth and thus any standard ODE solver can be employed here, e.g. Rouge-Kutta methods.

In Figure 1 we show the error convergence at time $t = 1$ of the Hermite-chaos expansion and Jacobi-chaos expansion, subject to Gaussian input and beta input, respectively. It can be seen on the semi-log plot that the errors decay exponentially fast as the highest order of expansion (P) increases. Given the simplicity of the equation, we can estimate the solution error of the finite-term chaos expansion. In particular, we estimate analytically the relative error in variance, denoted $\epsilon_2(t)$, for different distributions of k as a function of time [9]:

$$\epsilon_2(t) = \left(\frac{\sigma t}{1+\sigma t}\right)^{2(M-1)}, \tag{11}$$

$$\epsilon_2(t) \leq \frac{(\sigma t)^{2(M+1)}}{e^{(\sigma t)^2} - 1}\left[(M+1)!\left(1 - \frac{(\sigma t)^2}{M+1}\right)\right], \tag{12}$$

for the **Laguerre-chaos** when k has exponetial distribution and the **Hermite-chaos** when k has Gaussian distribution, respectively. Similar estimates can

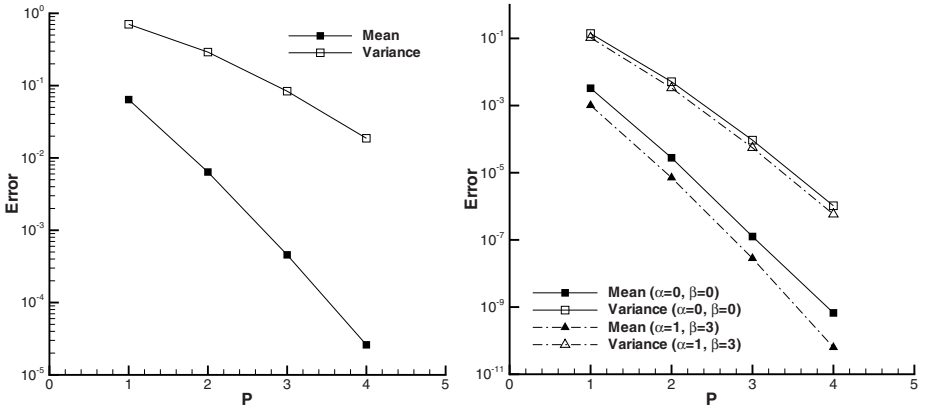


Fig. 1. Error convergence of the mean and the variance ($\sigma_k = 1$). Left: Hermite-chaos with Gaussian random input, Right: Jacobi-chaos with beta random input.

be applied to Legendre-chaos with uniform random input, although no explicit analytical formula is available and the estimation has to be evaluated numerically. By using these error estimates, we examine the number of expansion terms needed for a given error control threshold. The results of Hermite-chaos with Gaussian input and Legendre-chaos with uniform input are plotted in Figure 2, for fixed relative error of 10^{-7} in variance. It can be seen that the number of Hermite-chaos needed is larger than Legendre-chaos, and grows faster over time. For details of these estimates, see [9].

3.2 Second-Order ODE

In this section, we consider a linear oscillator subject to both random parametric and external forcing excitations, in the form

$$\begin{aligned} &\ddot{x}(t) + 2\zeta\omega_0\dot{x}(t) + \omega_0^2 x(t) = F(\omega)\cos(\omega t + \phi), \\ &x(0) = x_0, \qquad \dot{x}(0) = \dot{x}_0, \quad t \in [0, T] \end{aligned} \tag{13}$$

We assume the parameters and the forcing amplitude are random variables, i.e.

$$\begin{aligned} c &\doteq 2\zeta\omega_0 = \bar{c} + \sigma_c\xi_1 \\ k &\doteq \omega_0^2 = \bar{k} + \sigma_k\xi_2 \\ F &= \bar{F} + \sigma_F\xi_3, \end{aligned} \tag{14}$$

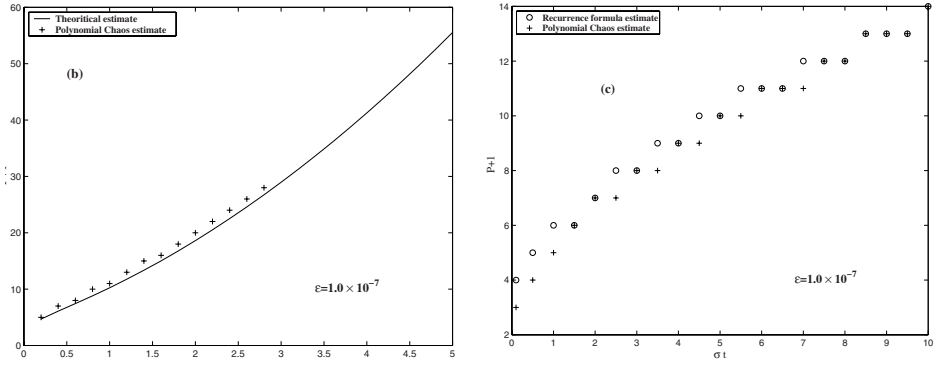


Fig. 2. Number of expansion terms needed for given relative error in variance $\epsilon_2 = 10^{-7}$. Left: Hermite-chaos with Gaussian random input, Right: Legendre-chaos with uniform random input.

where ξ_1, ξ_2 and ξ_3 are three *independent* random variables with zero mean; σ_c, σ_k and σ_F scales as the standard deviations of c, k and F, respectively. Here we will consider the uniform and Gaussian distributions. Correspondingly, the Legendre-chaos and Hermite-chaos will be employed.

The numerical integration is performed up to $T = 100$ when the solution reaches the asymptotic periodic state. We examine the convergence of the relative error in mean and variance at the final time versus the expansion order(see figure 3). It can be seen that the errors of the mean and variance decrease exponentially fast as the expansion order increases. However, in the Legendre-chaos the error in the *mean* and *variance* decay at the same rate in contrast to the Hermite-chaos.

4 Efficiency

The expanded equations from generalized polynomial chaos is a set of coupled equations, with each one of them resembles the deterministic equation. Since these equations are solved explicitly, the overall computational cost of generalized polynomial expansion is roughly the number of expansion terms times the cost of a single deterministic solution. On the other hand, the cost of Monte Carlo simulations is the number of realizations times that of a deterministic solution. Normally, thousands of realizations are needed for Monte Carlo simulation to obtain the solution statistics with moderate accuracy. Thus, the generalized

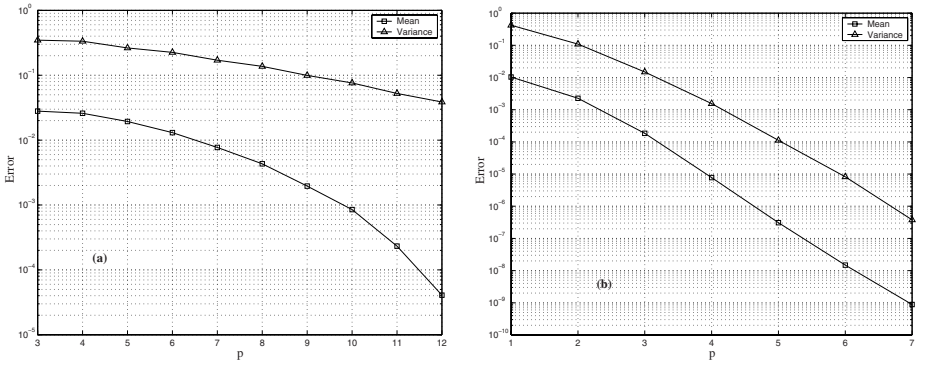


Fig. 3. Error convergence of the mean and variance. Left: Hermite-chaos with Gaussian random input; Right: Legendre-chaos with uniform random input.

polynomial chaos offers a great potential of computational speed-up due to its fast convergence rate.

For example, if k is an exponentially distributed random variable in the first-order ODE problem discussed in 3.1, the error convergence of the mean solution of the Monte-Carlo simulation and the corresponding Laguerre-chaos is shown in table 2.

Table 2. Speed-up factors S based on the relative error in mean (ϵ_{mean}) with exponential random input, for the first-order ODE problem (7). (S is defined as the cost of Monte Carlo simulations over that of generalized polynomial chaos.)

	ϵ_{mean}	Monte Carlo simulation (no. of realizations)	Generalized polynomial chaos: (no. of expansion terms)	S
Exponential	4%	100	1	100
	1.1%	1,000	2	500
	0.05%	9,800	3	3,267

For the second-order linear oscillator (13), we examine the efficiency of generalized polynomial chaos and Monte Carlo simulation by fixing the error requirement in mean at $T = 100$. Both uniform random inputs and Gaussian random inputs are considered, and the results are summarized in table 3. Smaller speed-up factors are observed for the Hermite-chaos expansion with Gaussian inputs,

compared with Legendre-chaos with uniform inputs. The advantage of the exponential convergence of chaos expansion is obvious, especially when smaller errors are required. *In the case of uniform random input with error requirement of* 0.001% *in mean, a speed-up factor of 17 millions is observed for Legendre-chaos expansion.*

Table 3. Speed-up factors S based on the relative error in mean (ϵ_{mean}) with Gaussian and uniform random inputs, for the second-order ODE problem (13).

	ϵ_{mean}	Monte Carlo simulation (no. of realizations)	Generalized polynomial chaos: (no. of expansion terms)	S
Gaussian	2%	350	56	6.25
	0.8%	2,150	120	18
	0.2%	33,200	220	151
Uniform	0.2%	13,000	10	13,000
	0.018%	$1,58 \times 10^6$	20	79,000
	0.001%	6.1×10^8	35	1.7×10^7

5 Conclusion

The efficiency of the generalized polynomial chaos can be orders of magnitude higher than the methods that require sampling, e.g. Monte Carlo method. This is especially obvious when the random inputs have low to moderate dimensions. However, if the random input has large dimensions, i.e. in the limit of white noise, the cost of generalized polynomial chaos grows fast, for a large number of expansion terms is required. In this case, the efficiency of the generalized polynomial chaos expansion is reduced drastically and the Monte Carlo method may be more efficient. This is still an unresolved problem for the generalized polynomial chaos and more work is required to resolve this issue.

Acknowledgements.

This work was supported by the ITR-DDDAS program of the National Science Foundation.

References

1. Cameron, R., Martin, W.: The orthogonal development of nonlinear functionals in series of Fourier-Hermite functionals. Ann. Math. **48** (1947) 385–392
2. Ghanem, R.: Ingredients for a general purpose stochastic finite element formulation. Comput. Methods Appl. Mech. Engrg. **168** (1999) 19–34

3. Ghanem, R., Spanos, P.: Stochastic Finite Elements: a Spectral Approach. Springer-Verlag, 1991
4. Wiener, N.: The homogeneous chaos. Amer. J. Math., **60** (1938) 897–936
5. Xiu, D., Karniadakis, G.: Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos. Comput. Methods Appl. Math. Engrg. **191** (2002) 4927–4948
6. Xiu, D., Karniadakis, G.: The Wiener-Askey polynomial chaos for stochastic differential equations. SIAM J. Sci. Comput. **24** (2002) 619–644
7. Xiu, D.,Karniadakis, G.: Modeling uncertainty in flow simulations via generalized polynomial chaos. J. Comput. Phys. (to appear)
8. Xiu, D., Lucor, D., Su, C., Karniadakis, G.: Stochastic modeling of flow-structure interactions using generalized polynomial chaos. J. Fluids Eng. **124** (2002) 51–59
9. Lucor, D., Su., C., Karniadakis, G.: Generalized polynomial chaos and random oscillators. Int. J. Numer. Methods Eng. (submitted)

Driving Scientific Applications by Data in Distributed Environments

Joel Saltz[1], Umit Catalyurek[1], Tahsin Kurc[1], Mike Gray[1], Shannon Hastings[1], Steve Langella[1], Sivaramakrishnan Narayanan[1], Ryan Martino[2], Steven Bryant[2], Malgorzata Peszynska[2], Mary Wheeler[2], Alan Sussman[3], Michael Beynon[3], Christian Hansen[3], Don Stredney[4], and Dennis Sessanna[4]

[1] Department of Biomedical Informatics, The Ohio State University
http://medicine.osu.edu/informatics
[2] Center for Subsurface Modeling, The University of Texas at Austin
http://www.ticam.utexas.edu/CSM
[3] Department of Computer Science, University of Maryland
http://www.cs.umd.edu/projects/adr
[4] Interface Laboratory, The Ohio Supercomputer Center
http://www.osc.edu

Abstract. Traditional simulation-based applications for exploring a parameter space to understand a physical phenomenon or to optimize a design are rapidly overwhelmed by data volume when large numbers of simulations of different parameters are carried out. Optimizing reservoir management through simulation-based studies, in which large numbers of realizations are sought using detailed geologic descriptions, is an example of such applications. In this paper, we describe a software architecture to facilitate large scale simulation studies, involving ensembles of long-running simulations and analysis of vast volumes of output data. This architecture is built on top of two frameworks we have developed: IPARS and DataCutter. These frameworks make it possible to implement tools and applications to run large-scale simulations, and generate and investigate terabyte-scale datasets efficiently.

1 Introduction

Numerical simulations provide a powerful mechanism for investigating and understanding complex systems and the interactions between various entities in those systems, and for effectively exploring design alternatives. In such applications, a large ensemble of simulations are carried out using different parameter values that descripe different potential initial states of the complex system under study. These applications are highly data-driven. Choosing the next set of simulations to be performed requires analysis of data from earlier simulations. As high-performance parallel and distributed platforms become more ubiquitous, traditional simulation approaches are overwhelmed by the vast volumes of data that need to be queried and analyzed. In this paper, using oil reservoir management applications as an example, we describe the design and a prototype implementation of a software system for large scale simulation studies.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 355–364, 2003.
© Springer-Verlag Berlin Heidelberg 2003

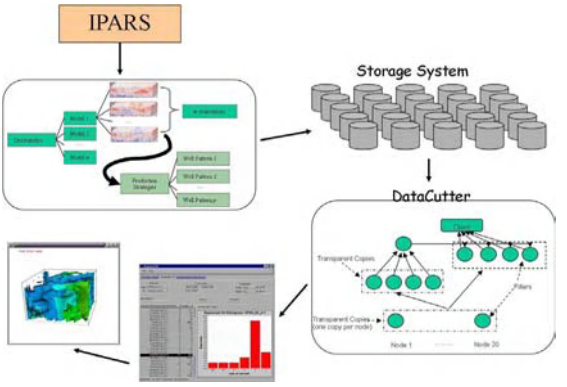


Fig. 1. Software system architecture

Numerical simulation of oil and gas reservoirs can aid the design and implementation of optimal production strategies. With a better understanding of oil and gas produced from existing reservoirs, better techniques can be devised to locate new reserves and maximize oil production from the existing reserves. Complex models of subsurface allow better assessment of the risk to the environment of existing and new reservoirs and remediation and storage of hazardous waste.

Despite technological advances in methods of determining reservoir properties, operators still have at best a partial knowledge of critical parameters such as rock permeability which govern production rates. Thus a major challenge to these objectives is incorporating geologic uncertainty while maintaining operational flexibility in large, detailed flow models. One approach to this problem is to simulate alternative production strategies (number, type, timing and location of wells) applied to multiple realizations of multiple geostatistical models. In a typical study, a scientist runs an ensemble of simulations to study the effects of varying oil reservoir properties (e.g., permeability, oil/water ratio, etc.) over a long period of time. With the help of high-performance computers, even for relatively coarse descriptions, this approach can lead to unmanageably large volumes of output data. Storage, analysis and visualization of large volumes of data generated by an ensemble of simulations is key to achieve a better understanding and characterization of oil reservoirs.

Figure 1 illustrates the overall architecture of the software system for large scale oil reservoir simulation studies. The system consists of two main frameworks that we have developed.

- **IPARS (Integrated Parallel Accurate Reservoir Simulator)** is a framework that supports multiple physical models and algorithms for the solution of multiphase flow and transport problems in porous media. The framework provides common memory management for general geometric grids, portable parallel communication, linear solvers with state-of-the-art preconditioners, keyword input, and output with visualization.

- **DataCutter** is a middleware framework for subsetting and processing multi-dimensional datasets in a distributed environment. The application processing structure is implemented as a set of interacting components, referred to as filters. Application filters can be placed on the machines in a setting so that communication and computation overheads are minimized.

In this architecture, IPARS is used to simulate alternative production strategies for a large number of geostatistical realizations of a hypothetical reservoir. Output from IPARS is stored on distributed collections of disk-based storage systems for interactive data analysis. Using the DataCutter framework, various data analysis operations can be implemented that query and manipulate those datasets. These operations can be executed on the storage systems where the datasets are stored or on other machines dispersed across a network. A graphical user interface allows a scientist to formulate queries to carry out different analysis scenarios, such as economic ranking of the alternatives, exploration of the physical basis for differences in behavior between realizations, in particular to identify regions of bypassed oil, and identification of representative realizations, which could be used as a basis for further optimization. Visualization of the datasets can be carried out remotely.

In the rest of the paper, we present the components of the software system in more detail. We describe the implementation of several data analysis scenarios for a sample case study.

2 System Components

2.1 IPARS Framework

IPARS [6, 18] represents a new approach to reservoir simulator development, emphasizing modularity of code, portability to many platforms, and ease of integration with other software. It models multiphase, multiphysics flow in porous media, and is suitable for massively parallel computers or clusters of workstations. There are currently ten physical models in IPARS, including multiphase gas–oil–water and air–water flow and reactive transport models. These models can be coupled for multiphysics simulations including couplings between IPARS models or with external codes. For example, the IPARS black–oil model was used in a loosely coupled geomechanics and flow implementation driven by reservoir subsidence problems. Solvers used by IPARS employ state-of-the-art techniques for nonlinear and linear problems including multigrid and other preconditioners.

A key feature of the IPARS framework is that it explicitly builds upon the multiblock multiphysics approach ([14, 18, 10]) which allows for the mathematically rigorous treatment of multiple domains in which different physical processes are occurring, as well as providing a basis for implementing different numerical schemes in different parts of the domain using nonmatching grids while preserving mass and momentum conservation.

The black-oil model implemented in IPARS is a three phase (water, oil and gas) model describing the flow in a petroleum reservoir [13] with three components. As such it can be considered as a subset of a *compositional* model [13].

Here it is assumed that no mass transfer occurs between the water phase and the other two phases and that water phase can be identified with water component. In the hydrocarbon (oil-gas) system, only two components: light and heavy hydrocarbons, are considered. The black-oil model described here has been shown [11] to give accurate results by comparing them with available analytical solutions and in other cases with solutions obtained by a recognized industrial reservoir simulation tool Eclipse [4]. In addition, the models and solvers under IPARS Framework were shown to be scalable in parallel [17].

2.2 DataCutter

A number of toolkits integrate processing with parallel data retrieval on tightly-coupled systems [5, 3]. Component-based frameworks provide an viable programming environment for application development in distributed environments. Besides the ease of complex application development, such models facilitate application implementations that can adapt to the heterogeneous and dynamic nature of the environment. Several research projects have focused on developing different types of component-based models [7, 12, 15].

DataCutter [2, 1] is a component framework designed to support subsetting and processing of large datasets. DataCutter implements a filter-stream programming model for developing data-intensive applications. In this model, the application processing structure is implemented as a set of components, referred to as *filters*, that exchange data through a *stream* abstraction. The interface for a *filter*, consists of three functions: (1) an initialization function (*init*), in which any required resources such as memory for data structures are allocated and initialized, (2) a processing function (*process*), in which user-defined operations are applied on data elements, and (3) a finalization function (*finalize*), in which the resources allocated in *init* are released. Filters are connected via *logical streams*. A *stream* denotes a uni-directional data flow from one filter (i.e., the producer) to another (i.e., the consumer). A filter is required to read data from its input streams and write data to its output streams only. We define a *data buffer* as an array of data elements transferred from one filter to another. The current implementation of the logical stream delivers data in fixed size buffers, and uses TCP for point-to-point stream communication.

The overall processing structure of an application is realized by a *filter group*, which is a set of filters connected through logical streams. When a filter group is instantiated to process an application query, the runtime system establishes TCP/IP socket connections between filters placed on different hosts before starting the execution of the application query. Filters placed on the same host execute as separate threads. An application query is handled as a *unit of work* (UOW) by the filter group.

The programming model provides several abstractions to facilitate performance optimizations. A *transparent filter copy* is a copy of a filter in a filter group. The filter copy is transparent in the sense that it shares the same *logical* input and output streams of the original filter. A transparent copy of a filter can be made if the semantics of the filter group are not affected. That is, the output

of a unit of work should be the same, regardless of the number of transparent copies. The transparent copies enable data-parallelism for execution of a single query, while multiple filter groups allow concurrency among multiple queries. The filter runtime system maintains the illusion of a single logical point-to-point stream for communication between a logical producer filter and a logical consumer filter. For distribution between transparent copies, the runtime system supports a Round-Robin (RR) mechanism and a Demand Driven (DD) mechanism based on the buffer consumption rate. DD aims to send buffers to the filter that would process them fastest. When a consumer filter starts processing of a buffer received from a producer filter, it sends an acknowledgment message to the producer filter to indicate that the buffer is being processed. A producer filter chooses the consumer filter with the minimum number of unacknowledged buffers to send a data buffer to, thus achieving a better balancing of the load.

3 A Case Study

In this section, we describe a case study we have implemented using IPARS and DataCutter. This case study involves generation of a large collection of data sets from IPARS simulations, and implementation and execution of various data exploration scenarios.

3.1 Data Generation

We have generated a large dataset consisting of 207 separate realizations using the IPARS simulation framework. The input data for this dataset is based on the industry benchmark SPE9 problem[8] and comes from a black-oil (three phase) flow problem on a grid with 9,000 cells. At each time step, the value of seventeen separate variables is output for each node in the grid. A total of 10,000 time steps are taken and the total output stored for each realization is about 6.9 GB. The total of 207 realizations were taken from among 18 geostatistical models and 4 well configurations/production scenarios. The geostatistical models are used to randomly generate permeability fields that are characterized by statistical parameters such as covariance and correlation length. The total dataset size is roughly 1.5 Terabytes[1] and was generated and stored on a storage cluster of 50 Linux nodes (PIII-650, 128MB, Switched Fast Ethernet) with a total disk storage of 9TB.

3.2 Data Exploration Scenarios

We have implemented several data exploration scenarios using the DataCutter framework. These scenarios involve user-defined queries for economic evaluation

[1] The case study that is described in this paper was demonstrated at Supercomputing 2001; a study with a larger dataset (5 Terabytes), which is distributed across three sites (San Diego Supercomputer Center, University of Maryland, and Ohio State University), was demonstrated at Supercomputing 2002. We plan to report on the performance evaluation of the latter study in a future work.

as well as technical evaluation, such as determination of representative realizations and identification of areas of bypassed oil.

Economic Evaluation In the optimization of oil field production strategies, the objective function to be maximized is the resulting economic value of a given production strategy. The value can be measured in a variety of ways. In our model we compute both the net present value (NPV) and return on investment (ROI). In the computation of the NPV for a given realization, a query integrates over time the revenue from produced oil and gas, and the expenses from water injection and production, accounting for the time value of the resources produced. This calculation is performed for a subset of the realizations chosen by the user. As all of the well production and injection data for each realization resides in a single file in a single disk, the data access pattern for this application is relatively simple, and most computation time is spent parsing the output file. The well data is also a relatively small part of the output data at each time step, so this is not a compute and data intensive computation. Presently, the operations for the economic evaluation is implemented as a single DataCutter filter, which also performs data retrieval from the storage system.

Bypassed Oil Depending on the distribution of reservoir permeability and the production strategy employed, it is possible for oil to remain unextracted from certain regions in the reservoir. To optimize the production strategy, it is useful to know the location and size of these regions of bypassed oil. To locate these regions, the user selects a subset of datasets (D), a subset of time steps (T), minimum oil saturation value ($O_{s,tol}$), maximum oil velocity ($V_{o,tol}$), and minimum number of connected grid cells (N_c) for a bypassed oil pocket. The goal is to find all the datasets in D that have bypassed oil pockets with at least N_c grid cells. A cell (C) is a potential bypassed oil cell if $S_{o,c} > S_{o,tol}$ and $V_{o,c} < V_{o,tol}$.

We implemented a set of filters that carry out the various operations required to find the bypassed oil regions. The implementation consists of three filters. **RD – Read data filter** retrieves the data of interest from disk and writes the data to its output stream. A data buffer in the output stream contains oil velocity and oil saturation values, and corresponds to a portion of the grid at a time step in a data set. **CC – Connected component filter** performs operations to find bypassed oil pockets at a time step on data buffer received from **RD**. These oil pockets are stored in a byte array, passed to the next filter in the pipeline. Each entry of the byte array denotes a grid cell and stores if the cell is bypassed oil cell or not. The **CC** filter writes the data buffer for each time step to the output stream, which connects **CC** to the **MT** filter. **MT – Merge over time filter** performs an AND operation on the data buffers received from CC, and finds the bypassed oil pockets. The result is sent to the client.

This scenario accesses the large four-dimensional (three spatial dimensions and time) datasets which are output for each realization. Each of the output variables are written to separate files, so this computation involves the subsetting

of data spread across several files. Additionally, if the simulation was run in parallel, the data for different parts of the domain could reside on separate disks or nodes.

Representative Realization Running multiple realizations with the same geostatistical model and well configurations can give an idea of the upper and lower bounds of performance for a particular production strategy. It is also of interest to find one realization for a given production scenario that best represents the average or expected behavior. A client query to find a representative realization for a given subset of realizations computes the average of the primary IPARS unknowns – oil concentration (C_o), water pressure (W_p), gas pressure (G_p) – and then finds the realization in the subset which is closest to the average in the sense that

$$\min \sum_{\text{all grid points}} \frac{|C_o - C_{o_avg}|}{C_{o_avg}} + \frac{|P_w - P_{w_avg}|}{P_{w_avg}} + \frac{|P_g - P_{g_avg}|}{P_{g_avg}}$$

is realized.

The DataCutter implementation consists of four filters. **RD - Read filter** retrieves the data of interest from disk. The read filter sends data from each dataset to the **SUM** and **DIFF** filters. A data buffer from the read filter is a portion of the grid at one time step. **SUM - Sum filter** computes the sum for C_o, W_p, and G_p variables at each grid point across the datasets selected by the user. **AVG - Average filter** calculates the average for C_o, W_p, and G_p values. **DIFF - Difference filter** finds the sum of the differences between the grid values and the average values for each dataset. It sends the difference to the client, which keeps track of differences for each time step, carries out average over all time steps for each dataset.

3.3 Visualization

We have developed two different implementations for visualizing output from a realization. The first visualization employs isosurface rendering implemented using the Active Data Repository (ADR) framework [3], which is designed to support processing of large, out-of-core datasets via generalized reduction operations on distributed memory systems. The ADR implementation uses the marching cubes and polygon rendering functions of the Visualization Toolkit (VTK) [16] for extracting and rendering an iso-surface from large, out-of-core datasets on a distributed-memory parallel system with a local disk farm [9]. Figure 2 shows a visualization of bypassed oil regions using the isosurface rendering. The second visualization tool is based on direct volume rendering. We implemented a DataCutter filter using the volume rendering library, developed at Ohio Supercomputer Center. This implementation uses a texture based volume rendering approach and takes advantage of 3D hardware texture rendering (e.g., NVIDIA GeForce 3) cards.

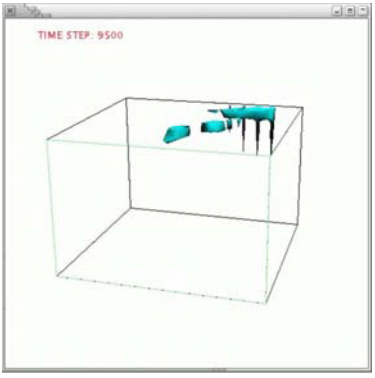
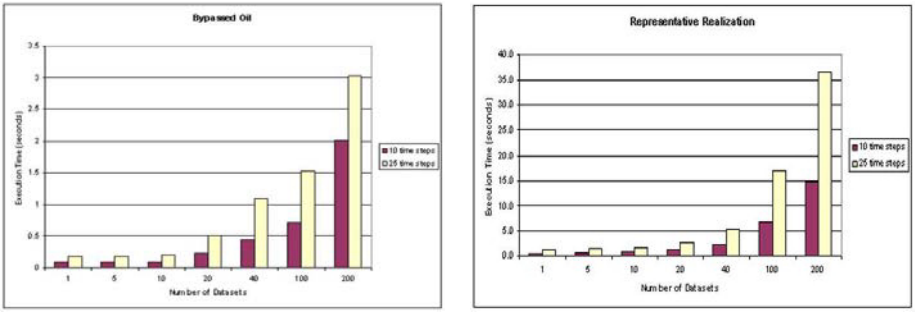


Fig. 2. Visualization of bypassed oil

4 Results

We present results for the representative realization and bypassed oil scenarios. The experiments were carried out using the DataCutter implementations of the two scenarios and the 1.5TB data set generated in this study. The data set is stored on 19 nodes of the 9.5TB storage cluster at University of Maryland. In the experiments, we varied the number of data sets accessed by a query. For this purpose, we submitted a total of 28 queries (varying the number of datasets from 1 to 200); 14 queries for the bypassed oil analysis and 14 queries for the representative realization analysis. Half of the queries in each set requests data over 10 time steps (time steps 0 through 9999, with increments of 1000 time steps), while the other half retrieves 25 time steps (time steps 0 through 9999 with increments of 400 time steps).

Figures 3 shows the execution time for each query. As is seen from the figures, up to 40 datasets the query execution time remains below 1 seconds for the bypassed oil scenario. As the number of datasets is increased the query execution time increases, as expected. For a query that accesses 200 datasets over 25 time steps, the execution time is about 3 seconds. Thus, we are able to achieve interactive rates even for queries that access a large number of datasets from the collection. The experimental results show that queries for representative realization scenario take longer, as the operations involved are more expensive. As is seen from the figure, the query execution time remains below 5 seconds for queries that access up to 40 datasets over 25 time steps. Our preliminary results show that the query execution does not scale well after 40 datasets for the representative realization scenario. This is because of the fact that in the experiments the number of transparent copies for the SUM and DIFF filters are fixed at four. In future work, we plan to examine the effect on performance of varying the number of transparent copies.



(a) (b)

Fig. 3. (a) Performance results for bypassed oil computation (b) Performance results for representative realization computation

5 Conclusions

We have demonstrated a new paradigm for applying reservoir simulation to the challenges of reservoir management. The selected challenge was to enable the evaluation of large numbers of realizations, both of geological models and of well patterns. The black oil model within the IPARS framework provided the numerical solutions to the forward flow problems, while the DataCutter middleware provided the means for subsetting and filtering the multidimensional output. The volume of data resulting from such studies can be extremely large. Such datasets would be unmanageable for most evaluation tools, especially for complex queries such as identifying representative realizations or locating regions of bypassed oil. The IPARS/DataCutter applications enable the creation, interrogation and visualization of such datasets while maintaining the familiarity and speed of interaction of the traditional simulation workflow. Thus many more realizations of higher resolution geologic models and more production strategies can be studied in greater detail within a given time, increasing the utility of the study for decision making.

Acknowledgments. The work described in this paper was partly supported under the grant of National Partnership for Advanced Computational Infrastructure (NPACI) 10181410. In additional, the first four authors were supported partially by grants DOE: DE-FG03-99ER25371, DOD: PET2/ UTA01-347, and the NSF grants: SBR 9873326, ITR EIA-0121523. This research was also supported in part by the National Science Foundation under Grants #ACI-9619020 (UC Subcontract #10152408), #EIA-0121177, #EIA-0203846, #ACI-0130437, and #ACI-9982087, Lawrence Livermore National Laboratory under Grant #B500288 and #B517095 (UC Subcontract #10184497), and the Department of Defense, Advanced Research Projects Agency, USAF, AFMC through Science Applications International Corporation under Grant #F30602-00-C-0009 (SAIC Subcontract #4400025559).

References

1. M. Beynon, C. Chang, U. Catalyurek, T. Kurc, A. Sussman, H. Andrade, R. Ferreira, and J. Saltz. Processing large-scale multidimensional data in parallel and distributed environments. *Parallel Computing*, 2002.

2. M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Computing*, 27(11):1457-1478, Oct. 2001.
3. C. Chang, T. Kurc, A. Sussman, and J. Saltz. Optimizing retrieval and processing of multi-dimensional scientific datasets. In *Proceedings of the Third Merged IPPS/SPDP (14th International Parallel Parallel and Distributed Processing Symposium)*. IEEE Computer Society Press, May 2000.
4. Eclipse-100, Technical Description. Schlumberger Technology Corporation. 1998.
5. S. Goil and A. Choudhary. PARSIMONY: An infrastructure for parallel multidimensional analysis and data mining. *Journal of Parallel and Distributed Computing*, 61(3):285-321, March 2001.
6. *IPARS: Integrated Parallel Accurate Reservoir Simulator.* http://www.ticam.utexas.edu/CSM/ACTI/ipars.html.
7. C. Isert and K. Schwan. ACDS: Adapting computational data streams for high performance. In *14th International Parallel & Distributed Processing Symposium (IPDPS 2000)*, pages 641-646, May 2000. IEEE Computer Society Press.
8. J. E. Killough. Ninth (SPE) comparative solution projection: a reexamination of black-oil simulation. The 13th Symposium on Reservoir Simulation, pages 135-147, February 1995. SPE 29110.
9. T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Visualization of large datasets with the Active Data Repository. *IEEE Computer Graphics and Applications*, 21(4):24-33, July/August 2001.
10. Q. Lu, M. Peszynska, and M. F. Wheeler. A parallel multi-block black-oil model in multi-model implementation. In *2001 SPE Reservoir Simulation Symposium*, Houston, Texas, 2001. SPE 66359.
11. Q. Lu. *A Parallel Multi-Block / Multi-Physics Approach for Multi-Phase Flow in Porous Media.* PhD thesis, University of Texas at Austin, Austin, Texas, 2000.
12. R. Oldfield and D. Kotz. Armada: A parallel file system for computational. In *Proceedings of CCGrid2001: IEEE International Symposium on Cluster Computing and the Grid*, May 2001. IEEE Computer Society Press.
13. D. W. Peaceman. *Fundamentals of numerical reservoir simulation.* Elsevier Scientfic Publishing Company, Amsterdam-Oxford-New York, first edition, 1977.
14. M. Peszynska. Advanced techniques and algorithms for reservoir simulation III. Multiphysics coupling for two phase flow in degenerate conditions. In John Chadam, Al Cunningham, Richard E. Ewing, Peter Ortoleva, , and Mary Fanett Wheeler, editors, *IMA Volumes in Mathematics and its Applications, Volume 131: Resource Recovery, Confinement, and Remediation of Environmental Hazards*, pages 21-40. Springer, 2002.
15. B. Plale and K. Schwan. dQUOB: Managing large data flows using dynamic embedded queries. In *IEEE International High Performance Distributed Computing (HPDC)*, August 2000.
16. W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics.* Prentice Hall, 2nd edition, 1997.
17. M. F. Wheeler, M. Peszynska, X. Gai, and O. El-Domeiri. Modeling subsurface flow on PC cluster. In A. Tentner, editor, *High Performance Computing*, pages 318-323. SCS, 2000.
18. M. F. Wheeler, J. A. Wheeler, and M. Peszynska. A distributed computing portal for coupling multi-physics and multiple domains in porous media. In L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, editors, *Computational Methods in Water Resources*, pages 167-174. A. A. Balkema, 2000.

Simulating Sellers' Behavior in a Reverse Auction B2B Exchange*

Subhajyoti Bandyopadhyay[1], Alok R. Chaturvedi[2], John M. Barron[2], Jackie Rees[2], and Shailendra Mehta[2]

[1] University of Florida, 364 Stuzin Hall, Gainesville, FL 32611-1310
shubho.bandyopadhyay@cba.ufl.edu
[2] 1310 Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907-1310
{alok, barron, jrees, mehta}@mgmt.purdue.edu

Abstract. Previous research in reverse auction B2B exchanges found that in an environment where sellers collectively can cater to the total demand, with the final (i.e. the highest-priced bidding) seller catering to a residual, the sellers resort to a mixed strategy equilibrium [2]. While price randomization in industrial bids is an accepted norm, it may be argued that managers in reality do not resort to advanced game theoretic calculations to bid for an order. What is more likely is that managers learn that strategy and over time finally converge towards the theoretic equilibrium. To test this assertion, we model the two-player game in a synthetic environment, where the agents use a simple reinforcement learning algorithm to put progressively more weights on selecting price bands where they make higher profits. We find that after a sufficient number of iterations, the agents do indeed converge towards the theoretic equilibrium.

1 Introduction

The recent industry excitement around business-to-business (B2B) e-commerce has brought into focus the large potential size of the market as well as the mechanisms that are expected to bring multiple benefits to the participants of B2B exchanges. Forrester Research predicts that there will be $1.5 trillion in goods and services transacted online among U.S. businesses domestically by 2003, and $2.5 trillion internationally [14]. Therefore, it is critical to attempt to better understand the dynamics comprising this particular type of market.

Some of the more prominent advantages B2B exchanges are expected to bring include lower costs due to automating the procurement process, reverse auctions, interoperability among users, collaborative planning and collaborative

* This research is partially funded by the National Science Foundation Grant No. DMI-0122214.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 365–374, 2003.
© Springer-Verlag Berlin Heidelberg 2003

design [10]. For example, Ford announced in July 2001 that it had saved $70 million through Covisint (the online automotive exchange by the Big Three automakers) in terms of reduced paperwork and lower supplier prices, which is more than its initial investment in the exchange [10]. Bandyopadhyay, Barron and Chaturvedi [2] analyzed the competition between sellers in reverse auctions in a game-theoretic framework, and established the Nash equilibria in several scenarios. It was found that in an environment where sellers can collectively cater to the total demand, with the final (i.e. the highest-priced bidding) seller catering to a residual, the sellers resort to a mixed strategy Nash equilibrium. While price randomization in industrial bids is an accepted norm, it may be argued that managers in reality do not resort to advanced game theory calculations to bid for an order. What is more likely is that managers learn that strategy over time and finally converge towards the theoretic equilibrium. This paper tests that assertion by creating an artificial B2B exchange in the synthetic environment for analysis and simulation (SEAS) [6]. It models the sellers' behavior with artificial software agents that start bidding randomly, and use a simple reinforcement learning (RL) algorithm to "learn" the ideal strategy over time.

The competition among sellers in the environment mentioned above is different from the traditional oligopolistic Cournot competition between firms facing a downward facing demand curve, where both firms sell at the same price point. It is also distinct from a capacity-constrained Bertrand model that has been analyzed extensively in the literature, where a quantity precommitment and Bertrand competition yield Cournot outcomes that have equilibrium prices above marginal cost [12].

The analysis of Bandyopadhyay et al. [2] also established the nature of the equilibrium under various assumptions of the sellers' cost, capacities and the market demand. The problem is most interesting when we assume that there is no *combined* capacity constraint as such: the sellers were supplying to the entire demand before the birth of the exchange, and continue to do so after it comes into play. However, it is conceivable that the firms individually cannot supply to the entire market (in fact, it is to their benefit not to have too much capacity, when there is industry overcapacity in the first place, as then the competition reduces to a Bertrand game with all firms supplying at their marginal cost). Since the set of suppliers is limited and all are reputed in the marketplace, the buyers would not mind getting their orders fulfilled by any one or several of these suppliers. This means that while there is a competition between the firms to be the low-price bidder, it is not as extreme as a Bertrand game that results in prices equal to marginal cost. However, there remains an incentive to be the low-price bidder and have the "first invitation" to supply a requirement.

We use an agent-based simulation approach to study the sellers' behavior in a B2B market place. The use of simulation allows repeated and detailed study of the behaviors exhibited by the players in the market under various experimental treatment conditions. These artificial agents are endowed the ability to learn from previous actions by the use of a type of Reinforcement Learning (RL) algorithm described below. Artificial agents have been used to simulate human agents or players in a number of different settings such as for automated negotiations in an e-commerce environment [7, 16].

RL has been used to examine various competitive scenarios such as sealed bid k-double auction under asymmetric and incomplete information dynamics [17], market entry games [8] and rule learning in repeated games [3]. RL is an appropriate choice for the application presented in this paper due to the ability of RL agents to incorporate previous experience (either reward or no reward) into action. The model under which the artificial agents operate in this research is discussed in the next section.

The remainder of the paper is organized as follows: The model of the reverse auction is provided in Section 2. Section 3 presents the RL algorithm deployed in the simulation. Section 4 states the research assertions, hypotheses that are tested in the simulation, and provides the results. Conclusions and future research directions are discussed in Section 5.

2 The Model

SEAS uses intelligent agents (IA) to represent economic realities of electronic markets and hierarchies in a decentralized manner [5]. SEAS' intelligent agents are autonomous processes that are adaptive and behave like human agents in a narrow domain. In their respective domains, each agent has a well-defined set of responsibilities and authorities so that it can execute its tasks effectively. Examples of SEAS' IAs are: economic agents --consumer IAs, producer IAs, and regulator IAs; political agents -- government IAs, special interest IAs, etc. An agent in SEAS is equipped with reasoning, action, and communication skills required for performing their respective tasks. A SEAS IA is characterized by the knowledge it possesses.

SEAS markets are implemented in JavaSpace (Figure 1). JavaSpace is a descendant of the Linda system developed at Yale University [4]. Java Space defines the market structure. The rules for all the markets are implemented through JavaSpace, which in turn synchronizes the thread between the agents. Agents maintained in the space are updated through their working status. JavaSpace also forms the connectivity between the agents and the Database, and therefore, after completing the transactions, it updates the database.

Let us suppose that there are two buyers who bought from two sellers (one from each) before the advent of the exchange. What prevented buyers from establishing contact with both the sellers (and vice versa) are the search costs and the ongoing cost of establishing relationships within large organizations (dedicated account management teams for buyers, sales force for sellers, cost of sending individual RFQs to the entire universe of sellers, etc.) [11].[1] With the lack of competition, the sellers could afford to sell the required quantities to the buyers at their reservation price, which we assume to be the same for both buyers at r. With the advent of the exchange, the buyers put forward their requirements to the exchange, and the sellers can then bid for the total requirement from both buyers. The above example is just illustrative, and is not crucial to the analysis. There can be in fact only

[1] It has been estimated that in terms of reduction of paperwork alone, B2B exchanges can bring down costs per purchase order from $75-$150 to $10-$30 [10]

a single large buyer, whose requirements cannot be met by one seller – however, two sellers together have a combined capacity that is more than the buyer's requirement.

SEAS B2B model derives much of its intuition from the basic 2-player model, and therefore it is instructive to first consider the 2-player model in detail. We consider the case when both sellers have equal capacities K that is more than the respective individual requirements of the buyers, but is lesser than the total requirement of both buyers Q (i.e. $2K - Q > 0$). In such a setting, the lower priced seller is invited first to sell the required quantity, and after he has supplied his total capacity K, the other seller can then sell the residual demand $Q - K$. Both sellers have a common fixed marginal cost of production, c.

From the modeling point of view, it is immaterial whether the sellers respond to an aggregate demand of several buyers or one single demand from a buyer. What is important to note is that the entire requirement is auctioned to the sellers, and for any unfulfilled demand, a lower priced bidder is invited before a higher priced bidder to satisfy the unfulfilled demand. It is readily apparent that with unlimited capacity, the sellers respond with a Bertrand competition in prices with the seller or sellers with the lowest marginal cost outbidding the others. This is not to the advantage of the sellers. Kreps and Scheinkman [12] (and several variants of the original model, such as [1])) show that if sellers could limit capacity, then a quantity precommitment and Bertrand competition yield Cournot outcomes that have equilibrium prices above marginal cost. At the other end of the spectrum, if the total capacity of the sellers is so limited as to be less than the total demand, it is easy to see that the sellers can sell their entire capacities at the buyer's reservation price.

The analysis shows that there exists a mixed strategy equilibrium of prices where the sellers randomize between a range of prices. The intuition behind such an equilibrium is as follows: With two similar players, there cannot be any equilibrium in pure strategies with the players settling on different prices. Settling on the same price is also ruled out, since the best response to any price is to set a price that is ε lower than that price. Thus, if any Nash equilibrium exists, it has to be mixed strategy equilibrium. It can further be shown that the support of the strategy lies between p_1 and r, where r is the reservation price for the buyer and p_1 is given by

$$. p_1 = \frac{(r-c)(Q-K)}{K} + c \tag{1}$$

An intuitive way of looking at p_1 is that below this price, a seller makes less profit by "winning" (supply to capacity) than by "losing" and supply the residual at the highest possible price r (which is the best price the seller can supply the residual, since he is losing anyway).

The equilibrium strategy for either player can be expressed in terms of their price randomizing cumulative probability density function *F(p)*:

$$. F(p) = \frac{(p-c)K - (r-c)(Q-K)}{(p-c)(2K-Q)} \tag{2}$$

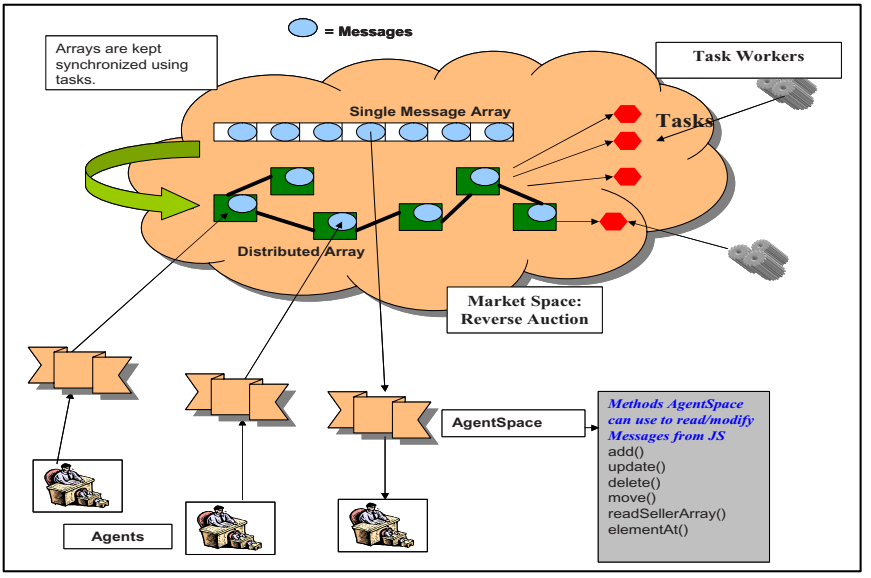


Fig. 1. SEAS Agent-based Architecture

This is a continuous probability distribution within the range (p_1, r), and effectively defines the symmetric Nash equilibrium strategy of the two players (i.e. the suppliers). The suppliers randomize their bids within this interval, such that their randomizing has a probability distribution given by *F(p)* in (2). By definition the Nash equilibrium maximizes the expected return of the suppliers.

The analysis will be similar for the n-player model, where the highest bidder supplies the residual, and the rest supply to capacity ($(n-1)K < Q < nK$). The support of the strategy for the sellers is given by (p_1^n, r), where

$$p_1^n = \frac{(r-c)(Q-(n-1)K)}{K} + c, \tag{3}$$

and the expression for the distribution function is given by

$$F_n(p) = \left[\frac{(p-c)K - (Q-(n-1)K)(r-c)}{(p-c)(nK-Q)} \right]^{1/n-1} \tag{4}$$

While price randomization in industrial bids is an accepted norm, it might be difficult to accept that managers go through advanced game theory calculations (and in any case, the real-life situations are far more varied than the simplified model scenarios that make any game theory analysis extremely complex) to determine their bids. It is conceivable that sellers learn from their past experiences to bid in a fashion that maximizes their surplus. It is this assertion that we test in the remainder of this paper.

3 The Algorithm

To test our assertion, we model the competing sellers as artificial software agents. Like human subjects, we propose that these agents "understand" the following (without resorting to knowing game theory):

1. There are two opposing forces in the pricing strategy – a higher price (towards *r*) means greater per-unit profit, but also brings about a higher probability of "losing" to the competition (in terms of being the first invited bidder to supply the demand).
2. It does not make any sense to price below p_1, as is clear from the above analysis.
3. Since there is a need to balance between higher probability of winning and higher per-unit profit, there is no a priori reason to rule out any price between p_1 and *r*, and further, there is reason to believe that price randomizing might be the ideal (or equilibrium) solution.

The "sellers" thus start off initially with a totally random pricing strategy (i.e. the price distribution is uniform in its support), with the hope of learning over time about the ideal nature of the randomization. This is the same assumption as Erev and Roth [8] employ in their experiments and refer to as the "initial propensities" of the players for their pure strategies.

If we denote the average profit of player *j* (*j*=1,2) in division *k* (*k*=1,…10) in simulation round *t* as $\Pi_{jk}(t)$, then the probability $p_{jk}(t+1)$ of choosing that division in round *t+1* is given by

$$p_{jk}(t+1) = \frac{\Pi_{jk}(t)}{\sum_{k=1}^{10} \Pi_{jk}(t)} \tag{5}$$

The game is then repeated a sufficient number of times so that the players can hopefully learn sufficiently to converge to the ideal distribution. The experiment can be repeated with other values of *Q*, *K*, *c* and *r*.

4 Hypothesis Testing, Results, and Discussion

For testing the assertion, we selected various values of *Q*, *K*, *c* and *r*. In the 2-player model equilibrium, we can see from the expressions of p_1 and *F(p)* that the drivers of interest are the values ($2K - Q$) and ($r - c$). If the combined capacity (2*K*) is barely more than the demand (*Q*), the sellers have little incentive to lower prices, while if there is a large amount of overcapacity, the sellers would greatly reduce prices (since losing would mean catering to a very small residual demand). The difference ($r - c$) on the other hand would determine the range of the support of

prices. We keep Q (the total quantity demanded) fixed at 100 units and r (the reservation price) fixed at \$80. The values of K chosen reflect the amount of overcapacity: at K=65, we have moderate overcapacity, prompting moderate competition; at K=80, the possibility of supplying only a small fraction of the demand (i.e. if the supplier bids the higher price, he ends up supplying the residual of only 20 units) should prompt more severe competition. For each of these values of K, we choose the value of c, the marginal cost, as \$60.

For each of the pairs of values of K and c, we calculate p_1 and corresponding subdivision limits, which are shown in the *Bin* column. After running the simulation as described above, we find out the number of times the prices are picked in each subdivision, and this is given in the Frequency (O_i) column. The theoretical cumulative distribution gives us the theoretically expected number of observations in each subdivision, and this is presented in the E_i column. We compute the χ^2 statistic as $\sum_{i=1}^{10} \frac{(O_i - E_i)^2}{E_i}$, and this is compared with the corresponding chi-square value with $p = 0.05$ (16.92). Formally stated, we would reject the null hypothesis that the data follows the distribution specified in (2), if the calculated χ^2 exceeded the corresponding χ^2 value with a significance level α of 0.05:

H₀: $F_n(p) = F_n^*(p)$ where $F_n^*(p)$ is the experimentally generated distribution.

H₁: $F_n(p) \neq F_n^*(p)$

The simulation results show that as expected from the theoretical results, lower prices are preferred over high prices (Tables 1 & 2). We run a chi-square goodness of fit test with each of the simulation settings. As the results show (the 'Chi-sq.' column in the tables compute the χ^2 statistic, whose sum is shown in the final row, and this sum is compared to the corresponding chi-square value with p = 0.05 which is shown in the final column), the fit with the theoretical distribution is always very good. In all the nine cases, we do not reject the null hypothesis that the experimental frequency distribution follows the theoretical probability distribution. In other words, the simulation runs results in the agents learning over time to come very close to the ideal distribution with every set of values of the parameters K and c.

It is therefore observed that the artificial software agents start off selecting their prices uniformly throughout the interval of (p_1, r), but gradually learn over time to select lower prices with monotonically higher probabilities. In fact, the final frequency distributions show that the learning is 'perfect' within margins of statistical error. The results have interesting ramifications in real-world scenarios. Managers might not have the luxury of learning over a large number of observations themselves as in these simulations, but they can utilize the "organizational memory" (i.e. the

experiences of him as well as his predecessors) to effectively build the learning capability over time. Managers also have their own intuition, which these artificial agents lack that might result in accelerated learning towards equilibrium (and therefore optimal) behavior.

5 Conclusions

While we are currently addressing many of these issues in our ongoing research, the results of our simulations with the 2-player model show considerable promise. We hope that these results spur the interest of using automated agents that will enable organizations to effectively compete in the increasing number of electronic transactions. While one of the main attractions of B2B exchanges remains in their ability to automate the processes by which organizations can participate in electronic transactions with each other, the problem of overseeing each and every one of them is still very much an issue. This problem will likely exacerbate in future as more and more organizations start to utilize these electronic services. While the algorithms that need to be used in real-world scenarios will be much more complex than those presented in this research, we think that organizations might over time develop such algorithms of increasing sophistication. At first, very basic transactions having routine processes would be entrusted to such learning mechanisms. As algorithms get more complex, and simultaneously organizations also gain confidence in such mechanisms, more complex transactions would probably be entrusted. Organizations might also develop processes by which unusual procedures set off triggers for either human intervention or even a complete abort.

Table 1. Simulation run results with $K = 65$ units, $c = \$60$

Bin	Frequency (O_i)	Th. Cum. Fr.	Th. Freq. dist. (E_i)	Chi-sq.	Chi-sq. value (p=0.05)
71.69230769	176	171.0526316	171	0.143093117	
72.61538462	152	317.0731707	146	0.244855636	
73.53846154	133	443.1818182	126	0.376585912	
74.46153846	109	553.1914894	110	0.009266784	
75.38461538	86	650	97	1.206752397	
76.30769231	78	735.8490566	86	0.717628032	
77.23076923	75	812.5	77	0.035558781	
78.15384615	67	881.3559322	69	0.050024511	
79.07692308	68	943.5483871	62	0.052534014	
80	56	1000	56	0.351041475	
	1000		1000	3.187340659	16.92

Table 2. Simulation run results with K = 80 units, c = $60

Bin	Frequency (O_i)	Th. Cum. Fr.	Th. Freq. dist. (E_i)	Chi-sq.	Chi-sq. value (p=0.05)
66.5	284	307.6923077	308	1.824307692	
68	197	500	192	0.114492308	
69.5	140	631.5789474	132	0.538947368	
71	99	727.2727273	96	0.114229904	
72.5	79	800	73	0.541022727	
74	61	857.1428571	57	0.260357143	
75.5	48	903.2258065	46	0.079749309	
77	36	941.1764706	38	0.100264137	
78.5	35	972.972973	32	0.322752385	
80	21	1000	27	1.344027027	
	1000		1000	5.24015	16.92

References

1. Allen, B., Deneckere, R., Faith, T. and Kovenock, D., "Capacity precommitment as a barrier to entry: a Bertrand-Edgeworth approach," *The North American Winter Meetings of the Econometric Society*, January 1992
2. Bandyopadhyay, S., Barron, J.M. and Chaturvedi, A.R., "A game-theoretic analysis of competition among sellers in an online exchange," Working Paper, 2002
3. Bell, A. M., "Reinforcement Learning Rules in a Repeated Game," *Computational Economics,* Vol. 18, pp. 89-111, 2001.
4. Carriero, N. and Gelernter, D., How to write parallel programs: A guide to the perplexed, ACM Computing Surveys, 21, 3 (September 1989), 323-357.
5. Chaturvedi, A and Mehta, S., The SEAS Environment 2, Technical Report, Purdue University, West Lafayette, IN, 2002.
6. Chaturvedi, A. R. and Mehta, S., "Simulations in Economics and Management: Using the SEAS Simulation Environment," Communications of the ACM, March 1999.
7. Epstein, J. M. and R. Axtell, Growing Artificial Societies: Social Science from the Bottom Up, Brookings Institution Press, Washington DC, 1996.
8. Erev, I. and Rapoport, A. "Coordination, 'Magic,' and Reinforcement Learning in a Market Entry Game," Games and Economic Behavior, Vol. 23, pp. 146-175, 1998.
9. Erev, I. and Roth, A.E., "Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria," *The American Economic Review*, Vol. 88 No. 4, pp. 848-881, 1998
10. Helper, S. and MacDuffie, J.P. "B2B and Modes of Exchange: Evolutionary and Transformative Effects," *The Global Internet Economy*, edited by Bruce Kogut (forthcoming)

11. Kerrigan, R., Roegner, E.V., Swinford, D.D. and Zawada, C.C. "B2Basics," *Mckinsey Quarterly* No. 1, 2001, pp. 45-53
12. Kreps, D. and Scheinkman, J. "Quantity precommitment and Bertrand competition yield Cournot outcomes", *The Bell Journal of Economics*, 1983, pp. 326-337
13. Kuhn, H.W. and Nasar, S. "Editor's Introduction to Chapters 5, 6 and 7", *The Essential John Nash*, pp. 48
14. Larson, P. "B2B E-Commerce: The Dawning of a Trillion-Dollar Industry", *The Motley Fool's Internet Report*, March 2000
15. Nash, J.: "Non-Cooperative Games" *Annals of Mathematics*, Vol. 54, September 1951, pp. 286-295
16. Oliver, J. "A Machine Learning Approach to Automated Negotiation and Prospects for Electronic Commerce," *Journal of Management Information Systems,* Vol. 13, no. 3, pp. 83-112, 1996.
17. Rapoport, A., Daniel, T. E. and D. A. Searle, "Reinforcement-Based Adaptive Learning in Asymmetric Two-Person Bargaining with Incomplete Information," *Experimental Economics,* Vol. 1, pp. 221-253, 1998.
18. Roth, A.E. and Erev, I. "Learning in Extensive-Form Games: Experimental Data and Simple Dynamic Models in the Intermediate Term," *Games and Economic Behavior*, Special Issue: Nobel Symposium, 8(1), pp. 164-212, 1995
19. Sutton, R. S. and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA 1998
20. Thorndike, E.L. "Animal Intelligence: An Experimental Study of the Associative Processes in Animals," *Psychological Monographs*, 2(8), 1898.

Rapid Real-Time Interdisciplinary Ocean Forecasting Using Adaptive Sampling and Adaptive Modeling and Legacy Codes: Component Encapsulation Using XML

Constantinos Evangelinos[1], Robert Chang[1], Pierre F.J. Lermusiaux[2], and Nicholas M. Patrikalakis[1]

[1] Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.
[2] Harvard University, Cambridge, MA 02138, U.S.A.

Abstract. We present the high level architecture of a real-time interdisciplinary ocean forecasting system that employs adaptive elements in both modeling and sampling. We also discuss an important issue that arises in creating an integrated, web-accessible framework for such a system out of existing stand-alone components: transparent support for handling legacy binaries. Such binaries, that are most common in scientific applications, expect a standard input stream, maybe some command line options, a set of input files and generate a set of output files as well as standard output and error streams. Legacy applications of this form are encapsulated using XML. We present a method that uses XML documents to describe the parameters for executing a binary.

1 Introduction

Effective ocean forecasting is essential for efficient human operations in the ocean. Application areas include among others fisheries management, pollution control and maritime and naval operations. The advances in physical oceanography numerical models and data assimilation (DA) schemes of the last decade have given rise to complete Ocean Prediction systems [1] that are used in operational settings. Recent developments in the availability of high-performance computing and networking infrastructure now make it possible to construct distributed computing systems that address computationally intensive problems in interdisciplinary oceanographic research, coupling physical and biological oceanography with ocean acoustics [2].

Poseidon [3] is such a distributed computing based project, that brings together advanced modeling, observation tools, and field and parameter estimation methods for oceanographic research. The project has three main goals:

1. To enable efficient interdisciplinary ocean forecasting, by coupling physical and biological oceanography with ocean acoustics in an operational distributed computing framework.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 375–384, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2. To introduce adaptive modeling and adaptive sampling of the ocean in the forecasting system, thereby creating a dynamic data-driven forecast.
3. To allow seamless access, analysis, and visualization of experimental and simulated forecast data, through a science-friendly Web interface that hides the complexity of the underlying distributed heterogeneous software and hardware resources. The aim is to allow the ocean scientist/forecaster to concentrate on the task at hand as opposed to the micro-management of the underlying forecasting mechanisms.

The Poseidon project employs the Harvard Ocean Prediction System (HOPS) [4] as its underlying advanced interdisciplinary forecast system. HOPS is a portable and generic system for interdisciplinary nowcasting and forecasting through simulations of the ocean. It provides a framework for obtaining, processing, and assimilating data in a dynamic forecast model capable of generating forecasts with 3D fields and error estimates. HOPS has been successfully applied to several diverse coastal and shelf regions [1], and analyses have indicated that accurate real-time operational forecast capabilities were achieved. HOPS' advanced DA scheme (Error Subspace Statistical Estimation - ESSE [5]) that is quasi-optimal and at the same time provides an estimate of the dominant uncertainty modes in the forecast is central to the project's stated goal of adaptive modeling and sampling.

The architecture of Poseidon is being designed based on HOPS, but keeping in mind the future HOPS developments and also very importantly allowing the replacement of certain elements of HOPS by other components, e.g. employing different physical oceanographic models for adaptive physical modeling. Moreover, the ESSE methodology, that is computing and data intensive, is also an important driving force behind the architectural design decisions.

One of the first practical problems we faced in the process of designing Poseidon had to deal with the fact that HOPS (as well as other ocean modeling systems, eg. for physical oceanography ROMS [6] or for ocean acoustics OASES [7]) are, like most scientific applications, legacy[1] programs. They consist of binaries that expect a standard input (*stdin*) stream, maybe some command line options, a set of input files and generate a set of output files as well as standard output (*stdout*) and error (*stderr*) streams. In such a setup, any workflows are either executed interactively (a very common approach) or (after all potential problems are handled) hardcoded in scripts. While such an approach, which dates from the days when graphical user interfaces (GUIs) were not available, is efficient for a skilled user, it is cumbersome and error-prone and entails a steep learning curve. Moreover it is not suited for remote use over the Web.

After examining various ways of dealing with this issue in the context of our distributed computing architecture and keeping in mind that the Poseidon system should allow for future handling of non-HOPS components, we opted to keep working with Fortran binaries and to encapsulate their functionality and

[1] The term "legacy" should not be misconstrued to imply aged code in this context: these are all codes with an active development community and recent enhancements. For various reasons they are still being written in Fortran 77.

requirements using the eXtensible Markup Language (XML) [8]. Thereby we are creating a computer-readable manual for the binaries, allowing us to generate a GUI, check for parameter correctness and drive execution in a transparent manner.

In what follows, Section 2 briefly describes ESSE-based forecast workflows, and adaptive sampling and modeling in the Poseidon system. Section 3 discusses the restrictions legacy software impose on our system design and some proposed solutions. Section 4 describes the XML schema [9] designed for constraining XML descriptions of encapsulated binaries. Section 5 presents some results from our initial XML-based implementation. Section 6 concludes the paper.

2 Forecast Workflow and Adaptivity

The initial implementation of Poseidon is based on HOPS. Poseidon distributes its forecasting and data assimilation components, and will include new HOPS developments in adaptive modeling and adaptive sampling. The HOPS forecasting software is distributed by separating/parallelizing its sequential modules. For example, physical and biological time-stepping modules are run simultaneously on different CPUs with data-exchanges between modules. In this section, the focus is on the data assimilation and adaptivity components of HOPS.

Data assimilation (DA) provides estimates of natural fields which are better than what can be obtained by using only observations or a dynamical model. Data and models are usually combined in accord with their respective uncertainties, by quantitative minimization of a cost function. These computations are very expensive. The ESSE schemes provide an optimal reduction of the problem: only the dominant errors are minimized. For example, if a variance criterion is used to combine data and dynamics, the "dominant errors" are then defined by the dominant ordered eigendecomposition of a normalized form of the error covariance matrix. Even with such optimal error reductions, ESSE still involves massive throughput requirements. However, by design, it provides tremendous opportunities for scalable parallelism within the Poseidon system.

The ESSE workflow is as follows: First, the interdisciplinary error subspace is initialized based on an error decomposition on multiple scales (Fig. 1, far left oval). These dominant initial errors are then evolved by an ensemble of perturbed nonlinear and stochastically-forced dynamical model integrations (Fig. 1, center left oval). As the size of the ensemble is increased, convergence criteria are evaluated. Usually, converged ensemble sizes are $O(100 - 1000)$. This provides a significant opportunity for throughput parallelism. Individual model integrations can also be parallel simulations (depending on the problem size and interdisciplinary nature of the problem), further increasing the scope for parallelism, while also stressing available computing and data resources.

Once error estimates have converged, adaptive sampling forecasts are issued (Fig. 1, bottom left oval). For example, future sampling patterns of autonomous underwater vehicles are computed so as to maximize the reduction of forecast errors. As new data are made available, data-forecast misfits are computed and

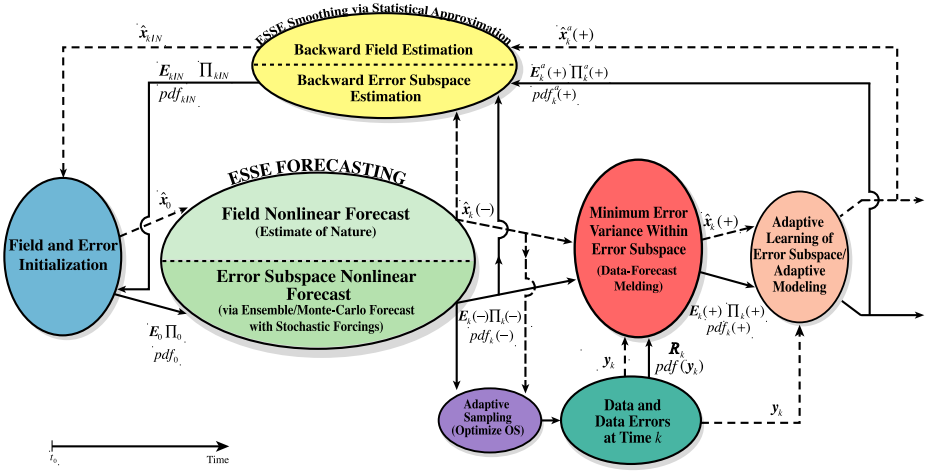


Fig. 1. The ESSE schematic workflow, adapted from [10]

used to correct the predicted fields by minimum error variance estimation in the interdisciplinary error subspace (Fig. 1, center right oval). Outputs are the filtering field and error estimates. A posteriori data misfits are then calculated and used for adaptation of the dominant errors and adaptation of model structures and parameters (Fig. 1, right oval). Ultimately, the smoothing via ESSE is carried out (Fig. 1, top oval) to correct, based on future data, the past coupled fields and uncertainties [10].

Automated objective adaptive modeling allows the optimal use of approximate models for rapidly evolving ocean dynamics. Presently, a model quantity is said to be adaptive if its formulation, classically assumed constant, is made variable as a function of data values. Both structural as well as parametric adaptation are possible. Physical adaptive modeling includes regime transition (e.g., well-mixed to stratified) and evolving turbulent mixing parameterizations. Biogeochemical adaptive modeling includes variations of biological assemblages with time and space (e.g., variable zooplankton dynamics, summer to fall phytoplankton populations, etc) and evolving biogeochemical rates and ratios. This is especially important because biogeochemical modeling is in its infancy and model uncertainties are very large. The adaptive component also greatly facilitates quantitative comparisons of competing biogeochemical models, thus ultimately leading to better scientific understanding.

From a computational point of view, three basic cases are considered in the Poseidon design: i) a single interdisciplinary model is run and adapted, ii) a single physical model is coupled to a set of competing biological models whose parameters are adapted, iii) competing interdisciplinary models are run and their parameters are adapted. In all cases, the basis of the adaptation is the value of the misfits between model estimates and data. When misfits are large, either models are adapted or, in the later two cases, eliminated when too inadequate.

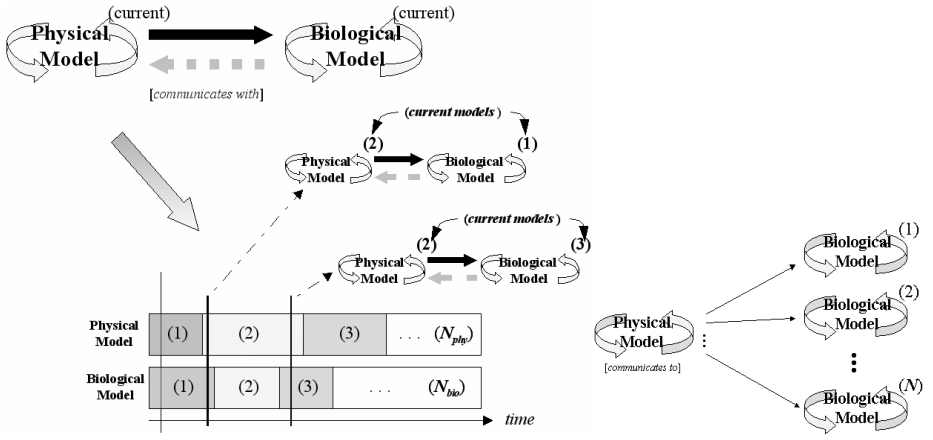


Fig. 2. Approaches for adaptive biophysical modeling

Within ESSE, misfits are large when they become significant compared to their expected values, the ESSE uncertainties.

In the first case (Fig. 2, left), specific model input files are continuously checked by the model software as it is running. These input files are updated when data-forecast misfits are significant. Model parameters and/or model structures are then modified accordingly. For example, the carbon-to-chlorophyll ratio is modified or the number of state variables is increased (adding mesozooplankton). The later is a structural adaptation which is efficiently implemented using a set of C function pointers, with little change to the legacy software. Adaptive updates on the physics or biology sides do not have to be concurrent. In the second case (Fig. 2, right), it is the forecasts of competing biogeochemical models which are compared based on their respective data-forecast misfits. The parameter values are updated and the best biological functional forms can then be selected. The third case is as the second except that several physical as well as several biological models are run and compared. In both cases ii) and iii), only the parameters of the competing model structures are adapted (i.e. fitted) to the data. The best model structures correspond to the fitted model which has the least significant misfits.

3 Legacy Codes

The software components used in Poseidon are legacy Fortran 77 codes (be it HOPS or ocean acoustics codes). As such they do not fit well within a modern distributed computing model: For example, there is no support for dynamic memory allocation, remote procedure calls and component-based programming. At worst, they require manual intervention to work with each other (conversion, concatenation, file editing). At best, they have been designed to interoperate in a prescribed workflow. The individual components of such a workflow are the

legacy binaries and the connections between them are files and/or streaming input/output. This type of setup is common among scientific codes that have evolved over many years.

There are two major options for dealing with this issue of legacy: Migration and Encapsulation (wrapping). The cleanest approach is to migrate the code and rewrite all applications in a modern language, employing object-oriented and component technologies. Use of platform-agnostic programs (by transferring the codes to Java for example) and Mobile Agents [11] for the distributed calculations would be a radical form of such modernization. Such an option is obviously costly. Moreover, it is error-prone as it is impractical to convert existing procedural programs to object-oriented components [12].

The more traditional approach is to encapsulate legacy codes as modern software components using wrappers. For example, CORBA [13] wrappers (for clear standardization reasons) can be used. The code can also be wrapped in C/C++ and then called from Java (in a more complicated Mobile Agent setting for distributed computing) using JNI [14]. However this powerful encapsulation approach involves breaking the Fortran code into separately callable components that a main program can access.

While CORBA was not designed with high performance computations in mind, there is recent activity in the area of software components for scientific computation, centered mostly around the Common Component Architecture Forum (CCA) [15]. We chose not to follow this approach as it would still entail significant changes to our software components; moreover it would have to force their developers to migrate to a new paradigm to allow us to keep up with enhancements and bug fixes to their codes. While this is possible (but not necessarily practical) for in-house codes, it is not reasonable for third party codes and impossible with programs that are provided in binary format.

Another approach compatible with stand-alone binaries involves wrapping at the binary level, preparing the input files and arguments to the binary, executing the program and capturing and interpreting the output. This can again be accomplished using CORBA or Mobile Agents (losing much of their advantages in the process). It can also be done via a master application generating job scripts for a Grid computing [16] environment.

This way working with legacy codes reduces to devising an extensible encapsulation of the software components (as binaries), that treats them as black boxes with a set of inputs/outputs and a set of valid types and ranges of runtime parameters. The advent of XML [8] provides a standards-based way to accomplish this. XML describes data through the use of custom tags thus eliminating the need to conform to a specific programming structure and offering the possibility to integrate legacy software with new technology.

Some related work has been done but none focused on wrapping legacy software components at the binary level to allow users to modify their runtime parameter values. Sneed [17] worked on techniques for encapsulating legacy COBOL programs with an XML interface. This requires some modifications of legacy software within an architecture so as to adapt the components for

reading and writing XML interfaces. Walker et al [18,19] illustrate the software architecture of a problem solving environment (PSE) used for the construction of scientific applications from software components. Users visually construct domain-specific applications using a Java/CORBA-based PSE for scientific simulations and computations. Each legacy component is encapsulated as a CORBA object, with its interface and constraints defined in XML.

The main difference between our approach and approaches such as the above is that we aim to encapsulate software components without having to adapt them for XML. We are in fact generating software metadata (in XML format) that describe completely how to control the legacy code's runtime behavior.

4 XML Schema Design

The XML interface to the binaries should be self-contained and should not require modifications to the binaries. By providing a detailed XML description for a binary, this binary is treated as a black-box. An application should then be able to parse in the XML description, and from its contents, determine the specifics on how to properly execute the binary with the appropriate input parameters.

Several key concerns have to be addressed and supported in the XML schema [20]. One issue is that the resulting XML documents should provide as much information to the user as possible, so that well informed decisions can be taken while making changes to the runtime parameters. There should be a set of default parameter values so that manual entry of all values for each execution is avoided, especially since there can be hundreds of them.The XML descriptions conforming to our schema design should also be capable of specifying input and output files for the binary execution. Since runtime parameters should be checked for legality after any user changes, the schema must support datatypes and constraints on parameter values. This is to facilitate the execution of the binary and ensure that all input parameters are acceptable. Each parameter value can be validated against its constraints and datatype before proceeding.

Our schema design supports the description of the binary's input and output files, as well as the runtime input parameters that are read in from *stdin*. It also handles command line arguments and other runtime parameter sources. It consists of two layers: the top layer handles the basic information about the binary, child layers include the runtime inputs. Most elements in the schema have parameters for name and description. These parameters are very useful for generating a GUI that provides sufficient information for the users. A more detailed description of the schema with example XML descriptions of software components will be presented in a forthcoming paper.

5 Initial Results

We initially tested our implementation with the Primitive Equation (PE) Model binary in the HOPS system. The PE model is at the heart of the forecasting system: it is the solver for the ocean temperature, salinity and velocity fields.

The PE Model binary has runtime parameters read in from *stdin*. A sample of the runtime parameter data is provided in the *stdin* file *pemodel.in*. We wrote a schema-conforming XML description based on the values and types of these parameters.

After validating the XML description using the schemas, our application was able to process the parameters for display by the GUI, as shown in Fig. 3. The system presents the contents of the *stdin* file in an organized manner that is easily understood. Instead of editing the input file directly, the user can update parameter values in the GUI and have changes checked for validity (type, range) before the system generates the new input file and execution script automatically.

A subset of the PE Model *stdin* file generated by our system is included in Fig. 4. The entire PE Model input file consists of 39 *cards*, which are analogous to the *sets* defined in our schema. Each *card* contains the group of variables relating to a specific aspect of the PE Model binary. For instance, *Card 10* deals with the tidal mixing variables used during execution. The PE Model binary ignores every other line that contains the textual description of the parameter value(s) for each *card*. The last line beginning with "99" signifies the end of the input file for the binary.

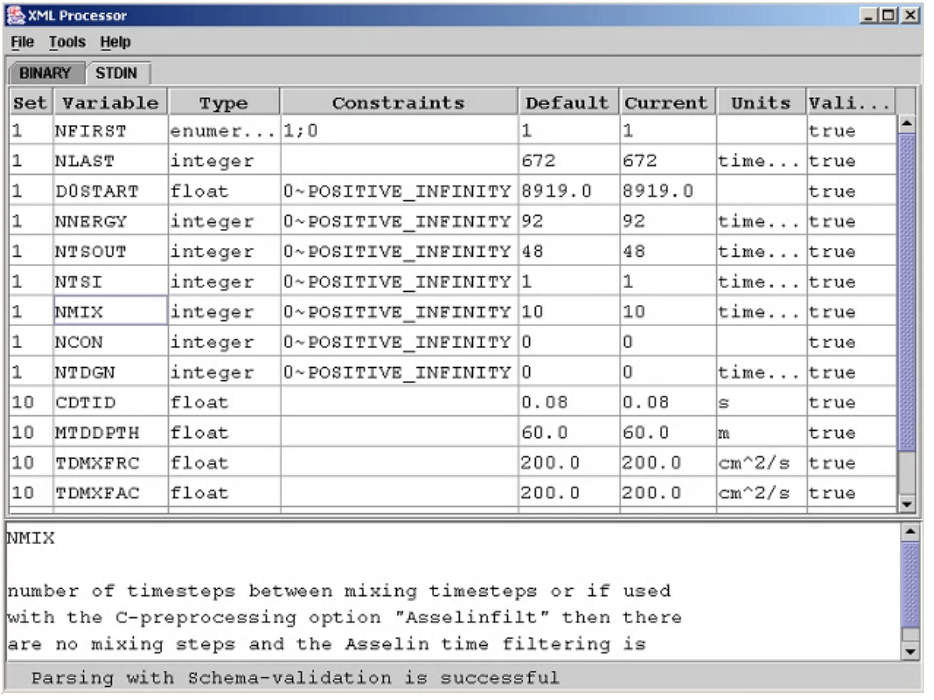


Fig. 3. Screen capture of GUI displaying runtime parameters

```
*** CARD 1:  Various Initialization Parameters
1  672  8919.00  92  48  1  10  0  0
*** CARD 10:  Tidal Mixing
0.08  60.0  200.0  200.0  0.1
*** CARD 39:  Input File Defining Tidal Regions
dev/null
99   END of input data
```

Fig. 4. Subset of *stdin* file generated automatically by our system

6 Conclusion and Future Research

Portions of the high level architecture for Poseidon, a distributed computing system for Real-Time Interdisciplinary Ocean Forecasting employing adaptive modeling and sampling were presented. This includes ESSE-based workflows and adaptive modeling and sampling schemes used to enhance forecasting capabilities. Poseidon efficiently manages legacy codes by using an XML-based method for the encapsulation of legacy binaries. The schema-validated XML descriptions provide a machine (and human) readable standard for handling legacy codes. By wrapping the binaries using XML, their input and output files and *stdin*-provided runtime parameters are described. A prototype system was implemented and shown to generate a graphical interface displaying user-prescribed parameters. The GUI allows for user customization of parameters and validates user changes. It produces input files and a script file for the execution of the binaries.

Future research includes extending our approach to encompass build-time information. To allow XML-descriptions of binaries from other systems, the list of supported datatypes needs to be expanded. The full system will be integrated within a Grid infrastructure using the Globus toolkit [21].

Acknowledgements. The authors would like to acknowledge the assistance of Professors J.J. McCarthy, A.R. Robinson and H. Schmidt, Drs. P.J. Haley, S. Lalis and R. Tian and Mr. S.K. Geiger. This work was funded in part from NSF/ITR (under grant EIA-0121263) and from DoC (NOAA via MIT Sea Grant) (under grant NA86RG0074).

References

1. Robinson, A.: Forecasting and simulating coastal ocean processes and variabilities with the Harvard Ocean Prediction System. In Mooers, C., ed.: Coastal Ocean Prediction. AGU Coastal and Estuarine Studies Series. American Geophysical Union (1999) 77–100
2. Patrikalakis, N.M., Abrams, S.L., Bellingham, J.G., Cho, W., Mihanetzis, K.P., Robinson, A.R., Schmidt, H., Wariyapola, P.C.H.: The digital ocean. In: Proceedings of Computer Graphics International, GCI '2000, Geneva, Switzerland, IEEE Computer Society Press (2000) 45–53 Los Alamitos, CA: IEEE, 2000.

3. Patrikalakis, N.: (Poseidon: A Distributed Information System for Ocean Processes) http://czms.mit.edu/poseidon/.
4. Robinson, A.: (Harvard Ocean Prediction System (HOPS)) http://oceans.deas.harvard.edu/HOPS/HOPS.html.
5. Lermusiaux, P., Robinson, A.: Data assimilation via Error Subspace Statistical Estimation. Part I: Theory and schemes. Month. Weather Rev. **127** (1999) 1385–1407
6. Haidvogel, D., Arango, H., Hedstrom, K., Malanotte-Rizzoli, A.B.P., Shchepetkin, A.: Model evaluation experiments in the North Atlantic basin: Simulations in nonlinear terrain-following coordinates. Dyn. Atmos. Oceans **32** (2000) 239–281
7. Schmidt, H., Tango, G.: Efficient global matrix approach to the computation of synthetic seismograms. Geophys. J. R. Astr. Soc. **84** (1986)
8. : (eXtensible Markup Language) http://www.w3.org/XML.
9. Roy, J., Ramanujan, A.: XML schema language: Taking XML to the next level. IT Professional **3** (2001) 37–40
10. Lermusiaux, P., Robinson, A., Haley, P., Leslie, W.: Advanced interdisciplinary data assimilation: Filtering and smoothing via Error Subspace Statistical Estimation. In: The OCEANS 2002 MTS/IEEE, Holland Publications (2002) 795–802
11. Houstis, C., Lalis, S., Christophides, V., Plexousakis, D., Vavalis, E., Pitikakis, M., Kritikos, K., Smardas, A., Gikas, C.: A service infrastructure for e-Science: the case of the ARION system. In: 14th Intern. Conf. on Advanced Information Systems Engineering (CAiSE 2002). Number 2512 in Lecture Notes in Computer Science, Toronto, Canada, Springer (2002) 175–187 E-Services and the Semantic Web workshop (WES2002).
12. Terekhov, A., Verhoef, C.: The realities of language conversions. IEEE Software (2000) 111–124
13. : (Common Object Request Broker Architecture) http://www.corba.org.
14. : (Java Native Interface) http://java.sun.com/j2se/1.4.1/docs/guide/jni.
15. : (Common Component Architecture Forum) http://www.cca-forum.org.
16. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
17. Sneed, H.M.: Wrapping legacy COBOL programs behind an XML-interface. In: Proceedings of the Eighth Working Conference on Reverse Engineering. (2001) 189–197
18. Walker, D.W., Li, M., Rana, O.F.: An XML-based component model for wrapping legacy codes as Java/CORBA components. In: Proc. of the Fourth Intern. Conf. on High Performance Computing in the Asia-Pacific Region, Beijing, China, IEEE Computer Society Press (2000) 507–512
19. Walker, D.W., Li, M., Rana, O.F., Shields, M.S., Huang, Y.: The software architecture of a distributed problem-solving environment. Concurrency: Practice and Experience **12** (2000) 1455–1480
20. : (XML Schema Specification) http://www.w3.org/XML/Schema.
21. : (The Globus Project) http://www.globus.org.

A Computational Infrastructure for Reliable Computer Simulations

J. Tinsley Oden[1], James C. Browne[1], Ivo Babuška[1], Kenneth M. Liechti[2], and Leszek F. Demkowicz[1]

[1] Institute for Computational Engineering and Sciences
The University of Texas at Austin
Austin, TX 78712, U.S.A.
{oden, babuska, leszek}@ticam.utexas.edu
browne@cs.utexas.edu
http://www.ticam.utexas.edu
[2] Research Center for the Mechanics of Solids, Structures and Materials
The University of Texas at Austin
Austin, TX 78712, U.S.A.
kml@mail.utexas.edu

Abstract. The reliability of computer simulations of physical events or of engineering systems has emerged as the single most critical issue facing advancements in computational engineering and science. Without concrete and quantifiable measures of reliability, the confidence and usefulness of computer predictions are severely limited and the value of computer simulation, in general, is greatly diminished. This paper describes a mathematical and computational infrastructure for the systematic validation of computer simulations of complex physical systems and presents procedures for the integration of computer-based verification and validation processes into simulations. A general class of applications characterized by variational boundary-value problems in continuum mechanics is considered, but the approach is valid for virtually all types of models used in simulations. To simulate a particular feature or attribute of a physical event, four basic concepts are used: 1) hierarchical modeling and *a posteriori* estimation of modeling error, 2) *a posteriori* error estimation of approximation error, 3) quantification of uncertainty in the data and in the predicted response, and 4) the development of dynamic a data management paradigm, based on code composition of associated code interfaces and use of annotated source code. The result is a computational framework for computing bounds on, and estimating accuracy of, computer predictions of user-specified features of the response of physical systems.

1 Introduction

The systematic study of reliability of computer simulations of physical phenomena is generally divided into two basic processes,Verification and Validation (V and V). As a scientific discipline, V and V is in its very early stages of development. Verification is the process of determining the accuracy with which a

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 385–390, 2003.
© Springer-Verlag Berlin Heidelberg 2003

given mathematical model of physics is solved; validation is the process of determining that the mathematical model represents the actual physical system with sufficient accuracy. Here verification does not include software engineering issues of the logical correctness of the program which implements the simulations (this is so-called *code* verification). Verification is thus concerned with estimating and controling numerical approximation error. In recent years, significant progress has been made in this area (see Ainsworth and Oden [1], Babuska and Strouboulis [3], Oden and Prudhomme [17], Becker and Rannacher [9]).

Validation, until recently, has been primarily an experimental process in which data from physical observations are compared with computer simulations. The notion of hierarchical modeling provides a mathematical structure that can be useful in directing validation studies. In this structure, a class of models of events of interest is defined in which a "fine" model is identified which possess a level of sophistication believed to be high enough to adequately capture the event of interest with good accuracy. This model, however, may be intractable. The complexity of this model, may be such that one cannot solve it, even computationally. We then identify (families of) coarse models which are solvable. Using the fine model as a datum, the error in the solution of coarse models can be estimated and controlled, with the goal of obtaining a model best suited for the simulation goal at hand.

Physical data must be collected to characterize the fine model, and this must be furnished by data generated in actual laboratory tests. In general, these data will exhibit scatter (uncertainty) – and the basic fine-model data are then random functions. Thus the validation process is complicated by the fact that the models themselves are stochastic.

In the sections to follow, we outline briefly the program components that need to be integrated to produce a V and V computational system.

2 Problem Definition

As a representative physical situation, we consider a heterogeneous solid body D occupying a region in $\mathbb{R}^3$ and heated along its boundary ∂D and subject to an internal heat source $b - \sigma u$, u being the temperature field, b the heat supplied per unit volume and $\sigma > 0$ a reaction coefficient. The fine mathematical model is characterized by the equations,

$$\begin{aligned} -\mathrm{div} A\boldsymbol{\nabla} u + \sigma u &= b \quad \text{in } D \\ \mathbf{n} \cdot A\boldsymbol{\nabla} u &= g \quad \text{on } \partial D \end{aligned} \tag{1}$$

g being a prescribed heat flux and $\mathbf{n}$ a unit exterior normal. While the temperature field u is governed by this system, we are specifically interested in predicting certain features of the event, for instance, the temperature or the heat flux at a point $\mathbf{x_0}$ in direction $\boldsymbol{\nu}$ interior to D. These *quantities of interest* are functionals on u; e.g.

$$Q(u) = u(\mathbf{x_0}) \quad \text{or} \quad Q(u) = \boldsymbol{\nu} \cdot \mathbf{A}(\mathbf{x_0})\boldsymbol{\nabla} u(\mathbf{x_0}) \tag{2}$$

A weak form of this problem is

$$\begin{aligned} &\text{Find } u \in V \text{ such that} \\ &\qquad B(u,v) = F(v) \qquad \forall\, v \in V \end{aligned} \tag{3}$$

where V is an appropriate space of admissible functions, and

$$\begin{aligned} B(u,v) &= \int_D (A\nabla u \cdot \nabla v + \sigma uv)\, dx \\ F(v) &= \int_D bv\, dx + \int_{\partial D} gv\, ds \end{aligned} \tag{4}$$

Problem (2) (or, equivalently, (1)) is, in general, intractable. Typical complications that make it impossible or infeasible are:

1) the coefficient (thermal conductivity) A is a highly oscillatory function characterizing rapid oscillatory variations of mechanical properties at scales much smaller than the characteristic dimensions of D, and

2) the laboratory data needed to characterize A exhibit much scatter and, therefore, A is a random function (the data σ, b, and g may also be random). This uncertainty in the data must be quantified in some sense. Thus, it is desirable to develop statistical information sufficient to determine the probability density function (PDF) p for A, the variances, covariance, etc.

If p is the PDF, the mean or expected value of the random function f is $E[f] = \int_{-\infty}^{\infty} pf d\omega$. The solution u of (1) is, in this case, a random field, $u = u(\mathbf{x}, \omega)$, $\mathbf{x} \in D$ and $\omega \in \Omega$, Ω being an appropriate set of elementary events. Then, instead of (4), we use

$$\bar{B}(u,v) = \int_D E[A\nabla u \cdot \nabla v + \sigma uv]\, dx \tag{5}$$

$u, v \in V \times W$, W being a probability measure space.

3 Coarse and Discrete Models

Consistent with the theory of hierarchical modeling, we replace (2) by a simplified or coarse model,

$$u_0 \in V: \quad B_0(u_0, v) = F(v) \quad \forall\, v \in V \tag{6}$$

where

$$B_0(u_0, v) = \int_D (A_0 \nabla u_0 \cdot \nabla v + \sigma u_0 v)\, dx \tag{7}$$

Here A_0 is an effective thermal conductivity of the material obtained through homogenization methods and, for the purpose of this example, is assumed to be deterministic. The coarse model prediction of the quantity (or quantities) of interest is $Q(u_0)$.

In general, problem (7) is also intractable, but it can be solved approximately using numerical techniques such as finite elements. Thus, let V^h be a member

of a family of finite dimensional subspaces of V. We seek an approximation u_0^h of u_0 satisfying

$$u_0^h \in V^h : \quad B_0(u_0^h, v) = F(v) \quad \forall\, v \in V^h \tag{8}$$

Now, summing up our sources of error, we see that the error on the quantity of interest $Q(u)$ is,

$$\begin{aligned} Q(u) - Q(u_0^h) &= \underbrace{Q(u) - Q(u_0)}_{modeling\ error} + \underbrace{Q(u_0) - Q(u_0^h)}_{approaximation\ error} \\ &= \boldsymbol{\varepsilon}_{mod} + \boldsymbol{\varepsilon}_{approx} \end{aligned} \tag{9}$$

Estimating and controling the error component $\boldsymbol{\varepsilon}_{mod}$ is regarded as validation process (or as an aide to validation considerations), while controling $\boldsymbol{\varepsilon}_{approx}$ is a verification process. A general theory for controling this type of modeling error in quantities of interest Q through *a posteriori* error estimation and adaptive modeling has been developed by Oden and Prudhomme [19,17], and Oden and Vemaganti [15,16]. Techniques for deriving *a posteriori* error estimates for $\boldsymbol{\varepsilon}_{approx}$ and goal-oriented adaptive meshing have been advanced by Babuska and Strouboulis [3], Oden and Prudhomme [18,19], Rannacher, Becker and others [9]. Thus we assume that using the data available, it is possible to obtain computable lower and upper bounds on these components:

$$\boldsymbol{\gamma}_{mod}^{low} \leq \boldsymbol{\varepsilon}_{mod} \leq \boldsymbol{\gamma}_{mod}^{upp}, \quad \boldsymbol{\gamma}_{appox}^{low} \leq \boldsymbol{\varepsilon}_{approx} \leq \boldsymbol{\gamma}_{approx}^{upp} \tag{10}$$

Averages of these bounds can yield estimates of the errors (e.g. $\boldsymbol{\varepsilon}_{mod} \approx \frac{1}{2}(\boldsymbol{\gamma}_{mod}^{low} + \boldsymbol{\gamma}_{mod}^{upp})$). If these estimated errors exceed preset tolerances, the errors must be reduced by adaptive meshing (for $\boldsymbol{\varepsilon}_{approx}$) (see [17]) and adaptive modeling (for $\boldsymbol{\varepsilon}_{mod}$).

Another source of error relevant to validation is due to the randomness of the coefficients A. There are several approaches available to quantify this error. One direct approach is provided by the perturbation method described by Kleiber and Hien [14], which, for a first order approximation, takes the form

$$A(\mathbf{x}, \omega) = \bar{A}(\mathbf{x}) + \varepsilon A_1(\mathbf{x}, \omega), \quad u(\mathbf{x}, \omega) = \bar{u}(\mathbf{x}) + \varepsilon u_1(\mathbf{x}, \omega) \tag{11}$$

$\bar{A}$ and $\bar{u}$ being mean values of A and u. Then one treats first the deterministic fine problem (1) with the forms (4), involving $\bar{A}$ instead of A and $\bar{u}$ instead of u. Then $Q(u) = Q(\bar{u}) + \varepsilon Q(u_1)$. The random field $Q(u_1)$ is completely characterized in a stochastic sense, once the probability field of $A(\mathbf{x}, \omega)$ is provided and deterministic component $\bar{u}$ is known. Bounds on the truncation error inherent in the perturbation (11) can be calculated and used as additional measure of the modeling error [4]. The second approach is based on the theory of stochastic functions characterized by Karhunen-Loève expansion (see [5,11,6]). In [7], the problem of determination of the Karhunen-Loève expansion from experimental data is described. A brief survey [8] of the mathematics of verification and validation is presented when the experimental data [2] are used.

4 The Integration of Computational Components

Separate codes are written or are available for the component error estimations and adaptivity. The integration of these modules requires a major new dynamic data structure innovation to coordinate a component-oriented code development process, and requires a dynamic scaling of common data structures. An integrated compiler and runtime system is being developed to compose appropriate components and map data structures across interfaces (see [10,12,13]). Annotations of source code is to be used to guide compilation and enable interfacing different data structures. Different computational models provide for the specification of associative interfaces and the annotation language. The essential integration components are as follows:

1) Experimental data are collected to fully characterize the fine model, including statistics to give bounds on the data or PDF's and covariance matrices.

2) Quantities (or a quantity) Q of interest are (is) specified as the target physical entity to be predicted in the simulation (perhaps in the form of probability of the predicted values of the quantity).

3) The coarsest model is used to extract a preliminary estimate of Q and modeling and approximation errors are computed.

4) If the estimated error exceeds the prescribed tolerance, the model is enhanced and the calculation is repeated, each time adapting the computational model to reduce ε_{approx} until a model yielding results within the preset bounds is obtained.

5) The truncation error of the perturbation expansion is estimated; if the total error exceeds a preset tolerance, the data set and the fine model definition must be updated. If not, the predicted Q and the probability that it will takes on values in a given interval are produced as output.

In the absence of sufficient data to adequately characterize the random data fields, a worse-case analysis can be done giving upper and lower bounds on the predicted values.

The implementation of this and related algorithms is underway and will be the subject of forthcoming reports.

Acknowledgment. The assistance of Dr. Yusheng Feng and Dr. Fabio Nobile on this project is gratefully acknowledged. This work is done under ITR NSF grant 0205181. The support and encouragement of Dr. Frederica Darema of NSF is greatly appreciated by the authors.

References

1. Ainsworth, M., Oden, J.T.: A Posteriori Error Estimation in Finite Element Analysis. John Wiley & Sons, New York, (2000)
2. Babuška, I., Jerina, K., Li, Y., Smith, P.: Quantitative Assessment of the Accuracy of Constitutive Laws for Plasticity with Emphasis on Cyclic Deformation. Material Parameter Estimation for Modern Constitutive Equations. MD-Vol.**43**/AMD-Vol.**168** ASME (1993)

3. Babuška, I., Strouboulis, T.: Finite Element Method and its Reliability. Oxford Univ. Press (2001)
4. Babuška, I., Chatzipantelidis, P.: On Solving Elliptic Stochastic Partial Differential Equations. Comput. Methods Appl. Mech. Eng. **191** (2002) 4093-4122
5. Babuška, I., Tempone, R., Zouraris, G.E.: Galerkin Finite Element Approximations of Stochastic Elliptic Differential Equations. TICAM Report 02-38, The University of Texas at Austin, (2002)
6. Babuška, I., Liu, K-M.: On Solving Stochastic Initial-Value Differential Equations. Math. Models Methods Appl. Sci., (2003) In press
7. Babuška, I., Liu, K-M., Tempone, R.: Solving Stochastic Partial Differential Equations Based on Experimental Data. Math. Models Methods Appl. Sci., (2003) In press
8. Babuška, I.: Mathematics of the Verification and Validation in Computational Engineering. In: Proceeding of Int. Conf. on Mathematical and Computer Medeling in Sci. Eng., Prague Czech Republic (2003)
9. Becker, R., Rannacher, R.: An optimal control approach to *a posteriori* error estimation in finite element methods. In: Isereles, A. (ed.): Acta Numerica, Vol. 10. Cambridge Univ. Press (2001) 1-120
10. Browne, J.C., Berger, E., Dube, A,: Compositional Development of Performance Models in POEMS. Int. J. High Perf. Comput. Appl. **14**(4) (2000) 283–291
11. Deb, M.K., Babuška, I., Oden, J.T.: Solution of Stochastic Partial Differential Equations Using Galerkin Finite Element Techniques. Comput. Methods Appl. Mech. Eng. **190** (2001) 6359-6372
12. Edwards, C, Browne, J.C.: Scalable Dynamic Distributed Array and its Application to a Parallel hp-adaptive Finite Element Code. Proceedings of the 1996 POOMA Workshop, Santa Fe, New Mexico, (1996)
13. John, A., Browne, J.C.: Compilation of Constraints Programs with Noncyclic and Cyclic Dependencies to Procedural Parallel Programs, Int. J. Parallel Prog. **26**(1) (1998) 65-119
14. Kleiber, M. Hien, T.D.: The Stochastic Finite Element Method: Basic Perturbation Technique and Computer Implementation. John Wiley & Sons, (1992)
15. Oden, J.T., Vemaganti, K.: Adaptive Hierarchical Modeling of Heterogeneous Structures. Predictability: Quantifying Uncertainty in Models of Complex Phenomena. Phys. D., **133** (1999) 404-415
16. Oden, J.T., Vemaganti, K.: Estimation of Local Modeling Error and Goal-Oriented Modeling of Heterogeneous Materials; Part I: Error Estimates and Adaptive Algorithms. J. Comput. Phys., **164** (2000) 22-47
17. Oden J.T., Prudhomme, S.: Goal-Oriented Error Estimation and Adaptivity for the Finite Element Method. Comput. Math. Appl., **41**(5-6) (2001) 735-756
18. Oden, J.T., Prudhomme, S.: Estimation of Modeling Error in Computational Mechanics. J. Computat. Phys., **182** (2002) 496-515
19. Prudhomme, S., Oden, J.T.: On Goal-Oriented Error Estimation for Elliptic Problems: Applications to the Control of Pointwise Errors. Comput. Methods Appl. Mech. Eng., **176** (1999) 313-331

Automating Metadata Web Service Deployment for Problem Solving Environments

Ozgur Balsoy[1], Ying Jin[2], Galip Aydin[2], Marlon Pierce[2], and Geoffrey Fox[2]

[1]Computer Science Department, Florida State University, Tallahassee, FL 32306
Ozgur@csit.fsu.edu
[2]Community Grids Lab, Indiana University, 501 N. Morton Street, Suite 224, Bloomington, IN 47404-3730
{Yinjin, Gaydin, Marpierc, Gcf}@indiana.edu

Abstract. XML-based metadata information services are a crucial core service needed by Problem Solving Environments built over emerging service-based, globally-scaled distributed systems, as envisioned by the Open Grid Services Architecture and the Semantic Web. Developing user interfaces and services bindings for manipulating instances of particular schemas is thus extremely important and needs to be made as simple as possible. In this paper we describe procedures for automating the creation of Web Service environments that can be used to simplify the creation and deployment of schema-based metadata services.

1 Introduction

One of the most important underpinnings of Problem Solving Environments (PSEs) is the management of diverse types of information. Grid computing and distributed Web services are metadata rich: all entities should be described by XML metadata. Metadata must be used to describe the various parts that must be managed by the PSE, such as available applications, service interfaces and their binding points, batch queuing systems, hardware profiles, and so on. Many projects have attempted to define metadata pieces needed to build service grids, and such pieces will be needed as grid service data in the Open Grid Service Architecture [1,2]. In addition to the service metadata, there is also need to provide organizational metadata descriptors for the services themselves, and we have suggested data models for describing applications [3,4] and their associated service bindings. We thus anticipate that metadata richness will only increase.

We may place these issues into larger contexts. *Autonomic computing* (see [5] for a description and [6,7] for sample academic projects) seeks to define the interaction of components in next generation computing at all scales. In the autonomic computing vision, components mean everything from the physical component parts of individual computers to the service components of globally-scaled service environments. The ideal autonomic system is self-aware, self-monitoring, adaptive, and self-healing, among other characteristics.

Developing standards for Web services for computational grids is a major (and somewhat independent) component of the autonomic computing vision. Web services

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 393–402, 2003.
© Springer-Verlag Berlin Heidelberg 2003

are somewhat loosely defined, but may be characterized in general as using existing Web technologies and standards (such as HTTP and XML) to build the distributed computing environments. Important Web service standards include the Web Service Description Language (WSDL) [8] and the Simple Object Access Protocol (SOAP) [9]. The Open Grid Service Architecture, developed under the auspices of the Global Grid Forum and primarily led by IBM and the Globus team, seeks to extend the capabilities of Web services in order to support more sophisticated services. The key to this development is the extension of WSDL portTypes to include metadata and service state information.

In a major alternative effort, other groups are using various XML standards of the World Wide Web Consortium define what it terms the Semantic Web [10]. The Semantic Web vision borrows ideas from federated information systems and artificial intelligent, modified and simplified to apply to the much more ambitious problem of intelligent information retrieval and service execution on a global scale. The cornerstones of the Semantic Web are metadata descriptions (RDF) [11] and ontologies that provide semantic meaning.

XML-based information services are the key to both the OGSA and the Semantic Grid. We may go a step further and claim that the there is a general need for many informal, lightweight information management systems for particular schemas. We have developed such a system, described in Refs [12] and [13], which provides several per-schema implementations: newsgroups, citation and reference managers, training and conference registration systems. All such systems possess the same general capabilities: browsing, search, "push" and "pull" style event delivery options, and access controls on information and publishing. More sophisticated systems that support schema-spanning searches (with the associated meta-ontology research problems [14,15]) may be built on top of this practical base.

Given the demonstrable importance and proliferation of XML metadata, it becomes necessary to automate the creation of publication and editing tools for particular schemas. We describe the details of two such systems, which we refer to as "Schema Wizards". The first is designed to simplify the creation of XML wizards for specific schemas. These wizards are in turn used for creating and publishing instances in the XML messaging system described in [13]. The Schema Wizard is thus a wizard for wizards. The second wizard system, a follow-on to the first, is designed to automate metadata-based Web services (such as service data associated with Grid Web services). In both cases, a particular schema is used as the underlying data model, and the wizard generates the user interface (view) and action bindings (control) necessary to create and deploy the new metadata-based service.

2 Schema Wizard: Automating Metadata Creation for Publication

Applications that guide users through a complicated task are usually called wizards. XML wizards are applications that receive user inputs to generate XML documents. In our research on XML information frameworks (see particularly Ref [13]), XML is used as the medium for storing information, but the systems are user driven. Wizards help users generate schema-based XML content and publish into an XML messaging broker. Such documents are then typically published to various sources, including both persistent repositories and listeners demanding immediate notification.

In our current implementation, our information management system provides general purpose tools that can be used to create specific applications that are tied to single, specific schema. Thus to add a new application, one of the necessary tasks is to create a new publishing wizard to the particular schema. Creating such publishing wizards requires the following steps:

1. Writing an XML Schema which has all the necessary elements regarding with the current project.
2. Producing the Object Models of the XML Schema elements and attributes. This is done by using Castor's [16] Source Generator, which creates Java beans to correspond to the elements of the schema.
3. Obtaining user data. This step requires us to prepare HTML forms which have form elements to represent the schema elements. We use JavaServer Pages (JSP) technology, which allows us to interact with HTML forms and data objects at the same time. In that way we immediately assign the user entries to the data objects.
4. Generating the XML document. Using Castor's marshalling framework we create an XML document which contains the user inputs.
5. Publishing this XML document. This may use publish/subscribe systems such as the JavaMessaging System (JMS).

Given the repetitive, tedious nature of this process, we have developed a general purpose tool that automates the creation of this process. We refer to this tool as the SchemaWizard.

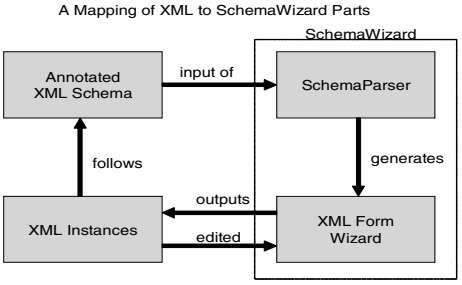


Fig. 1 The Schema Wizard is used to generate interfaces for particular XML schemas, which in turn generate XML instances for messaging.

2.1 Schema Wizard Architecture

The Schema Wizard maps XML Schema elements to HTML form elements through its schema parser, and creates the framework and logic for an XML form wizard, as depicted in Fig. 1. Users use newly generated wizards to create and publish XML instances, which follow a schema, to any destinations such as publish/subscribe messaging systems or through SMTP. XML form wizards are Web applications that also serve as validating XML editors and are customized through schema annotations.

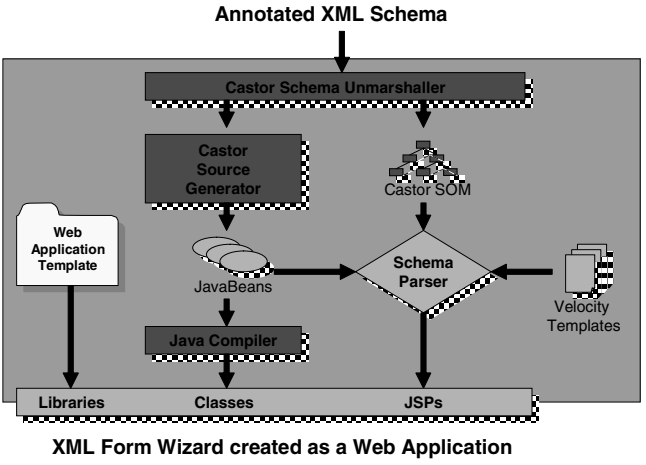


Fig. 2 The XML Schema Wizard simultaneously generates user interfaces, in-memory data bindings, and actions links between interface elements and data objects for an XML schema.

The Schema Wizard has been developed as a Web application compliant to JSP/Servlet specifications that generates and deploys new Web applications to help instantiate XML documents. A SW user defines her data model using a XML schema and annotates the schema to customize the HTML form view. [Fig. 2] After provided with a schema, the Wizard first creates the necessary directory structure and required libraries for a XML form wizard by unpacking a Web application template package. Second, the Wizard invokes Castor SourceGenerator [16] to create Java object definitions (a.k.a. javabeans) corresponding to individual XML schema elements, and compiles them into the previously created Web application directories. As the third step, the Wizard uses Castor's Schema Object Model (SOM) API to parse and process the schema in the memory. The SOM API is used for its convenient and specialized schema type access methods, which we prefer for this purpose over other XML processing APIs, i.e. DOM or JDOM.

The SchemaParser module, the heart of the Wizard application, traverses the schema using the SOM API and collects XML type and structure information. This information later is used to decide what type of template is necessary to generate final JSP pieces or *nuggets*. Velocity [21] templates are used for their simplicity to define JSP nuggets, each of which maps either a simple schema type or a complex type to one or more HTML elements rendered by JSP engine. The JSP page created for the root element is included by a higher level index page where the entire form view is finally constructed. The rules for mapping from XML schema types to HTML elements are listed in the following section.

2.2 Schema Element to HTML Mapping Rules

To generate XML instances based on XML schema elements, each element is mapped to one or more corresponding HTML form elements. The guidelines are as follows:

1. One HTML form page is generated for each possible root (or global) element, which is based on an anonymous or named global complex type.
2. XML schemas define elements for any level within XML document hierarchy. Some elements, for example, may not need any parent element to exist and can root a XML document. Such elements are called global elements and defined at the top level in a XML schema. Elements can also include sub elements and attributes. These elements are based on complex type definitions. SW generates forms for only global elements that are based on either named or anonymous complex types.
3. Each form has one form level submission, which submits all the content inside form elements and alters current values on the server side at once. Each form may also have more than one element level submission, which affects only related elements or attributes.
4. Simple, non-enumerating, non-repeating elements or attributes are mapped to a single input text field.
5. Simple typed elements or attributes must be based on predefined XML schema types such as numbers, character strings, and date and duration types. Any simple type can be restricted into a narrower type range. For an example, a new type can be generated from the integer type by restricting the values to the integer numbers between 0 and 100. Another example could be a character string type that must follow a pattern such as social security numbers with 9 digits and two dashes. Such schema types are mapped by SW to single input text fields and their values are only updated if a form level submission occurs. Data validation is performed on the server side while forms are used for some input size restrictions.
6. An enumerated simple type is mapped to a selection box with the values filled in. XML schemas allow some of the simple element or attribute values to be enumerated. Enumerated values are provided with type definitions inside schemas. Drop-down selection boxes with enumerated values filled in are used to map such types. If an enumerated element or attribute is optional, users are provided with an option to remove it. These types are updated by only form level submissions.
7. An unbounded simple type is mapped to an input field, a selection box, and two buttons to add and remove new values in addition to form level submissions. Unbounded simple types are elements that can repeat with the same or different values. The content of the input field is submitted and a new element is created when the submission is performed through the addition button. The new element is also added to the selection box. Users can remove any element by choosing its value from the selection box and submitting the form through the remove button.
8. A defaulted simple type is mapped to an input field with a default value.
9. A fixed value simple type is mapped to no form element, but the fixed value is written in plain text.
10. A simple Boolean-typed element or attribute is mapped to a pair of radio buttons with labels, “Yes” and “No.”

An unbounded complex value is mapped to, along with its sub elements, a selection box where the element's index can be chosen, and a pair of add and remove buttons. If any complex element index is chosen from the list, it can be removed by submitting through the remove button.

2.3 View Construction

Velocity templates are used to generate JSP pages. Examples of Velocity templates and JSP pages generated automatically may be found in [17]. Here, we will briefly go over an example and the generation of a JSP for an enumerated element.

All the Velocity variables starting with $ and surrounded by braces are replaced with the schema types information gathered by the SchemaParser. Elements and attributes are treated the same way in terms of the interface however, Castor SourceGenerator generates different source code for them, and the template reflects this through a Velocity variable, $attribute. Also, enumerated values are embedded into the JSP nugget through an array of values, $values, obtained through the SOM API and forwarded to the Velocity template engine by SchemaParser.

In the example, the parent element of the type is a complex type called rootElement. The JSP starts with retrieving the user input data, and checks whether this element has a parent or not. Based on the result of this check, the JSP nugget decides whether the memory object needs to be updated or the form is to be filled with the old data in memory. Finally, a selection box is printed out in HTML with enumerated values stored in an array of strings, values[].

All variables defined in generated JSP nuggets are local. Since parent elements include sub elements' JSPs, initializations are performed within parents' scope. Finally, the last part of the JSP decides whether this element is the root element and the submission buttons need to appear.

In XML schema, all the elements defined as global elements can be referenced at any level within the schema hierarchy. To support referenced global elements, we have designed the JSPs for parent elements so that they can be included by other complex typed elements.

The entry pages for HTML forms are index pages, which must not be included by other JSPs. SW does not provide such inclusions. Developers who use generated JSPs should also notice this requirement.

3 WSDL Wizards for Metadata Web Services

In the proceeding section we described an XML Schema Wizard that can be used to automate the process for creating specific wizards for generating XML messages. This system is appropriate for publish/subscribe style interactions. Here we describe our design for a variation of this system that is used to automatically generate remote procedure call (RPC)-style services. Instead of supporting all schema types, we need only to support one: WSDL. User interfaces generated from WSDL instances will be used to create server-side SOAP calls to back-end XML repositories.

As we argued in the introduction, an important subclass of Web services must be geared toward the creation and management of XML metadata. We refer to this particular subclass of Web services as Metadata Web services (MWS). Creating and deploying a new MWS has the following steps: *first*, create Java class bindings for the schema; s*econd*, create a Façade class to serve as Web service implementation and simplify manipulation of the data classes; *third,* deploy the service; *fourth*, generate client stubs from the service's WSDL, and *finally*, create the user interface to the service using the client stubs.

The Façade class for the server (step 2) may be used to simply manipulate the schema data objects, but for complicated schemas it also serves to simplify (at the expense of lost functionality) the task of step 5. These steps are somewhat tedious to perform manually. The tedious nature of creating these services has some unpleasant side effects: it inhibits the modification phase of trial schemas and (more globally) discourages the proliferation of lightweight, schema-based services.

The Schema Wizard itself, as described above, is designed for creating XML messages. In particular, it is not directly suitable for creating services and user interfaces from the WSDL schema. The WSDL document consists of two parts: service interface definitions and implementation bindings. The interface part contains the abstract definition of a service, which allows one to retrieve information of operations, and the types of the messages the operations receive and return. The implementation section contains concrete information on port and binding protocol. We thus do not want to create the user interface for a WSDL service description from the entire document but instead only the abstract portion (the portTypes) of the document.

The WSDL Wizard is designed to automate all parts needed to create a Metadata Web Service. This essentially has the following parts: a) selection of desired WSDL instance, b) generation of user interface and server-side bindings, and c) dynamic invocation of WSDL interface methods to manipulate the remote XML instance.

We are developing a Java package that allows SOAP procedures to be dynamically constructed for a given WSDL instance without needing to create client stub programs. This is described elsewhere [20], and in this paper we focus its combination with the Schema Wizard.

The major components need to accomplish this are illustrated in Figure 3. The user is assumed to have selected a particular service by inspecting a WSIL [19] document through a WSIL proxy client. The WSDL description of this interface is downloaded and the service generation procedure begins.

The important additional components are the WSDL Parser, GhostWSDL Parser, and User Data Parser. These parsers each are designed to provide access for manipulating in-memory representations of XML. A Controller component is responsible for managing all the components. The WSDL Parser is given the WSDL document that describes the desired service and creates an in-memory representation of the service interface. This document is then inspected by the Controller in order to dynamically generate a SOAP call.

As explained above, we do not need the entire WSDL description of a service to generate its interface. We thus construct a new “ghost” schema for the particular WSDL instance, based on the port type. This GhostWSDL instance contains only the information necessary to generate the user interface and the SOAP call. The generated GhostWSDL is passed to the Schema Wizard, which generates a user interface.

After the Schema Wizard user interface has been generated and the user has entered information, this must be passed to the Controller, which will need to use this data in the dynamically generated SOAP call. This data is accessed through the User Data Parser, which accepts the XML output of Schema Wizard. This output is simply an instance of the generated GhostWSDL schema, with the users’ data. For the simple types, the User Data parser parses the data in the XML to its corresponding Java value. For the complex type, the parser uses data to instantiate the Java object class which is generated by Schema Wizard. The results are placed to a Java object array.

Figure 3 shows how these components are related to each other. The arrow lines indicate the communication between the two components. The diagram uses the thicker lines for Internet request/response messages with the indicated protocols, and the thinner lines represent internal communications. The following is detailed explanation for each numbered step.

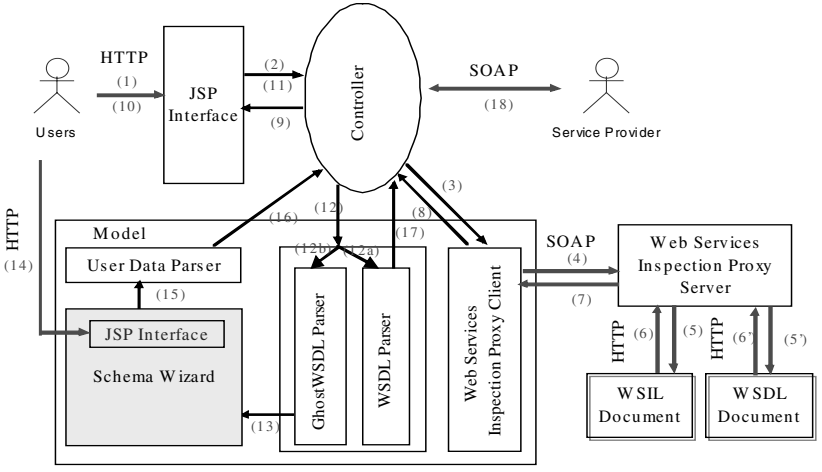


Fig. 3 A usage scenario illustrates the interaction of WSDI and Schema Wizard components.

1. The user requests the service list through a browser interface.
2. The JSP interface sends the user's request to Controller.
3. Controller communicates with the inspection proxy client (a WSIL client).
4. Proxy client makes the SOAP call to the web services inspection proxy server and requests the WSIL service list.
5. Proxy server makes the HTTP connection to a remote/local server for getting WSIL document
6. The remote/local server sends the WSIL document back to the proxy server
7. Proxy server sends back the service list to the client.
8. The proxy client responds to Controller.
9. Controller returns the service list to the user.
10. The user picks a desired Web service from the list WSIL list.
11. Steps (2) - (8) are repeated in order to retrieve the WSDL document, this time going through (5′) and (6′).
12. The WSDL document is sent to the parsers. In 12a, the WSDLParser creates in memory representation of WSDL. In 12b, the GhostWSDL is extracted from WSDL instance.
13. The GhostWSDL is sent to the Schema Wizard, which generates the user interface.
14. The user enters the required information of service into the system.
15. The wizard generates a GhostWSDL instance with the user's data, is created. If the GhostWSDL interface contains complex types, JavaBean classes are generated.

16. A Java Object array with the user data is passed to Controller.
17. The controller obtains an in-memory representation of the WSDL instance.
18. The Controller inspects the WSDL document, constructs, and launches the SOAP call to invoke the service.

4 Summary and Future Work

As part of our attempts to address a larger metadata management problem, this paper has presented two mechanisms for automatically deploying web-based applications for creating and manipulating metadata based on XML schemas. The Schema Wizard is a general purpose system for creating user interfaces and action bindings for a particular XML schema. As we have illustrated, the SW is used to generate browser interfaces automatically from a particular schema data model. It also binds the schema to in-memory data objects that can be used to manipulate schema instances. In addition to generating the view of the data model, we also generate the action bindings (in this case, HTML Form actions) that connect the user-supplied values to the data model. This is a useful component in message-based systems such as we describe in [13]. Future work on the Schema Wizard will be to introduce more customization of the presentation.

In contrast to one-to-many, message-style metadata services, the one-to-one, RPC-style metadata services are more meaningfully described with WSDL interfaces than message formats. To automatically generate the user interface to these services, we need to modify the Schema Wizard operation to display not the entire WSDL schema but the only the portion appropriate for display. In addition, the bindings must now change so that we can tie user interface actions to backend (rather than server-side) data instances, which we do with SOAP invocations dynamically generated from the WSDL instances. We have completed development of the preliminary libraries for dynamic WSDL invocations (supporting XML primitive types), and the WSDL wizard implementation based on these libraries is currently in progress. The next step will be to add support for WSDL complex types, as we describe in the paper.

References

1. Foster, I. Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Available from www.globus.org/research/papers/ogsa.pdf. See also Foster, I., Kesselman, C., Nick, J., Tuecke, S.: Grid Services for Distributed Systems Integration. Computer, 35(6), (2002).
2. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C.: Grid Service Specification, Draft 4. (2002).
3. Pierce, M., et al: Interoperable Web Services for Computational Web Portals. To be published in Proceedings of Supercomputing 2002 (2002).
4. Youn, C., Pierce, M., Fox, G.: Building Problem Solving Environments with Web Services. Submitted for publication in proceedings of Workshop on Complex Problem Solving Environments for Grid Computing, International Conference on Computational Science (2003).

5. Autonomic Computing: IBM's Perspecitive on the State of Information Technology. Available from www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.
6. Rhea, S., Wells, C., Eaton, P., Geels, D., Zhao, B., Weatherspoon, H., Kubiatowicz, J. : Maintenance-Free Global Data Storage. IEEE Internet Computing , Vol 5, No 5, pp 40–49 (2001).
7. Abramson, D., Buyya R., Giddy, J.: A Computational Economy for Grid Computing and Its Implementation in the Nimrod-G Resource Broker. To appear in Concurrency and Computation: Practice and Experience, Special Issue on Grid Computing.
8. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Service Description Language (WSDL) version 1.1 Available from www.w3c.org/TR/wsdl.
9. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D.: Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000. Available from www.w3.org/TR/SOAP/.
10. T. Berners-Lee, J. Hendler, and O. Lassila.: The Semantic Web. Scientific American (2001).
11. Lassila, O., Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. 22 February 1999.
12. Balsoy, O., et al: The Online Knowledge Center: Building a Component Based Portal. Proceedings of the International Conference on Information and Knowledge Engineering (2002).
13. Aydin, G., Kaplan, A.,Topcu, A. E.,Yildiz, B., Balsoy, O., Pierce, M., Fox, G.:. An XML Based System for Dynamic Message Content Creation, Delivery, and Control. Proceedings of IASTED International Conference on Information and Knowledge Sharing (IKS 2002).
14. Kahng, J., McLeod, D. :Dynamic Classification Ontologies. Computing the Brain: A Guide to Neuroinformatics, Academic Press, 241–254 (2001).
15. Thacker, S., Sheth, A.,Patel, S.: Complex Relationships for the Semantic Web S.. In *Creating the Semantic Web* (D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster, eds). MIT Press, 2001.
16. Birbeck, M. et al.: Professional XML, 2nd Ed.. Wrox Press, Inc. (671-721) 2001.
17. SchemaWizard Project Documentation. Available from ptlportal.communitygrids.iu.edu/schemawizard/index.html
18. Apache Axis User's Guide. Available from cvs.apache.org/viewcvs.cgi/~checkout~/xml-axis/java/docs/user-guide.html
19. Web Services Inspection Language, www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html
20. Jin, Y.: Web Service Dynamic Invocation. Community Grids Lab Internal Report. Available from grids.ucs.indiana.edu:8035/doc/Dynamics_Web_Services_Accessing.doc.
21. Apache Jakarta-Velocity Project. Available from jakarta.apache.org/velocity/index.html

Building Problem Solving Environments with Application Web Service Toolkits

Choonhan Youn[1,2], Marlon Pierce[2], and Geoffrey Fox[2]

[1] Department of Electrical Engineering and Computer Science, Syracuse University
cyoun@ecs.syr.edu
[2] Community Grid Labs, Indiana University
501 N. Morton Street, Suite 224
Bloomington, IN 47404-3730
{marpierc,gcf}@indiana.edu

Abstract. Application portals, or Problem Solving Environments (PSEs), provide user environments that simplify access and integrate various distributed computational services for scientists working on particular classes of problems. Specific application portals are typically built on common sets of core services, so reusability of these services is a key problem in PSE development. In this paper we address the reusability problem by presenting a set of core services built using the Web services model and application metadata services that can be used to build science application front ends out of these core services.

1 Introduction

Web browser-based scientific portals provide the user-centric view of computational grids [1]. Building on a foundation of core services such as job submission, file management, and session management, we can build sophisticated, domain-specific Problem Solving Environments (PSEs). Numerous such portals/PSEs have been developed, with varying degrees of specialization to applications. Some examples include NASA's Information Power Grid, San Diego Supercomputing Center's Hotpage and its application-specific spin-offs, Pacific Northwest National Laboratory's Ecce system, UNICORE, and our own Gateway project. References for these and other projects may be found in [2] and [3].

An important problem that must be addressed by PSE developers is the reusability and interoperability of their constituent core service components. Obviously, PSE developers want to reuse their own core service implementations to build new portals. We may go a step further and recognize the need for sharing reusable services between PSE development groups. Our experience has also shown that many of the basic services (such as batch script generation for queuing systems) are reinvented by many different groups [4]. A much improved process would be to build all portal services with well-defined interfaces and remote method invocation through a commonly accepted messaging system. Although consensus about common accepted definitions for interfaces, not to mention runtime interoperability, is hard to achieve between multiple groups, identification and reuse of another group's particular service

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 403–412, 2003.
© Springer-Verlag Berlin Heidelberg 2003

tool from a common repository is a realistic goal, provided that agreement may be reached on how to plug these services into one's PSE. One powerful approach is to use XML for interface definitions and messaging to facilitate implementation independence. Web services [5] and the Open Grid Service Architecture [6] provide the specific XML languages for these mechanisms.

In this paper we describe some specific core applications built in the Web service framework that can be used to build Problem Solving Environments out of reusable parts. We then address very important information requirements of these services: application metadata. Application metadata forms the basis for Application Web Service toolkits, which allow new scientific applications to be built out of core services.

2 Web Service-Based Computing Portal Architecture

Most computing portals are based on a three-tired architecture and thus have a classic stove-pipe problem in aspects of services. In order to integrate distributed services, the computing services should be designed for the interoperability and reusability. For addressing these challenges, we present a Web service based computing portal architecture around Web services model which have emerged as a popular standards-based, and service-oriented framework for accessing network-enabled applications. Web services have an XML-based, distributed computing paradigm to address the heterogeneous distributed computing services. It defines a technique for describing a service component, accessing a service, and discovering a service.

For reference we provide the following brief summary of the major constituents of Web service systems. More extensive descriptions can be found in Ref. [5]. Service description is provided by the Web Services Description Language (WSDL) [7]. Message invocation may use (at least in part) the Simple Object Access Protocol (SOAP) [8] for remote procedure calls. Information services and registries may use the WS-Inspection Language (WSIL) [9] and/or the Universal Description, Discovery and Integration (UDDI) [10] service. Web services are loosely organized, and we do not imply that the exclusive use of all these languages or services defines Web services.

Figure 1 illustrates the architecture for this kind of Web services system from the point of view of a portal. The basic point is that common interfaces can be used to link different multi-tiered components together. In the figure we have two distinct backend hosts, a high performance computing host and a database host, perhaps implemented by different groups. These hosts run various services that interact with the host to execute operating system command, etc. Information about these services is maintained by one or more information services. The user interacts with the service hosts and information servers indirectly, through client proxies maintained by the User Interface Server (UIS). The UIS is responsible for aggregating the various core services into application-specific PSEs. The various component interfaces may be collected as *portlets,* as described in [4], which define how user interface components can be plugged in and managed by portal administrators and users. Jetspeed [11], for example, is an open source portlet container system. Client stubs can bind and access these services with the protocol and mechanism prescribed in the service description by first contacting the service repository, UDDI that maintains links to the Web

service Providers' WSDL files and server URLs and finding a service to use. In this architecture, the control layer between the server that manages the user interface and the server that manages a particular service becomes decoupled. This separation makes it possible to provide the interoperable (or at least pluggable) services.

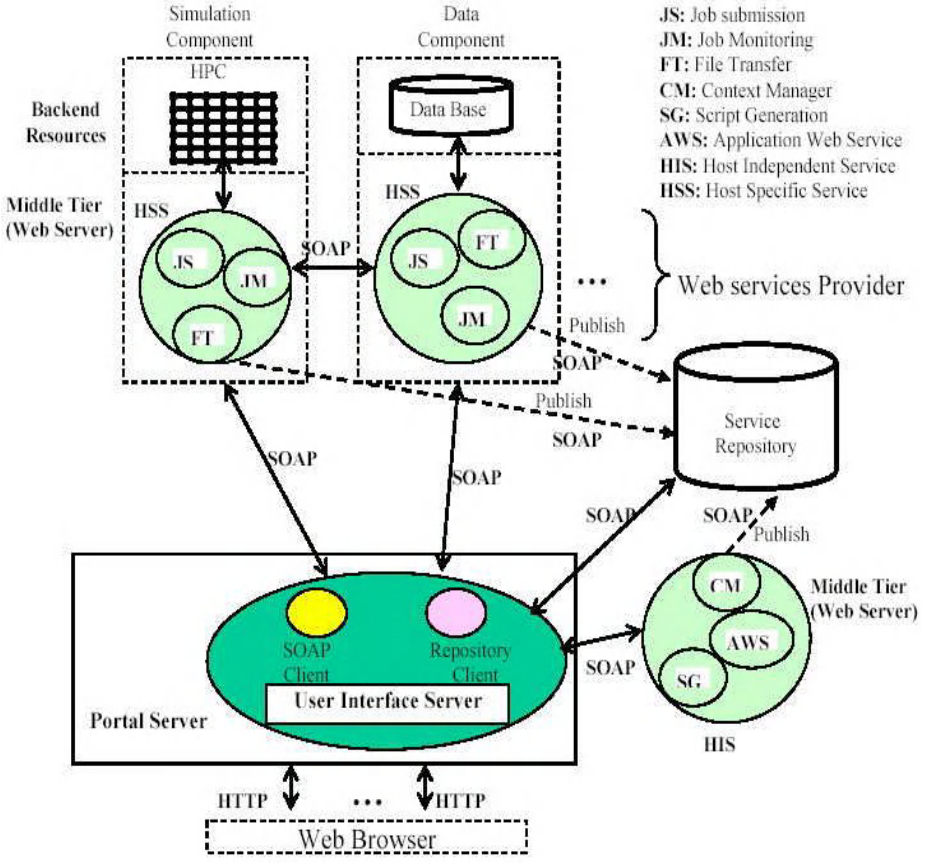


Fig. 1. Architecture of Web service-based computing portals.

3 Core Web Services for Computing Portals

We now consider several core services that must be implemented.

3.1 Job Submission

Computational portals must obviously allow users to execute scientific applications. We have defined a WSDL interface for executing commands on specific hosts systems (see Appendix, item 1). This service is remotely accessed through SOAP messages over HTTP. The service may execute operating system calls directly or may interact with Grid services through client APIs. We implement this service in Java and typically but typically use it to run external (non-Java) commands. We usually couple this service with the batch script service described below.

3.2 File Manipulation

Portal users must be able to move files between their desktops and various backend destinations, as illustrated in Figure 1. They must also be able to manipulate remote files transparently. We have defined Web services for such file management, allowing users to transparently move, rename, and copy files on remote back-ends. Files may also be transferred between desktop and backend, and cross-loaded between different backend sites. The full service interface may be obtained from the URL in the Appendix (item 2).

File uploading and downloading services illustrate the use of SOAP messages with attachments [12] in the RPC messaging style. SOAP attachments are non-XML files that are appended to the SOAP message and are useful for sending binary data and files with known MIME formats. For example, the file uploading service sends SOAP messages with attachments encapsulated in a MIME multipart format from the desktop to the SOAP server. We implemented file uploading and downloading service using the DataHandler class from the JavaBeans Activation Framework [13]. This class is used to represent arbitrary binary data (which could include text files, binary documents, and program data files). This provides a consistent interface to data available in many different sources and formats. Apache Axis [14] provides the serializer and deserializer for the DataHandler class so that a SOAP client (in Figure 1) may send SOAP messages with attachments to a remote SOAP server that is running the file management Web service. This service must still implement the file management details described in the WSDL interface for the received file.

We further allow the SOAP client to instruct one service to move a file to another service directly. We refer to this as cross-loading. For example, in Figure 1, a user may request that a file be transferred from the data storage component to the simulation component. The file management service implementation has the capability of also acting as a SOAP client to another service. Here the data component service receives a cross-load message from the SOAP client and uses this to construct a new SOAP upload request to the simulation component.

3.3 Context Management

The Context Management service archives interactions with the computational portal. Each user is assigned a unique set of context data, which is used to store all information from the user session. The default context data contains “Date”,

"LastTime", "Descriptor", "Directory", and "CurrentChild". In general, context data can be used to store any useful metadata. For example, we define user session data using application instance metadata (described below), but context data can be used to store the location of this XML file. Context data can later be recovered and edited by users to, for example, modify old sessions in order to resubmit jobs.

In our terminology, a context is a container that can hold an arbitrary number of string name-value pairs, as well as other contexts. These contexts are defined as a recursive XML schema (see example at URL given in item 3 of Appendix), so the schema supports an arbitrarily deep and complex tree-shaped data structure. In practice, a user's context data consists of a root context, with child contexts for particular problem groups for that user and grandchild contexts for particular problem sessions.

The context data service is built over the data model described above. We refer to the actual service interface for manipulating the context data as the Context Manager. This is defined in WSDL and exposes the following methods for manipulating context data:

1. A user can add one or more contexts according to the problem domain and sessions and so on. The context path should follow UNIX path. For example, the current path for a user context is empty character (""). So, if a user has "test" context for the problem domain and want to add new context, "session" under that context, the context path should be used "test/session". Using the JXPath [15] which implements XPath, the context path is checked whether or not the context is available.
2. A user can search the context data which is stored in a user context during the portal interaction using XPath queries and store the context data which is the input from the portal interaction for editing and managing in a specified context using the context path.
3. A user can remove the specified context including all of contexts and the context data in a user context using the context path.
4. A user can get the list of "children" list of that context for reviewing the context data, using the context path.

The WSDL for the entire interface may be obtained from the URL given in the Appendix (item 4). The above method calls are used for internally manipulating the context data. Internally, the context data instance is represented as a set of data classes created with Castor [16]. We store these schema instances persistently either on the file system (mapping context data nodes to directories) or in an XML-native database such as Xindice [17].

Context Management services may be used to store and retrieve arbitrary pieces of information. In practice, we use this service to manage instances of the application metadata created by the Application Web Services, described below.

3.4 Script Generation

We have developed a Batch Script Generation (BSG) Web services for users who are unfamiliar with high performance computing systems and queue schedulers. The WSDL interface may be obtained from the URL given in the Appendix (item 5). This service assists users in creating job scripts to work with a particular queuing system such as the Portable Batch System (PBS). From our experience, most queuing

systems were quite similar and could be broken down into two parts, a queuing system-specific set of header lines, followed by a block of script instructions that were queue independent. Queue scripts are actually generated on the server, based on user's choice of machine, application code, memory requirement, input/output file name and parameter, etc. This information is stored as an XML document, the Application Instance Descriptor. The BSG service accepts this XML document as an argument and generates the queue script of the requested type. The structure of the BSG service implementation allows it to extensibly support different queuing systems.

A previous version of this service has been described in Ref [18]. We have extended this service to integrate it with the Application Instance schema described below. The latest version returns both the generated batch queuing script and the shell script needed to submit the queue script to the queue of a specified host. All information needed to generate these scripts is obtained from the Application Instance data. Actual submission uses the Job Submission service described above. Scripts may be moved to the appropriate host using the File Management service.

3.5 Job Monitoring

Job monitoring services may be built in one of two ways: periodic client polling and server event notifications. We currently use the polling method, which we prefer for reliability, but have built event-style prototypes based on email notifications.

The polling service makes one method available for use by clients through a SOAP RPC for monitoring the execution of a job running in a queuing system. The WSDL interface may be obtained from the URL given in the Appendix (item 6). The input to this method is the user account name and the scheduler type, such as "PBS". The service implementation is designed as a factory so that support for particular schedulers can be added in a well defined way. If the scheduler type is not supported by the Web service, then a message to the effect is returned to the client. If the scheduler is supported, then the user name is passed to a handler created for that specific scheduler. The scheduler handlers are custom-written methods that generate a WSDL complex type, effectively an XML data object given the user name and return the array of the generated a WSDL complex type that contains the job status of the scheduler.

4 Application Web Services and Toolkits

The core services described above are intended to serve general purposes. These must be organized in a more meaningful fashion for use in PSEs. In particular, we have developed a set of XML schemas for describing scientific application metadata. Here "application" means specifically some science code on the computational grid. Given that the application has been installed on some hosts, we want to describe how this application may be added to a PSE.

We want to specifically enable two different classes of PSE users: application managers and application users. Application managers are responsible for adding and managing the user interfaces for applications, while application users are scientists or

engineers who wish to use the application. Given these two classes of users, application metadata descriptor data models come in two sets: abstract application descriptors for managers and application instance descriptors for end users. The descriptor components and rationale behind their structure is described in [4]. The URLs for these schemas are given in the Appendix (item 7).

For internal reference we review the schema structure briefly. Abstract application descriptors consist of one or more applications (schema complex types), which are described by various fields such as application name and version, application flags and flag formats. We also include elements that describe the Web service bindings for the applications input, output, and error. A particular application is also described by the Web services needed to run the application on a particular host. The service bindings are in turn bound to service hosts. Supplemental host information (such as host name and IP address, queuing system, standard working directories, and so on) are described is complementary schemas. We have attempted to design these schemas to be modular, so that third party host and queue/scheduler schemas may be plugged into the system.

The application metadata must now be turned into a useful service. The schemas themselves can be mapped to Java language bindings automatically using tools such as Castor [16]. The schemas are too complicated, however, to be used to define a useful WSDL interface, so we instead implement "Façade" wrapper classes that simplify access to the schemas, at the loss of some functionality. The wrapper interfaces are still quite large, and the URL for the wrapper interface definition is in given in the Appendix (item 8).

Web-based user interfaces may be built out of these interfaces. Application managers may provide various pieces of information needed to create instances of Abstract Application Descriptors. The illustrated form indicates the simple case of adding an application with one input and one output field, no command line flags, etc. The application can then be bound to a set of services on various hosts for submission and file transfer. The form elements are mapped to client stubs, which in turn (via SOAP) can be used to view and modify remote AWS schema instances.

Information provided by the application manager (which services may be used for file input and output, which may be used for application execution and so on) is used to generate user interfaces for application users. Web forms may be generated for the user for submitting a particular application (a finite element code, in this case) to the indicated queuing system, based on an Application Descriptor instance.

Application metadata is intended to serve as a supplemental, specialized data model that is more appropriate for Application Portals than WSIL or UDDI. We believe this has an important role in computational services. The OGSA specification provides an important mechanism for describing particular service metadata. On top of this we need service classification systems, which organize WSDL definitions in meaningful ways. Such classification schemes are subjective and not likely to be fully standardized, so we anticipate a future interesting problem of managing many different application metadata ontologies.

In addition to the application schema, we define the host and queue binding description schema which is modular container as our design mechanism. The host binding schema contains three elements about resources: First, the host information element such as DNS name and IP address; Second, the application information element need to invoke the actual application code running on that host such as location of the executable code, location of the workspace or scratch directory; Third,

queue information element such as the location of the executable queue command running on that host and the queue type which the host supports. Like the application schema, for maximizing the flexibility, we also provide a general purpose "parameter" element that allows for arbitrary name-value pairs to be included. This can be used for example to specify environment variable settings needed on a particular host by a particular application. This schema also has a queue binding that contains information needed to perform queue submissions. The queue binding schema contains the queue script information (the job name, user account name, memory size, the number of CPUs, the wall time, the email options and the general purpose parameter for the queue) based on the queuing system, such as PBS. These schemas are designed to be plugged into the application descriptor schemas, and may be replaced by other schema definitions.

5 Summary and Future Directions

In this paper, we have presented the design and implementation of several core portal services and Application Web Services. These form the basis of PSEs, and our emphasis has been on the development of reusable services that can form the basis for multiple PSEs. We have identified and classified the kinds of services depending on the service deployment. Based on this classification, the portal developer can construct specific implementations and composites of primitive service components and can also provide services that may be shared among different portals. We have demonstrated application-specific services and data models that can be used to encapsulate entire applications independently of the portal implementation. As we discuss elsewhere [4], the current infrastructure provides the service interoperability and reusability.

Next, we will consider some specific extensions (security, negotiation, job composition) to the architecture of a Web service based computing portal. First, Secure Web services will be considered that we need secure SOAP messages between user interface server and the repository and the service provider for the user authentication, based on the message-level security architecture. SOAP security should be provided through standard interfaces to independently specific mechanisms such as Kerberos [19], PKI [20]. The general approach is to use the assertion based security such as SAML [21], WS-Security [22] into SOAP messages. An assertion, for example, SAML, WS-Security, is an XML document describing the information about authentication acts performed by subjects, attributes of subjects and authorization decisions, created with a specific mechanism.

References

1. Foster, I., Kesselman, C. (ed.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kauffmann, 1999
2. Fox, G., A. Hey (eds): Concurrency and Computation: Practice and Experience. Special issue on Grid Computing Environments. Vol. 14, No. 13–14 (2002). Abstracts and links to accepted papers available from http://aspen.ucs.indiana.edu/gce/gce2001index.html

3. Berman, F., Fox, G., Hey, T. (ed.): Grid Computing: Making the Global Infrastructure a Reality. To be published by Wiley, London 2003. Book chapters are available from http://www.grid2002.org/
4. Pierce, M. et al.: Interoperable Web Services for Computational Portals. Proceedings of Supercomputing 2002. Available from http://sc-2002.org/paperpdfs/pap.pap284.pdf (2002)
5. Graham, S. et al.: Building Web Services with Java. SAMS, Indianapolis, 2002
6. Foster, I. et al.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Accessed from http://www.globus.org/rearch/papers/ogsa.pdf
7. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Service Description Language (WSDL) version 1.1. W3C Note 15 March 2001. Available from http://www.w3c.org/TR/wsdl
8. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D.: Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000. Available from http://www.w3.org/TR/SOAP/
9. Ballinger, K., Brittenham, P., Malhotra, A., Nagy, W. A., Pharies, S.: Web Service Inspection Language (WS-Inspection) 1.0. IBM and Microsoft November 2001. Available from http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html
10. Bellwood, T., Clemont, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y. L., Januszewski, K., Lee, S., McKee, B., Munter, J., von Riegen, C.: UDDI Version 3.0 Published Specification. 19 July 2002. Available from http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
11. Jetspeed Overview. Available from http://jakarta.apache.org/jetspeed/site/index.html
12. Barton, J. J., Thatte, S., Nielsen, H. F.: SOAP Messages with Attachments. W3C Note 11 December 2000. Available from http://www.w3.org/TR/SOAP-attachments
13. JavaBeans Activation Framework. Available from http://java.sun.com/products/javabeans/glasgow/jaf.html
14. Apache Axis. Available from http://xml.apache.org/axis/
15. JXPath. Available from http://jakarta.apache.org/commons/jxpath/
16. The Castor Project. Available from http://castor.exolab.org/
17. XML database, Xindice. Available from http://xml.apache.org/xindice/
18. Mock, S. et al.: A Batch Script Generator Web Service for Computational Portals. Proceedings of Communications in Computation. International Multiconference on Computer Science, 2002
19. Kerberos: The Network Authentication Protocol. Available from http://web.mit.edu/kerberos/www/
20. Understanding PKI. Available from http://verisign.netscape.com/security/pki/understanding.html
21. Blakley, B., Cantor, S., Erdos, M. et al: Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML). OASIS 10 January 2002. Available from http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-09.pdf
22. Atkinson, B., Della-Libera, G., Hada, S., Hondo, M., Hallam-Baker, P. et al: Web Services Security (WS-Security) Version 1.0. IBM 05 April 2002. Available from http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

Appendix: URLs for Schema Definitions and WSDL Interfaces

The schema definitions and interface definitions for the services described in this paper are available from the following URLs:

1. Submitjob WSDL interface. Available from http://www.servogrid.org/GCWS/services/Submitjob?wsdl

2. File Transfer WSDL interface. Available from http://www.servogrid.org/GCWS/services/FileService?wsdl
3. Context Manager Schema. Available from http://www.servogrid.org/GCWS/Schema/Cmhtml/cm.html
4. Context Manager WSDL interface. Available from http://www.servogrid.org/GCWS/services/ContextManager?wsdl
5. Script Generation WSDL interface. Available from http://grids.ucs.indiana.edu:8045/GCWS/services/ScriptGenerator?wsdl
6. Job Monitor WSDL interface. Available from http://grids.ucs.indiana.edu:8045/GCWS/services/Jobmonitor?wsdl
7. Application Web Service Schemas are available from http://www.servogrid.org/GCWS/Schema/index.html. These are also described in the following internal report: Pierce, M., Youn, C., and Fox, G.: Application Web Services. Available from http://www.servogrid.org/slide/GEM/Interop/AWS.doc and http://www.servogrid.org/slide/GEM/Interop/AWS2.doc
8. Application Descriptor WSDL Interfaces. Available from http://www.servogrid.org/GCWS/services/ApplicationDescriptor2?WSDL and http://www.servogrid.org/GCWS/services/ApplicationDescriptor3?WSDL

GSF: A Problems Solving Environment Supporting Multi-models Parallel Grid Programming

Qian-ni Deng and Xin-da Lu

Dept. of Computer Sci. & Eng., Shanghai Jiaotong University, Shanghai 200030, P.R .China
{deng-qn, lu-xd}@cs.sjtu.edu.cn

Abstract. Web service is a grid computing technology that promises greater ease-of-use and interoperability than previous distributed computing technologies. In this paper we propose Group Service Framework(GSF), a grid problems solving environment based on Microsoft .NET, to: (1) locate and harness volunteer computing resources, and (2) support multi parallel programming paradigms such as Master/Slave, Divide and Conquer, Phase Parallel and so forth in Grid environment, (3) allocate tasks and realize load balancing dynamically and transparently for different grid application.

Keywords. Web Service; Volunteer computing; Grid computing; parallel programming paradigm

1 Introduction

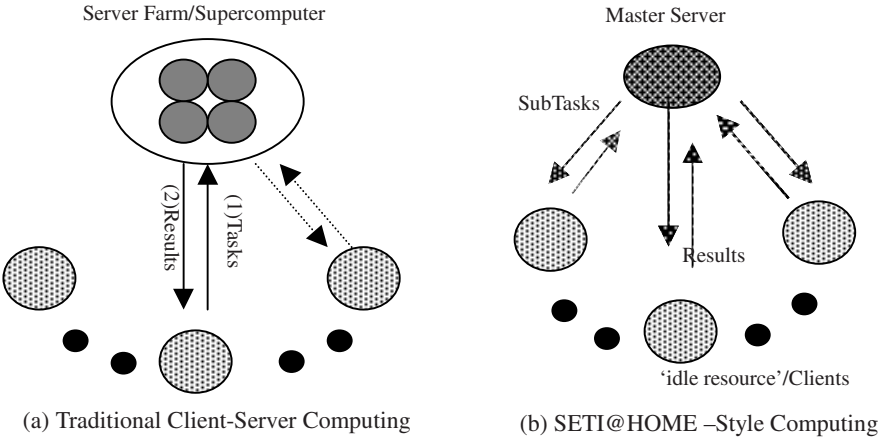
Volunteer computing has the idea that home PCs are mostly idle and thus could be harnessed for solving complex computational problems. This idea is to decompose the problem into many chunks that can run concurrently with very little interaction, referred by some as "embarrassingly parallel" computation. The best known example of this idea are SETI@HOME. A comparison of traditional high performance server versus volunteer computing is shown in Figure 1. Though volunteer computing model has been successfully used to handle some complex computation problems, there are many inherent shortcomings in this model:

(1) Volunteer computing can not be used to solve the computation problems that have to be divided into several chunks which run concurrently with interactions.
(2) Volunteer computing only supports Master/Slave programming paradigm.
(3) Generic volunteer computing system is application-specific and only can be used for one application.
(4) Lack of security and dynamic management mechanisms.

Aiming at above problems，we brought forward a grid computing framework and built a generic problem solving environment based on Master/Group/Worker model and Web Service[3] technology. Unlike those application-specific volunteer computing systems, this problem solving environment not only can locate and harness volunteer computing resource for different applications, but also support multi-programming models such as Master/Slave, Divide and Conquer, Phase Parallel [5] and so forth parallel programming paradigms. The main contribution of this paper is

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 413–419, 2003.
© Springer-Verlag Berlin Heidelberg 2003

that we attempt to explore a grid-oriented generic programming model well combined with web service technology.



(a) Traditional Client-Server Computing (b) SETI@HOME –Style Computing

Fig. 1. A comparison of traditional computation servers versus resource sharing approach

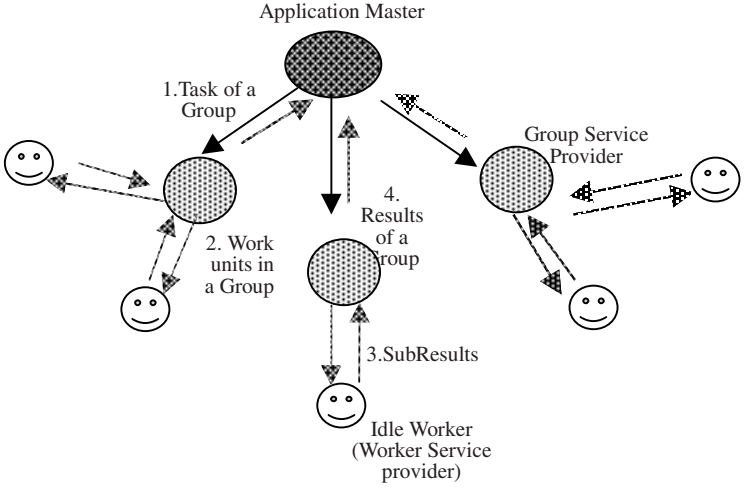


Fig. 2. Master/Group/Worker Style computing

2 Master/Group/Worker Model

Fig 2 shows the Master/Group/Worker model. There are three main roles:

(1) Worker Service Provider: A worker would be an idle PC who can provide volunteer computing service intermittently.

(2) *Group Service Provider:* Group service provider has the abilities to locate worker service providers and bring together the discovered workers into a group to solve an SPMD problem. In the lifetime of a group service provider, it locates workers, decomposes the task of a group into several work units, sends work units to located workers who perform computing. From the point of view of an application master, one group service provider can be looked as an SPMD computational web service. In our problem solving environment it can exists one or more group service providers.

(3) *Application Master:* In our problem solving enviroment an application can be of SPMD or MPMD programming model,when the application is SPMD model,only one group service can satisfy its computing request, when the application is MPMD model, serveral group services are needed. For example, to calculate the formula:

$$(u_1^2+u_2^2+\ldots+u_n^2\ ) / (A[1]*B[1] +A[2]*B[2] + \ldots + A[m]*B[m]) \quad (1)$$

The Application Master just need to divide the task into two groups, one calculates:

$$x=u_1^2+u_2^2+\ldots+u_n^2 \quad (2)$$

the other calculates:

$$y=(A[1]*B[1] +A[2]*B[2] + \ldots + A[m]*B[m]) \quad (3)$$

Firstly the two group service providers decompose the task in each group into work units respectively and parallely according to the partition strategy specified by application master, then the two groups work respectively and independently to compute the value of x and y, finally the application master callbacks the value of x and y from the two group respectively and compute the final result: *x/y*.

In above mentioned model a programmer need not to specify the number of workers in advance, it just need to specify the data partition strategies when the programmer define the application master. The benefits of this model are: easy to programming, dynamic load distribution and load balance. Apparently, besides Master/slave model, the architecture in fig.2 can support " divide and conquer" programming paradigm. If the process shown in fig.2 repeats several times, it can support "phase parallel" programming paradigm.

3 Data Partition and Merging Strategy

3.1 Automatic Data Partition

To partition the task of an SPMD program in a group automatically，it needs to specify the data partition strategies of a group when the programmer define the application master. By defining an object of C# class: DataStructure, the programmer can specify the dataset and partition strategie of one SPMD task.

```
public class DataStructure
{  const int BROADCAST=0;
```

```
const int PER_ELEMENT=1;
const int SAME_SIZE=2;
const int DIFFERENT_SIZE=3;
int dds;    //data dispatching strategy, range from 0~3
int num_of_worker;
String datatype; //it can be any C# data type
byte[] data; // the date set
int[] data_size;   }
```

Fig. 3. C# class : DataStructure

This structure supports following partition strategies:

- BROADCAST, broadcast the whole dataset to all workers in a group.
- SAME_SIZE, averagely partition the dataset to each workers according to the amount of works in a group.
- PER_ELEMENT: send first element of data-set to first worker, second element to second worker, the rest may be deduced by analogy until last worker receive the element. If the number of elements is greater than the number of workers, the residual elements may be allocated in terms of above rule until all elements in data-set been allocated.
- DIFFERENT_SIZE: each worker in a group sees after different size computing tasks. By defining the value of num_of_worker and data_size[] array in the class DataStructure, the application programmer can specify the allocation scheme and the number of elements allocated to each worker in a same group.

3.2 Automatic Computation Results Merging Strategy

To automatically merge sub-results collected from group service providers, it needs to specify the results merging strategies when the application programmer defines the *application master*. By defining an object of a C# class: Recv_DataStructure, the programmer can specify the result sets and merging strategy of an application. The merging strategies supported by our problems solving environment include: calculating maximum, minimum, summation and product.

```
public class Recv_DataStructure {
String recvtype;
int recvcounts;
byte[] recvbuf;
const int NOTHING=4;
const int MAX=0;  static final int MIN=1;
const int SUM=2;  static final int PROD=3;
int  merge_strategy; // range from 0~4
String result_type; // type of merged results
int result_counts;  //number of merged results
byte[] result;  }
```

Fig. 4. C# class: Recv_DataStructure

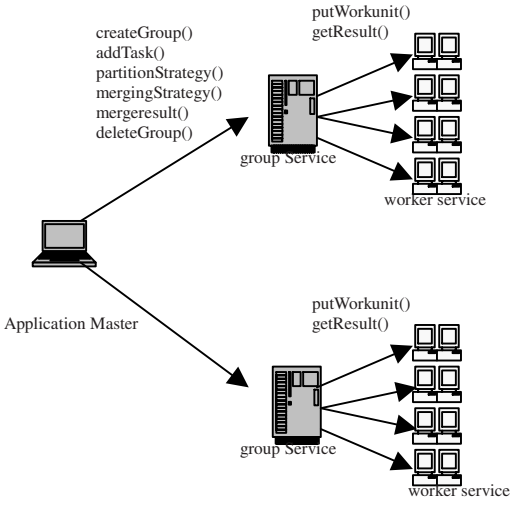


Fig. 5. Group Service Framework

4 Group Service Framework

We have already implemented the group service framework (GSF) and developed a genetic parallel problems solving environment based on Microsoft .NET. Microsoft .NET is a set of software technologies designed to connect information, people, systems, and devices. The foundation of .NET is XML Web services: reusable applications written in Extensible Markup Language that allows users to connect with applications and data on the Internet. Based on the platform of .NET, all the roles (Workers, Groups, Masters) in our problems solving environment are defined by WSDL language and communicate with each other by XML language. Fig. 5 shows an example of how group services hide volunteer computing for an MPMD application.

Recently, the Globus group has started integrating web service with globus toolkits, and defined a standard architecture OGSA[1,2]. OGSA gives a standard of how to define the grid service and doesn't limit the implementation of grid service. We think that our group service is one kind of grid services who can server for many parallel computing applications. To accord with OGSA(Open Grid Service Architecture), we plan to re-define and implement all the service interface following the standards of OGSA specification. Fig. 6 shows the future Group Service Framework coincident with OGSA.

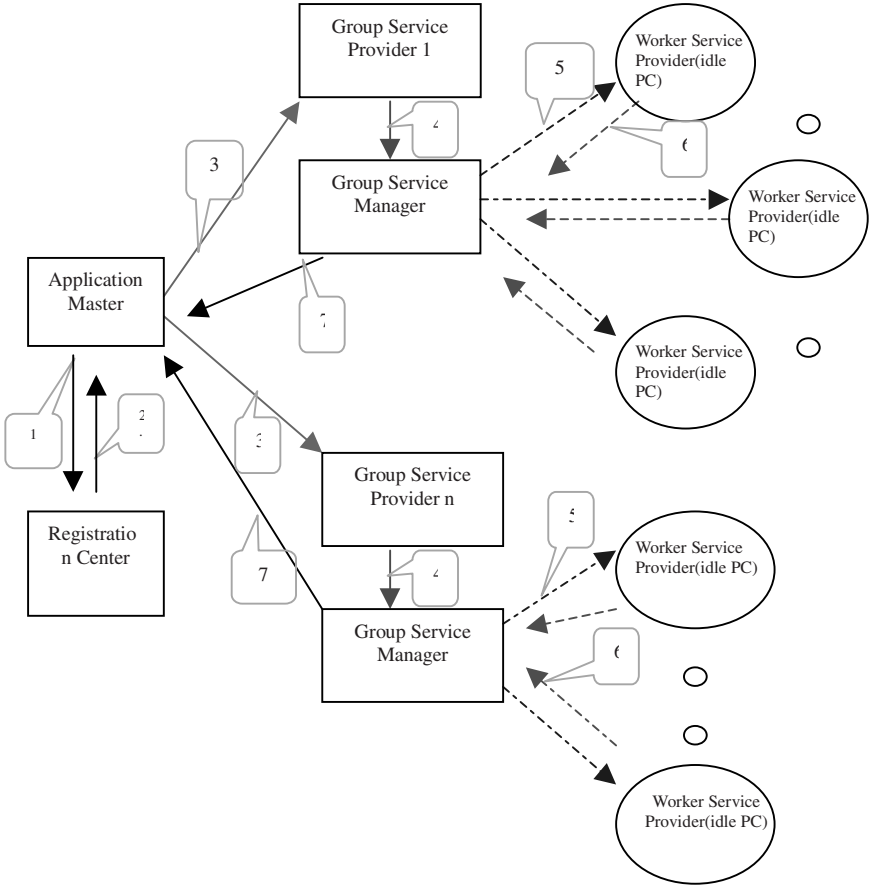


Fig. 6. Group Service Framework according with OGSA

According with OGSA, our future Group Service Framework should work as following processes:

(1) *Application master* submits one or more group service requests to *Registration center;*
(2) *Registration center* returns one or more *group service handlers* to *Application master;*
(3) *Application master* defines the lifetime of group service, asks *group service provider* for creating the grid service instance;
(4) *group service provider* creates an instance;
(5) *group service instance* discovers *worker services* , then dispatchs work units;
(6) after completing computing, *worker* returns the sub-results;
(7) *group service instance* keeps lifetime, returns the group results to *Application Master.*

5 Results, Related Work, and Conclusion

Currently, we have used Grid Service Framework based on Microsoft .NET and Java platform to implement several simple parallel computing applications, including π calculating parallel program, Mandelbrot set application, and parallel string matching, and have been able to use these with different kinds of computational worker clients including a .NET application running on PC and a Java applet running on non-PC. The related experiment results have already been presented on [4]. From the experiments we get the conclusion that Group service Framework proposed by this paper is suitable for generic parallel numerical computing. These results are just the beginning of our research in which grid computing and web services can be used together. We plan to redefine the Group service Framework according to OGSA specification and continue developing Group Service Framework as well by exploring issues such as authentication, authorization, and resource accounting.

References

[1] Ian Foster, Carl Kesselman , et al. Open Grid Service Infrastructure WG, Global Grid Forum. The Physiology of the Grid, An open Grid Service Architecture for Distributed Systems Integration. http://www.globus.org

[2] S.Tuecke, K. Czajkowski, I.Foster, J.Frey, S.Graham, C.Kesselman; Open Grid Service Infrastructure WG, Global Grid Forum. Grid Service Specifications. Draft 2, 7/17/2002. http://www.globus.org

[3] Microsoft Corporation.XML and .NET White Papers. http://www.microsoft.com/serviceproviders/whitepapers/xml.asp

[4] QianNi Deng, XinDa Lu. JOGR-Utilizing Java Object Group Relationship to Support Multi-model Parallel Programming, ACTA ELECTRONICA, Nov. 2002.

[5] Kai Wang, ZhiWei Xu. Scalable Parallel Computing: Technology, Architecture, Programming. McGraw-Hill. 1998.

Implementation of a Grid-Enabled Problem Solving Environment in Matlab

Hakki Eres, Graeme Pound, Zhouan Jiao, Jasmin Wason, Fenglian Xu, Andy Keane, and Simon Cox

School of Engineering Sciences,
University of Southampton,
Highfield, Southampton, SO17 1BJ, UK
{eres, gep, z.jiao, j.l.wason, f.xu, ajk, sjc}@soton.ac.uk
http://www.geodise.org/

Abstract. In many areas of design search and optimisation one needs to utilize Computational Fluid Dynamics (CFD) methods in order to obtain numerical solution of the flow field in and/or around a proposed design. From this solution measures of quality for the design may be calculated, which are required by optimisation methods. In large models the processing time for the CFD computations can very well be many orders of magnitude larger than the optimisation methods; and the overall optimisation process usually demands a combination of computational and database resources; therefore this class of problems is well suited to Grid computing. The Geodise toolkit is a suite of tools for Grid-enabled parametric geometry generation, meshing, CFD analysis, design optimisation and search, database, and notification tools within the Matlab environment. These grid services are presented to the design engineer as Matlab functions that conform to the usual syntax of Matlab. The use of the Geodise toolkit in Matlab introduces a flexible and Grid-enabled problem solving environment (PSE) for design search and optimisation. This PSE is illustrated here with an exemplar problem.

1 Introduction

The process of design search and optimisation involves the modelling and analysis of engineering problems to yield improved designs [1]. Independent design parameters that the engineer wishes to modify, and relevant design constraints are identified, and a measure of the quality of a particular design (the objective function) is computed using an appropriate model. A number of design search algorithms may then be used to yield more information about the behaviour of a model over the parameter space, and to minimise/maximise the objective function to improve the quality of the design by modifying the design parameters subject to any constraints imposed on them. This process may include lengthy and repetitive calculations to obtain the value of the objective function with respect to the design variables. Design optimisation with regard to fluid dynamics is relevant to, amongst others, the aerospace, automotive and oil industries. Computational Fluid Dynamics (CFD) methods allow the engineer to

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 420–429, 2003.
© Springer-Verlag Berlin Heidelberg 2003

analyse the properties of a design. However, they often require computer aided design (CAD) tools to generate parametric design geometries, mesh generation programs to mesh the flow domain, and CFD solvers to obtain an approximate solution for each design search point. Therefore, the modeling and analysis is usually the computationally expensive part of design search and optimisation process. To perform the numerous solutions required for extensive parameter exploration during a design search in this domain normally requires access to significant computational and database resources.

Engineering design search and optimisation is also a data intensive process. Data may be generated at different locations with different characteristics. It is often necessary for a design engineer to access a collection of data produced by design and optimisation processes to make design decisions, perform further analysis and carry out post-processing. Databases are valuable to expose the state of the design process to context sensitive design advisers, allowing them to provide dynamic advice to the user. Databases therefore play an essential role in our architecture, where it is important to capture the process of how results are obtained in addition to storing the results themselves.

The Geodise project [2] aims to aid the engineer in the design process by making available a suite of intelligent design optimisation and search tools, CAD packages, mesh generation tools, and CFD analysis packages integrated with distributed Grid-enabled computing, analysis, data, and knowledge resources. Facilitating the use of such design search tools requires the integration of intelligent design advisers, which are able to support the engineer throughout the design process by providing ontology services, annotation services, and context sensitive advice based on the states of the computation.

A problem solving environment (PSE) provides the user a complete and integrated environment for problem composition, solution, and analysis [3]. PSEs exist for numerous application areas, such as the solution of partial differential equations, scientific visualisation, and computational chemistry. Various parts of the design search and optimisation process can be integrated as a PSE to aid the design engineer during tedious steps. Graphical components with drag-and-drop capabilities can ease the involvement of a novice design engineer; on the other hand, an expert user might require more flexibility which can only be provided through direct access to a scripting language.

All these objectives impose a number of requirements upon our choice of environment. The environment should provide an intuitive interface to the available Grid resources. A Grid-enabled PSE abstracts the complexities of accessing Grid resources by providing a complete suite of high level tools designed to tackle a particular problem area [4]. There are various ongoing projects for Grid-enabled PSEs, for example, Nimrod/G [5] is a tool that facilitates parameter studies over computational Grids, and Triana [6] is a graphical programming environment which abstracts the complexities of composing distributed workflows.

Whilst it is possible to reduce the complexity of the technologies faced by the user, it is important that the environment chosen has the flexibility to tackle the subtleties of a wide range of workflows within the problem domain. A pre-

vious prototype [7] that consisted of a wizard style web portal that guided the user through design optimisation problems was useful. However, it proved to be inflexible, because it failed to provide the user with the ability to re-use and re-compose workflows for large-scale problems. The complexity and variation of the workflows involved in typical design processes mean that a scriptable environment where the user can tailor workflows to the task in hand is valuable.

The user interface used to expose the functionality provided by the Geodise PSE is chosen to be the commercial Matlab environment [8]. The Matlab package provides an interpretive language for numerical computation, built-in math and graphics functions and numerous specialised toolkites for advanced mathematics, signal processing and control design. The Matlab product is widely used in academia and industry to prototype algorithms, and to visualise and analyse data. Matlab 6.5 also features a number of 'just-in-time' acceleration technologies to increase the performance of native Matlab code. Additionally, Matlab enables the programmer to access Java classes, to create objects, and to call methods of these objects within the Matlab environment by using Matlab's functions and commands.

The rationale behind adopting Matlab as the user interface for the Geodise PSE is pragmatic. As a toolkit that may be integrated into an environment routinely used by our industrial and academic partners the Geodise PSE becomes a flexible tool, part of the engineer's arsenal. The NetSolve system [9] which uses a client-server architecture to expose platform dependent software libraries has also successfully adopted Matlab as a user interface. It is an examplar of a hosting environment which can be used to couple together a range of grid-services which expose themselves via open-standard protocols.

The final Geodise toolkit will be composed of a hierarchy of components. Low level compute and database functions will be available to the user, in addition to a number of higher level design search and optimisation, pre/post-processing, and CFD functions. All of these components will be available through a suite of intelligent design advisers that will guide the user through the design process, and facilitate the use of toolkit components. The remainder of this paper focuses on the process of exposing Grid enabled resources to the Matlab environment, allowing us to compose the required low level geometry and mesh generation, analysis, visualisation, data access, and notification functionality in a scriptable PSE. We first describe the architecture and the functionality of the existing Geodise toolkit. We then demonstrate the use of these functions in an example iteration of the design process.

2 Geodise Toolkit

The user of the Geodise toolkit acts as a client to remote compute resources that are exposed as Grid services. Users should be authenticated, and then authorised to access resources to which they have rights. They need to be able to submit their own code to compute resources, or run software packages that are available as services. The user should be able to discover available resources, decide

where to run a job and be able to monitor its status. It is essential that the user be able to easily retrieve the results of a simulation. Additionally, the requirements of design search and optimisation mean that compute resources must be available programmatically to algorithms that may initiate a large number of computationally intensive jobs serially or in parallel.

The Globus toolkit [10] provides middleware that allow the composition of computational grids through the agglomeration of resources which are exposed as Grid services. This middleware provides much of the functionality required by our toolkit including authentication and authorisation, job submission, data transfer, and resource monitoring and discovery.

Client software to Globus Grid services exists natively on a number of platforms [11] and also via a number of Commodity Grid (CoG) kits [12] that expose Grid services to 'commodity technologies'; including Java [13], Python, CORBA [14] and Perl. By using client software to Grid services written for these commodity technologies the developer of a PSE is able to remain independent of platform and operating system.

The independence allowed by adopting a commodity technology motivated development of the Geodise toolkit over the Grid service client APIs of the Java CoG kit v.0.9.13 [13]. Java is a mature technology that runs compiled byte-code within a virtual machine. The Matlab environment itself runs within a Java Virtual Machine (JVM), and provides an external interface which allows Java classes to be instantiated and invoked easily within the Matlab workspace. The support of Java version 1.3.1 by Matlab 6.5 provides the utility which makes the Java language suitable for programming Grid middleware.

The Java CoG provides a number of low-level mappings, in the form of a number of Java packages, which are APIs to the respective Globus Grid service clients. To expose the functionality available from the Java CoG to the Matlab user it was important to present functions which are consistent with the behaviour and syntax of the Matlab environment. All functions are implemented in the Matlab language, and they call Java classes which access the Java CoG API. Additionally, these functions are written with the intention that they may be incorporated programmatically into the higher level components of the toolkit.

Table 1 lists the implemented functions in the Geodise toolkit. This set of functions can be loosely categorized as: i) functions which allow the user to run and control jobs on Globus compute resources, ii) functions which are used to archive, query, and retrieve data, and iii) functions which are used to notify the user.

The toolkit command `gd_createproxy` allows a user to create a temporally limited Globus proxy certificate within the Matlab environment, essentially creating a point of single sign-on to the Grid resources that the user is entitled to use. `gd_certinfo`, `gd_proxyinfo`, `gd_proxyquery`, and `gd_destroyproxy` are associated utility commands which are used to query or invalidate the user's proxy certificate. The `gd_jobsubmit` command allows users to submit compute jobs to a GRAM job manager described by a Resource Specification Language

Table 1. SSS ted commands in the Geodise toolkit

Function Name	Description
yyyy	Stores a file in a repository with associated metadata
yyyy	Returns information about the user's certificate
yyyyys	Creates a Globus proxy certificate from the user's credentials
ysysyy	Destroys the local copy of the user's Globus proxy certificate
yyyy	Retrieves a file from a remote host using GridFTP
ssys	Terminates the GRAM job specified by a job handle
ssys	Queries the status of a Globus GRAM job until complete
ssyys	Returns the status of the GRAM job specified by a job handle
ssssy	Submits a GRAM job to a Globus server
ssss	Returns job handles for all GRAM jobs
yysyy	Returns information about the user's proxy certificate
yyssy	Queries whether a valid proxy certificate exists
yyyy	Transfers a file to a remote host using GridFTP
sys	Retrieves metadata about a file based on certain restrictions
yyyy	Retrieves a file from the repository to the local machine
syyy	Sends a text message to the specified mobile phone number

(RSL) [15] string. The eeeee command returns a unique job handle which identifies the job. The job handle may be used to query, poll, or terminate the status of the user's job by using eeeee , eeee , and eeee , respectively. In addition the eeee command may be used to query a Monitoring and Discovery Service (MDS) to return all the job handles associated with the user's certificate. Two file transfer commands, eeer and eeer , are provided to allow users to transfer files to and from Grid-enabled compute resources. These commands support the high performance file transfer protocol GridFTP [16]. A third-party type of file transfer command will be implemented in the future release of the Geodise toolkit.

Database related functions of the Geodise toolkit provide users with the ability to store files in a repository with associated metadata, query the metadata and retrieve the files, providing they have the correct access rights. The erer function will store a given file in a repository for an authenticated user. The function is able to generate a structure containing some standard metadata for the file, such as its local name, size, format, and creation time. The user may add additional metadata, for example comments, custom information specific to that format, and a list of users or groups who may access the file in the future. The function then transports the file to a server using GridFTP and also sends the metadata to a database accessed via a web service. The erer command returns a unique handle which can be used to retrieve the file at a later date. The metadata that is stored can be queried by an authorised user with the rrr command, in order to discover files that have certain characteristics and obtain information about them, such as their handle for retrieval. The reer function will locate a file based on a given file handle and

provides the user with the Matlab code which returns the file to a specified local directory.

For the event notification we use the widely popular Short Message Service (SMS) from mobile telecommunications technology. The Matlab command `gd_sendtext` can send a text message to the specified mobile phone number via a 'pay-per-use' service, independent of its geographical location, and inform the user about the state of the processes. The client and the server of the text messaging service run on Globus servers, hence, they both benefit from the security measures provided by the Globus toolkit. We also use Matlab's native function, `sendmail`, to e-mail the user simulation results by using an SMTP server.

3 Geodise Application Exemplar

To demonstrate the possible use of our Grid-enabled Matlab PSE we choose a basic problem of fluid dynamics, which is the two dimensional, external, laminar flow over a NACA four digit airfoil. A sketch and a sample solution of the problem are given in Fig. 1. At the velocity inlet the assumed free-stream velocity profile is constant, and the angle of attack is measured in the counter-clockwise direction to the horizontal. The upper and lower boundaries are periodic, and there is a pressure outlet on the right hand side of the computational domain. The airfoil profiles are generated by using standard NACA four digit expressions [17]. This problem can be solved using various CFD tools. The Geodise toolkit currently has two commercially available codes: Gambit and Fluent [18], for the mesh generation and solution processes, respectively. Nevertheless it is possible to integrate different codes as long as they can be run on the Globus server.

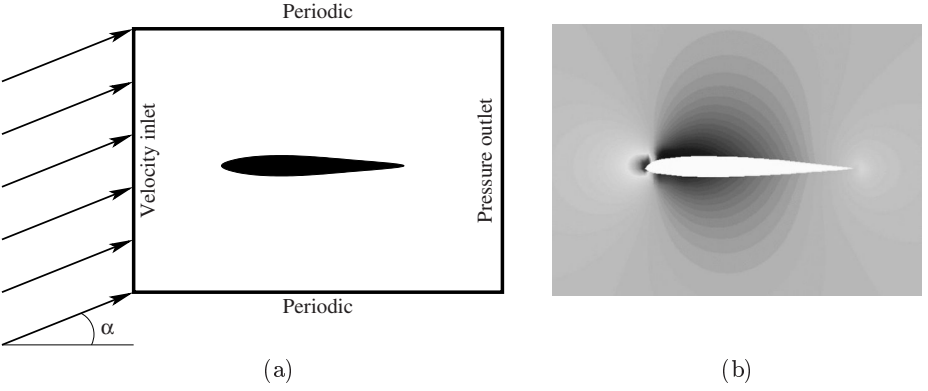


Fig. 1. (a) NACA four digit airfoil problem with boundary conditions. Here, α is the angle of attack. (b) Pressure contours visualized in Matlab

Since a valid Grid proxy is required to use the Grid-enabled resources, the user initiates their Grid proxy certificate by using `gd_createproxy` command.

This command invokes the Java CoG, which in turn pops a window where the user can enter their password. After the user enters their password and presses the "Create" button, a proxy certificate is generated for the user.

The next step involves the preparation of vertex data and journal files for Gambit and Fluent, and their transfer to the Globus server. The vertex data file for a NACA four digit airfoil is a text file containing the coordinates of the airfoil, which are used by the mesh generation tool Gambit. The journal files for Gambit and Fluent are tailored according to various input parameters entered by the user. The Gambit journal file informs Gambit to use the vertex file as the input file, to mesh the domain using a given mesh size parameter, and to export a Fluent compatible mesh file as output. Similarly, the Fluent journal file instructs that program to use the mesh file as input, to use inlet velocity and angle of attack parameters in the numerical solution, and to export a data file after the solution converges. When the journal files are ready, the user transfers them to the remote Globus server by using the `gd_putfile(<FQHN>,<Local file>,<Remote file>)` Matlab command. Here, `<FQHN>` is the fully qualified host name of the remote Globus server. A snapshot of Fluent journal file preparation and transfer is shown in Fig. 2.

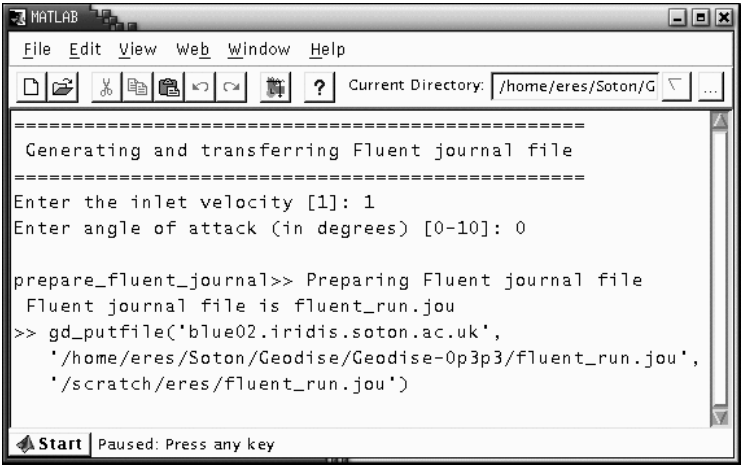
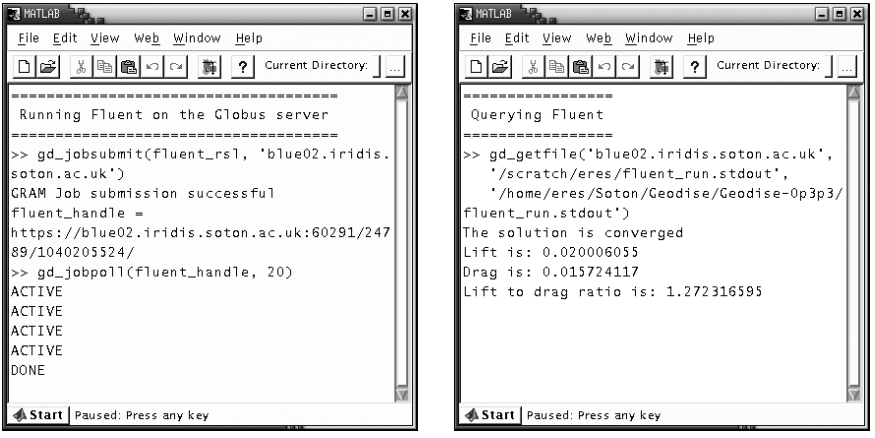


Fig. 2. Generating a Fluent journal file in Matlab environment, and transferring it to the Globus server. Here, the user inputs are the inlet velocity and the angle of attack

A properly generated mesh file is required by the analysis tool, and here the user must generate the mesh file by submitting the geometry to the Gambit mesh generation tool. The user then waits until Gambit finishes meshing, and the Globus server changes its status from "ACTIVE" to "DONE". Additionally, the user needs to make sure that the mesh generation process succeeds, and the quality of the generated mesh is acceptable for analysis. Therefore, before

running the analysis tool the state of the mesh generation and mesh quality are checked by transferring the standard error file produced by Gambit to the local file system and parsing the mesh quality information from that file. If the mesh generation step is satisfactory, the user can now submit the analysis job to the Globus server, get back a job handle, check the status of the job, and retrieve convergence information and objective function values by using a very similar process. Fig. 3 show the Matlab environment during these steps.



(a) (b)

Fig. 3. (a) Running Fluent on the Globus server by using a proper RSL string and previously generated journal file. The job status is polled every 20 seconds. (b) Transferring Fluent output file to the local file system, and parsing it to retrieve objective function values

Throughout these processes intermediate and solution files are archived in the Geodise repository, with the `gd_archive` command. By associating metadata with the files the design archive may be accessed interactively when required using `gd_query` command which runs a Web browser from the Matlab environment and gives access to archived files (see Fig. 4). Furthermore, the user could be notified about the progress of their simulation by text messages through a SMS. At the end of the simulation an e-mail message containing simulation results is also sent to the user. Ultimately this process would form one loop in an optimisation process.

4 Conclusions and Future Work

The Matlab environment along with the Java CoG [13] provides a flexible and robust user interface for Grid computing. By exposing the compute, data and notification features as toolkit components we are able to construct high level functions which utilise Grid resources for CFD and design search tasks. Given

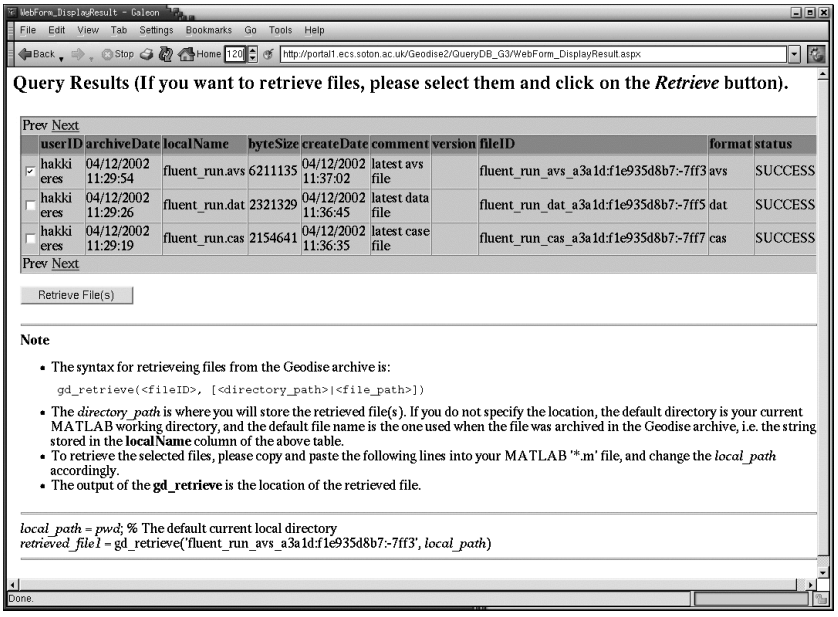


Fig. 4. The Web interface of Geodise toolkit showing query results

commands in a high level interpretive language it is straightforward for the engineer to exploit available Grid-enabled resources to tackle computationally and data intensive tasks.

Future work on this project will focus on the creation of high level application components. This work will include the exposure of heterogeneous legacy codes, the integration of various optimisation algorithms, and amalgamation of intelligent workflow composition and retrieval mechanisms to the Geodise PSE. Refining the existing compute components will involve adding client side tools to allow the user to discover compute resources. A future requirement for a fault tolerant data management system is the provision of a local personal metadata archive which replicates the data stored in the main repository. Data lifetime management is another issue, i.e., a mechanism is needed to specify how long a collection of data will be stored in Geodise repository, and to be able to extend the lifetime, or perform clean up tasks. The ability to send text messages from mobile devices to the Geodise PSE in order to control processes will also be implemented. In the near future the Geodise toolkit may also exploit the feature of sending binary images to third-generation mobile devices.

We expect that the architectures of the computational and database components will converge with a move to an Open Grid Services Architecture (OGSA) model. The implementation of OGSA defines a number of extensions to standard XML web services that provide the common functionality required by all the Geodise toolkit components.

Acknowledgements

This work is supported by UK EPSRC GR/R67705/01. The authors gratefully acknowledge many helpful discussions with the Geodise team, and researchers from the myGrid team (UK EPSRC GR/R67743/01). We thank Fluent, Microsoft, Level 9 Networks, Epistemics, and Intel for ongoing support.

References

1. Siddall, J.N.: Optimal Engineering Design: Principles and Applications. Marcel Dekker, Inc., New York and Basel (1982)
2. The Geodise Project. (2002) http://www.geodise.org/
3. Walker, D.W., Li, M., Rana, O.F., Shields, M.S., Huang, Y.: The software architecture of a distributed problem-solving environment. Concurrency: Practice and Experience **12(15)** (2000) 1455–1480
   http://www.cs.cf.ac.uk/User/David.W.Walker/papers/psearch01.ps
4. von Laszewski, G., Foster, I., Gawor, J., Lane, P., Rehn, N., Russell, M.: Designing grid-based problem solving environments and portals (2001)
   http://www-unix.mcs.anl.gov/~laszewsk/papers/cog-pse-final.ps
5. Abramson, D., Power, K., Kolter, L.: High performance parametric modelling with Nimrod/G: A killer application for the global grid. In: Proceedings of the International Parallel and Distributed Processing, Cancun, Mexico (2000) 520–528
   http://www.csse.monash.edu.au/~davida/papers/ipdps.ps.Z
6. Triana. (2002) http://www.triana.co.uk/
7. Cox, S.J.: Grid enabled optimisation and design search for engineering (GEODISE). NeSC Workshop on Applications and Testbeds on the Grid (2002)
8. Matlab 6.5. (2002) http://www.mathworks.com/
9. Casanova, H., Dongarra, J.: Netsolve: A network-enabled server for solving computational science problems. The International Journal of Supercomputer Applications and High Performance Computing (2000)
   http://citeseer.nj.nec.com/casanova00netsolve.html
10. The Globus Project. (2002) http://www.globus.org/
11. Global Grid Forum. (2002) http://www.gridforum.org/
12. Commodity Grid Kits. (2002) http://www.globus.org/cog/
13. von Laszewski, G., Foster, I., Gawor, J., Lane, P.: A Java commodity grid toolkit. Concurrency: Practice and Experience **13** (2001)
    http://www.globus.org/research/papers/vonLaszewski--cog-cpe-final.pdf
14. Verma, S., Parashar, M., Gawor, J., von Laszewski, G.: Design and implementation of a CORBA commodity grid kit (2002)
    http://www.caip.rutgers.edu/TASSL/Papers/corbacog-gcw01.pdf
15. The Globus Resource Specification Language (RSL) v1.0. (2002)
    http://www-fp.globus.org/gram/rsl_spec1.html
16. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S.: Secure, efficient data transport and replica management for high-performance data-intensive computing. IEEE Mass Storage Conference (2001)
    http://www.globus.org/research/papers/msc01.pdf
17. Abbott, I., von Doenhoff, A.: Theory of Wing Sections. Dover Publications, New York (1959)
18. Fluent. (2002) http://www.fluent.com/

GSiB: PSE Infrastructure for Dynamic Service-Oriented Grid Applications

Yan Huang

Department of Computer Science
Cardiff University
PO Box 916
Cardiff CF24 3XF
United Kingdom
Yan.Huang@cs.cardiff.ac.uk

Abstract. This paper describes the Grid-Service-in-a-Box(GSiB) system – a visual environment for both service providers and service clients who wish to deploy, manage, and use services in a Grid environment, without having to be experts in Grid and Web service technologies. GSiB provides a variety of easy-to-use graphical user interfaces for service providers to deploy, undeploy, update, compose, and monitor services, and for service clients to generate service-composite applications, and to track and monitor submitted jobs. In this paper, these interfaces are described and several implementation issues are addressed.

1 Overview

Building on Grid and Web service technologies, the Open Grid Services Architecture (OGSA) proposes a new generation of Grid infrastructure, in which services act as the building blocks of the Grid system. OGSA abstraction [1, 2] is quite a new concept and presents a lot of challenges for the scientists who wish to use it. In general, these scientists cannot be expected to keep abreast of the rapid development of Grid technologies, so easy-to-use tools and environments are needed to allow scientists to take full advantage of service-oriented Grids, such as those based on OGSA, without them having detailed knowledge of the underlying infrastructure. This is the main motivation for the Grid-Service-in-a-Box (GSiB) system. The GSiB project is developing a visual problem-solving environment for both service providers and service clients. For the service provider GSiB provides a visual environment for creating and publishing Grid services from existing software and libraries, browsing, monitoring and querying Grid services, and composing higher-level Grid services from other services. For service clients, GSiB presents a visual environment for browsing and querying Grid services and creating Grid applications by composing Grid services. It also provides an execution environment for running Grid applications.

GSiB is designed to function in conjunction with OGSA-compliant Grid systems, as well as with other similar service-oriented architectures. In this context we refer generically to a Grid service as any service that is consistent with this

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 430–439, 2003.
© Springer-Verlag Berlin Heidelberg 2003

architecture. We assume that all Grid resources are accessible as services, and that there are standard mechanisms for service publication, service querying and service invocation. In [5] and [4], we have explored techniques for building such a system and a brief description of these will be given in section 3.

2 GSiB Interfaces and Functionality

GSiB provides an easy-to-use graphical user interface for service providers and clients that allows users to take advantage of Grid services even though they may have little knowledge of Grid computing and Web services. The details of the underlying technologies are largely hidden within the GSiB system. GSiB consists of two packages: the Service Provider GUI (SP-GUI) package, and the Service Client GUI (SC-GUI) package. The interfaces and functionalities of these opackages are discussed in the following subsections:

2.1 The Service Provider GUI Package

The SP-GUI package contains a variety of graphical interfaces for service providers to easily and quickly create Grid services from existing legacy software and libraries, to update, monitor and manage Grid services, and to compose more complex Grid services based on existing Grid services. GSiB provides mechanisms for supporting the updating of Grid services and fault-tolerant lifecycle management. The SP-GUI package contains the following graphical interfaces and components:

- *Login Interface* for users to log in as authorized service providers.
- *Service Deployment/Undeployment Interface* for deployment and undeployment of services. This provides the following functionality:
  - Automatic deployment of existing software routines or libraries as OGSA-compliant Grid services. Java, C, and C++ are supported as the native languages of the service deployed by this mechanism.
  - Visual Service Composition Environment (VSCE) for service providers for building and deploying a service composed of existing services by visually drawing a diagram representing the workflow of the composite service.
  - Updating of existing Grid services. When a service is updated, its implementation or/and interface may be changed, so it is important to consider how to avoid interrupting running applications and other high-level services that are built on top of the updated services.
  - Undeployment of an existing Grid service. This also involves consideration of how to avoid interrupting a running application and disrupting high-level services.
- *Service Configuration Interface* for setting up or changing a service's security policy and lifetime.
- *Service Monitoring Interface* for dynamically monitoring the distribution and workload of Grid services.

– Other interfaces include an *Authorization Interface* for a user to log in as an authorized service administrator; an *Authentication Interface* for setting up a Grid service so that it is accessible by only authorized users for a required or assigned time; and, a *Lifecycle Management Interface* for dealing with any failures, and reclaiming service and state associated with failed operations.

2.2 The Service Client GUI Package

The SC-GUI package presents a variety of graphical interfaces for a service client to browse and query Grid services, create Grid applications, submit jobs, monitor the execution of jobs, and retrieve results. The SC-GUI package contains the following interfaces and components:

– *Login Interface* for users to log in as authorized service users.
– The *Visual Service Composition Environment* for service clients is similar to the VSCE for service providers, except that it produces a composite Grid application rather than a composite higher-level Grid service. A VSCE allows a user to create a representation of a Grid application by drawing its workflow. The graphical model created using VSCE can be transformed into an XML-based workflow description, and can also be sent to a Workflow Engine to be executed. This is discussed further in Section 4.
– The *Workflow Engine Component* provides an execution environment for Grid applications that are described in an XML-based workflow description language.
– The *Job Tracking and Monitoring Interface* allows users to track and monitor the jobs they have submitted.
– A *Service Browser* for browsing and querying services.

3 Implementation Aspects

3.1 Service Deployment

Usually additional code, such as a wrapper, is needed to make a piece of software available as a Web service. This can result in a lot of repeated work, especially when a large library of routines is to be wrapped as Web services. A legacy problem exists when the implementation language of the services and the language supported by the service hosting environments are different. Thus, service deployment is one of the main issues addressed by the SP-GUI package.

As mentioned above, we assume that Grid services are Web service compatible, which means that they are accessible by general Web service users regardless of how they are implemented. The public interfaces and bindings of a Web service are described in an XML document, and it is registered in a service registry to allow service discovery and querying [8]. The SP-GUI package can be used by service providers to publish existing software and libraries as Web services. Although Java is used to build the system framework, the implementation language of the services supported by the SP-GUI is not limited to Java – it can

be Java, C, or C++. In the implementation of the SP-GUI package, a Java-C Automatic Wrapper (JACAW) is used to automatically generate the Java JNI wrapping code for the legacy software [3]. This JNI wrapper provides a Java interface for the wrapped legacy code and invokes the original routine which is compiled into a dynamical library. Thus, the wrapped legacy code can be invoked and run in the same way as any other Java code. To deploy a Java program as a Web service the Axis Java2WSDL utility is used to generate the Web Service Description Language (WSDL) document from the Java code. The service is then published into a UDDI (Universal Description, Discovery and Integration) registry server by using UDDI4J which provides an Java API for interaction with a UDDI registry [7]. Figure 1 illustrates this service deployment scenario.

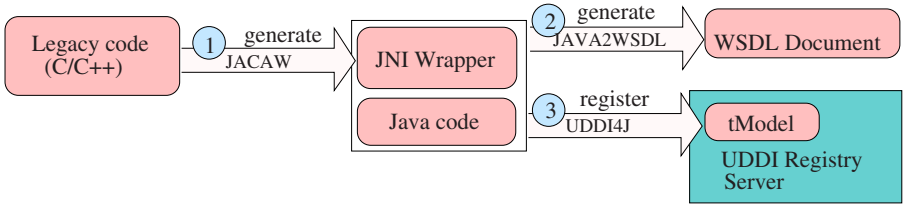


Fig. 1. Service deployment in the SP-GUI package.

3.2 Service Composition

It is useful to be able to build new services and applications through the composition of existing services, and this task can be made easier by using visual programming tools. GSiB provides a Visual Service Composition Environment (VSCE) for both service and application composition that allows a user to create a new composite service or application by drawing its workflow graph. The VSCE lists all the known service instances matching some set of requirements specified by the user, and allows the user to choose from these based on dynamically discovered metrics such as speed, workload, and network latency of the service instances[1].

In the workflow representation, or flow model, an *activity* represents a task to be performed as a single step within a process, and is implemented by one of the operations provided by a Web service. There are three kinds of activities: normal activity, assignment activity and control activity. A normal activity is implemented by an operation of a Web service; an assignment activity specifies an assignment normally between a variable and the output of an activity; a control activity specifies a control process such as a loop or conditional construct. The workflow representation is a directed graph, similar to a task graph, and

[1] In the future this task will be done automatically by a resource manager.

the activities correspond to nodes in the graph. Activities are wired together through *control links*. Each control link is a directed edge of the graph, and together they represent the control flow of the flow model. A *data link* is a second kind of directed edge of the graph and specifies the flow of data between a source activity and a target activity. A *data map* that describes how a field in the message part of the target's input message is constructed from a field part of the source's output message is defined for each data link. A flow model itself also has input and output: a *flow source* is a source of the data links targeting activities in the flow model, and a *flow sink* is a target of the data links of the flow model. Figure 2 shows the graphical symbols used in displaying a flow model.

Graphic Symbol	JSFL Symbol
	Data input port
	Data output port
	Control input port
	Control output port
	Data Link
	Control Link
	Nomal and assign activity
	Control activity

Fig. 2. Graphical symbols for constructing a flow model.

Figure 3 gives an example of the directed graph of a job flow model, together with the corresponding Java code.

After the graphical flow model representing an application has been created using VSCE, GSiB can convert it into an XML description document, and then (if the user requests it) the application can be submitted for execution by passing its this XML document to a workflow engine, as described in Section 5.

3.3 Service Workflow Language

One of the main aims of GSiB is to provide a visual composition and editing tool to create and submit service-based applications, described in an XML document and based on a standard job description language. Figure 4 shows how a user can create and run a service-based application through a drag-and-drop interface. In this scenario, the workflow engine is a service that accepts an XML document

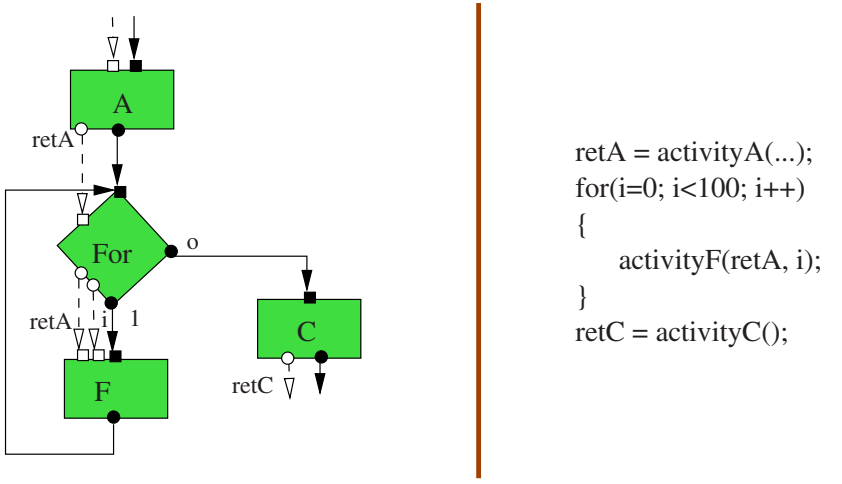


Fig. 3. The graphical workflow model of a *for* loop example.

describing an application, automatically produces a corresponding Java code, compiles the code, and executes it.

In building a Visual Service Composition Environment, choosing and defining a job description language is one of the critical issues. A VSCE must be able to convert a graphical job flow model into an XML-based description document, and then from this XML description document into a distributed Java code. It would be more convenient if two-way conversion is available because, for example, after saving a job as a XML document, the user might want to modify the job again. Thus, the mapping between the XML job description language and job's graphical flow model, and the mapping between the XML job description language and the Java program are the main requirements in defining a job description language. Therefore, an XML job description language that allows documents to be readily converted to both a graphical representation and a Java program meets our requirements.

Web Service Flow Language (WSFL) warrants attention because it can describe a composite job in a graphical form. Its activities can be mapped into nodes of the graph, and its data links and control links can be mapped onto directed edges of different kinds. However WSFL has its limitations, such as a lack of a one-to-one mapping between Java programming constructs and WSFL processes, inflexibility in how the variables that are used as parameters in the boolean expression of the transition and exit conditions are specified, and difficulties in displaying WSFL control processes in a clear and understandable graphical form [5]. We have, therefore, extended WSFL to create the Service Workflow Language (SWFL), which overcomes the limitations of WSFL and

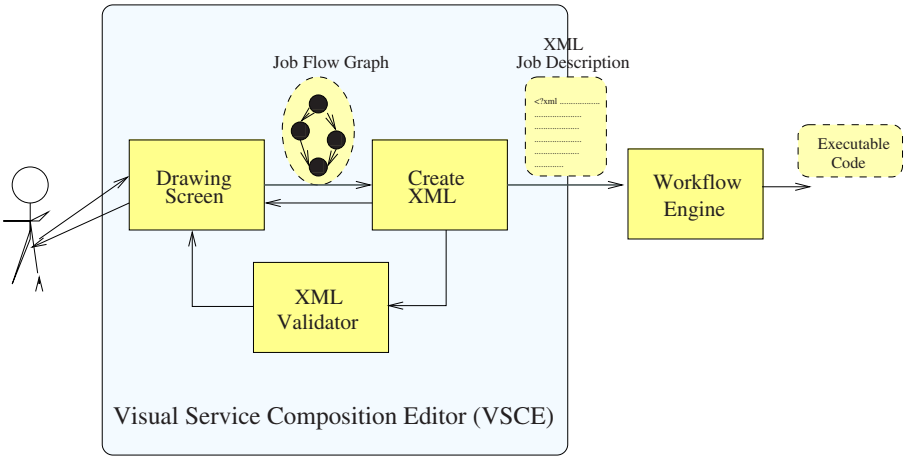


Fig. 4. Visual service composition.

meets the above requirements for a job description language. The details of SWFL is available in [5].

4 SWFL2Java

We have developed a tool, called `SWFL2Java`, that converts the description of a job in SWFL into executable Java code. The details of `SWFL2Java` are discussed elsewhere [5], however, we will give here an overview of the implementation. In `SWFL2Java` an SWFL document is not translated directly into a Java program but is stored in an intermediate form as a Java *FlowModel* object. This is made up of two Java *Graph* objects: *DataGraph* and *ControlGraph*. The former stores the data flow structure of the flow model, and the latter stores its control flow structure. One reason for storing the job description in this intermediate form is to be able to interact readily with Java-based graphical tools, such as VSCE, for the visual composition of Web services. The graph created with VSCE can be stored as a *FlowModel* object and converted to and from SWFL, as well as into a Java program. Another reason for using the *FlowModel* form is to reduce the overhead when the same job is used many times and scheduled on different resources. In such cases it is easier to generate the Java code from the intermediate form rather than starting from the original SWFL. Figure 5 illustrates the importance of the *FlowModel* representation of a composite application.

5 Workflow Engine

After it is generated by VSCE, a composite service-based application may then be submitted to a Workflow Engine to be executed. A Workflow Engine provides

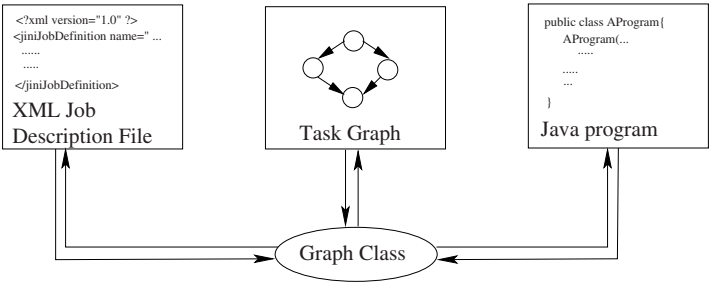


Fig. 5. Using a *FlowModel* object as an intermediate representation in `SWFL2Java`.

an execution environment for such applications described in an SWFL document. We have implemented such a Workflow Engine as part of our Jini-based Service Grid Architecture (JISGA) citeThesis.

In JISGA, the *WorkflowEngine* service supports both blocking and non-blocking submission of applications, as well as their sequential and parallel processing. For blocking sequential jobs, the Workflow Engine processes the job by generating the Java harness code using *SWFL2Java*, executing it, and sending the result to the job submitter. In the case of non-blocking and/or parallel jobs, Jini's JavaSpaces plays a key role in the job processing.

JavaSpaces provides object storage services through a tuple-space abstraction, and supports the sharing of objects between distributed applications [6]. In JISGA, JavaSpaces is used in the following ways:

1. As a shared memory, allowing message passing between different services and processes.
2. To store the *result* objects of non-blocking jobs so that job submitters can retrieve their job results later.
3. To store *message* objects used in communication between sub-job processes, thereby allowing job parallel processing.
4. In maintaining the two job queues of the JISGA system so that jobs and sub-jobs can be processed in an FIFO (First In First Out) order.

The JISGA environment makes use of two job queues. One is the Parallel-Job Queue, containing parallel jobs to be processed. The other is the Sub-Job Queue, containing sub-jobs generated from the partitioning of parallel jobs. The *JSHousekeeper* performs housekeeping tasks in JavaSpaces serving in the system. Its duty is to collect *result* objects, update job queues, and maintain data messages. It is also a service, and provides an interface for job submitters to retrieve their job results.

The *WorkflowEngine* and *JobProcessor* services both deal with the processing of parallel jobs. After the parallel job is submitted, the *WorkflowEngine* service will leave the job for the *JobProcessor* service to process by adding the job to the Parallel-Job Queue. A *JobProcessor* is a service working internally in the

environment supported by a number of workflow engine services. Its main duty is to process parallel jobs, which includes partitioning a parallel job into sub-jobs, adding these sub-jobs to the Sub-Job Queue, and processing sub-jobs in the Sub-Job Queue. There are two threads running synchronously in a *JobProcessor*. One of the threads takes one job at a time from the Parallel-Job Queue and then partitions the job into sub-jobs which are then added to the Sub-Job Queue. Another thread takes one sub-job a time from the Sub-Job Queue and processes it. As many as *WorkflowEngine* and *JobProcessor* services as deemed necessary may exist in the environment they support. It is up to the system administrator to determine the number of *WorkflowEngine* services and *JobProcessor* processes. However, more than one *JobProcessor* processes are needed to allow the processing of parallel jobs.

6 Concluding Remarks

The main motivation for developing GSiB was to provide a user-friendly visual PSE to assist computational scientists to build and run composite service-based applications. A variety of visual interfaces are supported in GSiB to achieve this aim. The GSiB interfaces are divided into two packages: Service Provider GUI (SP-GUI) and Service Client GUI (SC-GUI). The SP-GUI package supports deploying, undeploying updating and monitoring services,as well as service composition. The SC-GUI package supports the creation of composite service-based applications, job tracking and monitoring, and service advertising and subscription.

Several issues in the implementation of the underlying infrastructure have been addressed in the paper. The legacy problem in deploying existing software as services is solved by using JACAW to automatically generate JNI wrappers for C legacy codes. By using VSCE users can generate composite services and applications through a simple "drag-and-drop" interface. The graphical representation of the flow model of a composite service-based application has been discussed, and SWFL, an XML-based language for describing such applications has been introduced. SWFL is an extension of WSFL that can describe Java-like conditional and loop constructs, permits sequences of more than one service within conditional clauses and loop bodies, and overcomes limitations inherent in WSFL's data mapping approach. Given a SWFL job description the `SWFL2Java` tool can generate a representation of the corresponding data and control link structure in the form of a Java `FlowModel` object. An application created with VSCE and described in SWFL can be submitted to a Workflow Engine that provides an execution environment for such applications. By using `SWFL2Java`, code based on the SWFL document can be dynamically generated and run. The Workflow Engine supports blocking and non-blocking job submission, and provides both sequential and parallel job processing.

References

1. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed System Integration," http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf.
2. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, and P. Vanderbilt, "Grid Service Specification," http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-04_2002-10-04.pdf.
3. Y. Huang, I. Taylor, D. Walker, and R. Davies, "Wrapping Legacy Codes for Grid-Based Applications," in Proceedings of the Fifth International Workshop on Java for Parallel and Distributed Computing, held as part of the International Parallel and Distributed Processing Symposium (IPDPS) in Nice, France, April 2003.
4. Y. Huang and D. Walker, "JSFL: An Extension to WSFL for Composing Web Services," UK e-science All Hands Meeting, 2-4 September 2002.
5. Y. Huang, "The Role of Jini in a Service-Oriented Architecture for Grid Computing," Ph.D Thesis, Department of Computer Science, Cardiff University, January, 2003
6. W. Keith Edwards and Tom Rodden, "Jini Example By Example," Sun Microsystems Press, 2001
7. "UDDI4J Overview," http://oss.software.ibm.com/developerworks/opensource/uddi4j/
8. "Web Services Architecture Requirements," http://www.w3.org/TR/wsa-reqs, April, 2002

Partial Dual Unicasting Based Handoff for Real-Time Traffic in MIPv6 Networks*

DaeKyu Choi and Hyunseung Choo

School of Information and Communication Engineering
Sungkyunkwan University
440-746, Suwon, KOREA +82-31-290-7145
{eunpiri, choo}@ece.skku.ac.kr

Abstract. Recently QoS guaranteed real-time data support with user mobility in wireless Internet becomes more significant due to drastic increase of network users called netizens. Mobile IP(MIP) supporting user mobility evolves from versions v4 to v6 with comprehensive analysis of mobile users. However, Mobile IPv6 still has the buffering problem and problems occurred by the zigzag effect. In this paper, we propose a new MIPv6 handoff mechanism called partial dual unicasting(PDU) based on QoS supported RSVP signaling protocol. This scheme reduces the handoff failure rate up to about 24% and guarantees the time for the handoff completion. Furthermore, it resolves the ineffectiveness of path usage in dual unicasting(DU).

1 Introduction

The number of Internet users has been drastically increasing. Since the late 1990s, network users access various kinds of services such as e-mails, chattings, and multimedia applications in the wired life. Nowadays current users demand the similar services in the wireless mobile environment and even they want QoS guaranteed real-time data. Therefore, a mobile enabling protocol, Mobile IP(MIP)[1], is combined with Resource ReSerVation Protocol(RSVP)[2] as an active approach to support the enhanced user mobility.

The earlier version of MIP, Mobile IPv4(MIPv4), has various problems. Basically it employs 32-bit address hierarchy that is expected the lack of address space in the near future. It will become more serious with the popular use of mobile devices such as laptops and PDAs. There is a triangle routing problem in path establishment which creates not only an inefficient path in data transmission but also traffic bottleneck in home agent(HA). As we all know, sender-receiver communication is terminated when HA is failed.

In Mobile IPv6(MIPv6)[3], a mobile node(MN) sends a binding update(BU) to both correspondent node(CN) and HA when it moves to other networks. It takes some time to send data to the changed care of address(CoA) after CN receives it. If CN transmits data to MN during this time, data are lost. To

* Dr. Choo is the corresponding author.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 443–452, 2003.
© Springer-Verlag Berlin Heidelberg 2003

prevent this in MIPv6, MN sends a BU to the previous old access router(oAR) when MN moves to other networks. Then the oAR knows the movement of MN through this message, and transmits to the new AR(nAR) packets for MN. As a problem of this scheme, oAR must save packets arrived for MN until receiving BU message from nAR. Increase of the number of MNs leads to the required amount of buffers to be fairly increased. Therefore data loss occurs when buffers in AR are not enough to hold data for MNs in handoff. This problem becomes more significant when MN density is high. While MN moves into other networks, it is possible to be in ping-ponging state in a boundary area. Ping-ponging occurs when there is not much difference in the strength of signal power between two networks and thus MN registration to someone is ambiguous at the moment. According to the increased handoff messages on ping-ponging, two corresponding ARs should handle a certain amount of operations. The series of processing procedures burdens ARs and thus networks therefore the probability of handoff failure increases.

In this paper, we propose a fast handoff scheme called partial dual unicasting(PDU) based on QoS supporting RSVP which solves buffering problem in ARs and also ping-ponging problem in networks. PDU scheme concurrently maintains CoAs of old and new networks in the binding cache of CN. The proposed scheme provides the longer handoff execution time on MNs due to the relative increase of handoff area, and thus handoff failure rate is decreased up to about 24%. Likewise it solves the inefficiency of dual unicasting(DU) on path use and enables to eliminate effectively the ping-ponging state.

This paper is organized as follows. In section 2, related works, trends, and problems in handoff are discussed. The new scheme is presented in section 3 to improve the current method. In section 4 the proposed PDU scheme is evaluated. Finally, we conclude our paper with the future direction.

2 Related Works

In MIPv4, when CN sends a packet to MN, it passes through the following procedure. When MN is located at home network, CN sends data to HA that are relayed to MN. As MN moves into foreign networks, HA intercepts data which CN sends to MN and encapsulates its packet, and finally sends it to CoA of MN. Foreign agent(FA) of MN accepts and sends it to MN after the decapsulation process. On the other hand, MN sends data directly to CN without passing through additional agents and it is known as a triangle routing. After the triangle routing is established, the time for packet transmission delay becomes longer and the route is used inefficiently. Also packets must pass the HA, thus the HA becomes an obvious bottleneck in data transmission. Many studies[4,5] are done for solving the triangle routing problem. For this, the route optimization scheme is proposed in [5]. Although this scheme provides a solution for the routing path inefficiency, it still remains unsolved problems. First, buffering is required in FAs. Secondly, the extra reordering process is necessary when MN receives the buffered packets. Finally, IPv4 cannot handle security issues by itself.

In MIPv6, MN typically acquires its CoA through stateless or stateful (e.g., DHCPv6) address autoconfiguration based on the methods in IPv6 Neighbor Discovery. CoA for MN is changed when MN moves to other networks. When CoA is changed, MN sends BU message to its HA and all CNs. If MN demands an acknowledgement from CN for this, CN responds to MN by using the binding acknowledgement option. CN avoids the problem of the triangle routing by maintaining binding update of MN whenever MN moves. Even though MIPv6 improves MIPv4 in triangle routing, it still keeps some basic problems in MIPv4.

IETF Int-Service(Int-Serv) working group(WG) has researched on IP packet transmission methods based on the new service model which supports the real-time service demand for QoS [6]. There is an Int-Serv model proposed by the WG on real-time applications which divides services due to the flow unit, and thus QoS guaranteed services and unguaranteed services on relaying packets. The guaranteed employ RSVP signaling protocol to reserve the resource for the transmission delay. The paper [10] proposes MRSVP for RSVP adopted in mobile environments. For the QoS support in RSVP, MRSVP keeps the state in each protocol supported router. Using the state information stored, MN sends additional BU(ABU) message to determine merged AR(MAR) in the proposed scheme.

3 The Proposed Scheme

As shown in Fig. 1, we consider the case that MN moves from Network A to Network B. MN receives the radio advertisement for the first time from AR of Network B when it reaches P1. The advertisement message contains CoA2 value which is used when MN moves to Network B.

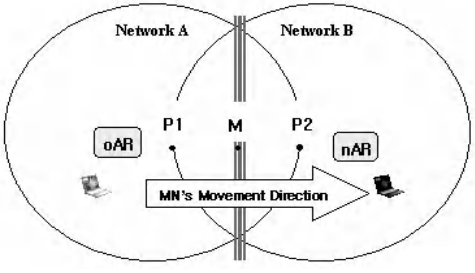


Fig. 1. Handoffs by movement of MN

As soon as MN receives the advertisement from AR of Network B, it generates a new BU message based on the value of CoA2. This message should be distinguished from the previously defined BU information and thus called as additional binding update(ABU). MN sends this message at the same time to both nAR and oAR, they deliver ABU message through the routing path. Routers supporting QoS protocols (MRSVP, RSVP), save a certain state. An

ABU message which is delivered at the same time to oAR and nAR follows the two corresponding routing paths, respectively. This message records the movements of each message in QoS supported router. The first router which takes two ABU messages becomes {*merged access router*}(MAR) on those routing paths. All packets sent from CN are made into two copies in MAR to transfer to nAR and oAR each. In other words, the original packets are delivered to oAR, and the copied that are made based on the routing extension header are transferred to nAR. We called this scheme as {*Partial Dual Unicasting*}(PDU).

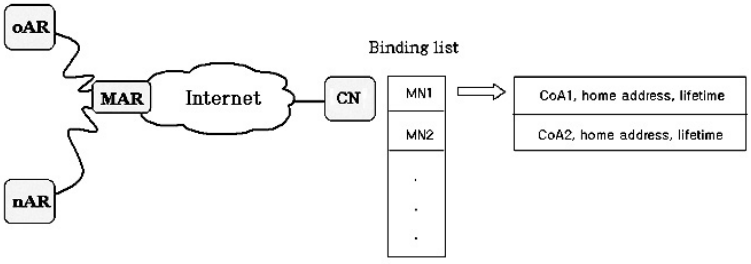


Fig. 2. CN binding list after receiving the ABU message

The PDU scheme is technically implemented as follows. CN should handle an ABU messages differently comparing to a BU. In general, CN which takes the BU message replaces the existing BU to the new one. Meanwhile, CN which takes an ABU message does not delete the field for CoA1 but rather it adds a new field for CoA2. Due to the completion of appending a new field to have CoA2 value, the binding list is updated as shown in Fig. 2. CN records the information to be added for CoA2 and delivers the packet eventually to MN through the addresses of CoA1 and CoA2 by using the routing extension header of IPv6. The packets made in this manner are delivered to MAR and the MAR checks the routing extension header of the packet to figure out that they must be delivered to both oAR and nAR. MAR decides those packets must be sent to both oAR and nAR, and generates two packages, one for oAR and the other for nAR. The generation method makes those packets go to the oAR with the information at the typical header and to the nAR based on the information at the routing extension header. This process eliminates the message duplication in the path from CN to MAR, and thus routing resources are more effectively.

Each message in Fig. 3 is described as follows. (1) MN arrives at P1. (2) MN takes the message of nAR(advertisement), and finds out new CoA(CoA2). (3) ABU is created. (4) It is transmitted to both oAR and nAR at the same time. (5) Both oAR and nAR send it to MAR. (6) MAR transmits it to CN. (7) CN gets the message and decides if it is either ABU or BU. (8) CN updates the ABU list. (9) ABU acknowledgement(ABUA) is created. (10) CN sends ABUA to MAR. (11) MAR transmits ABUA to both oAR and nAR. (12) oAR or nAR sends

data packets to MN. (13) MN maintains the link to oAR until MN completely moves to nAR.

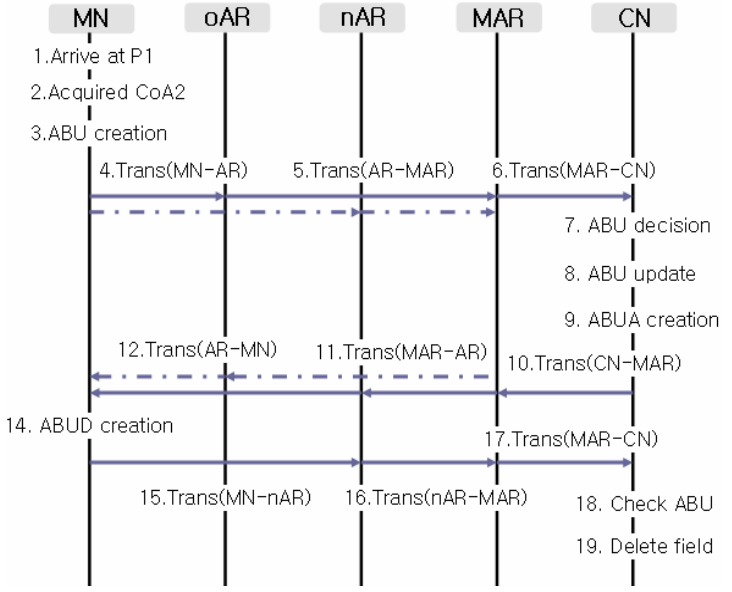


Fig. 3. Flow of total message

(14) When MN passes over the point P2(Fig. 1), it makes ABU deletion message(ABUD). (15) ABUD is sent to nAR. (16) The nAR relays it to MAR. (17) MAR sends it to CN. (18) CN checks the ABU list for message deletion. (19) CN eliminates the ABU field.

4 Performance Evaluation

4.1 Modeling

At each step the time to send a message is composed of transmission time, propagation time, and processing time, i.e. $M^i = \alpha^i + \beta^i + \gamma^i$. In here, i represents the step i. The transmission time α^i is computed by the size of the control message in bits over the bit rate of the link on which the message is sent, i.e. $\alpha^i = \frac{M_b}{B}$, where M_b is a control message assuming the fixed size. The B is the bandwidth of the link, B_l for the wired line, and B_w for the wireless case. The propagation time β^i varies depending on the transmission medium, i.e. β_l is the time for the wired line, and β_w is for the wireless one. The processing time γ^i has the same value at intermediate routers, MN, CN, oAR, and nAR. The wired medium is more stable than the wireless one, so the retransmission is not needed. Therefore, the physical transmission time(T^i) is represented by

$M^i(= M_l)$. Later in each step the message processing time on the wired and wireless cases are represented as M_l and M_w, respectively. At the wireless link, the message retransmission is necessary because a message can be lost in any moment. MN retransmits the message when lost in the air transmission. By considering the number of link failures (N_{fail}) and the probability of link failure, we obtain the additional signal processing time at these steps in the wireless case, i.e. $T_w^i = \sum_{N_{fail}=0}^{\infty}\{\tilde{T}_w^i(N_{fail}) \cdot Prob(N_{fail}\ failures\ and\ 1\ success)\}$. Whenever ACK signal may not be received for T_{out} after the request signal is sent, we assume that the message is lost and the control signal is retransmitted. If there are N_{fail} failures, then T_{out} and the message retransmission occur N_{fail} times. So $\tilde{T}_w^i(N_{fail})$ is induced as $\tilde{T}_w^i(N_{fail}) = M_w + N_{fail} \cdot (T_{out} + M_w)$. Therefore signal processing time for retransmission steps becomes

$$T_w^i = \sum_{N_{fail}=0}^{\infty}\{M_w + N_{fail}(T_{out} + M_w)\} \cdot Prob(N_{fail}\ failures\ and\ 1\ success)$$
$$= M_w + (T_{out} + M_w)\sum_{N_{fail}=0}^{\infty} N_{fail} \cdot Prob(N_{fail}\ failures\ and\ 1\ success) \quad (1)$$

Here $\sum_{N_{fail}=0}^{\infty}\{N_{fail} \cdot Prob(N_{fail}\ failures\ and\ 1\ success)\}$ is obtained by the infinite geometric progression. Usually, link failure probability q is smaller than 1. Therefore, $\sum_{N_{fail}=0}^{\infty}\{N_{fail} \cdot Prob(N_{fail}\ failures\ and\ 1\ success)\} = \frac{q}{1-q}$. Generally, q has the value of 0.5. So (1) becomes $M_w + (T_{out} + M_w)\frac{0.5}{0.5} = 2M_w + T_{out}$. In the proposed scheme, the additional message processing time is required. It is assumed to be the message processing time T_{proc}.

4.2 Total Handoff Time

The system parameters used to analyze the system are listed in Table 1. Each value is defined based on [7,8,9]. Fig. 3 represents the flows of messages. Based on the scheme in this figure, we compute the total handoff time. Handoff completion time is acquired by summing I, II, III, and IV below.

I. Sum of the processing time(SPT)

The processing time is required in steps 2, 3, 7, 9, 14, and 18. So, $SPT = T_{proc}^2 + T_{proc}^3 + T_{proc}^7 + T_{proc}^9 + T_{proc}^{14} + T_{proc}^{18}$. In the above case T_{proc}^i has a fixed value(T_{proc}) at each step which is just as much as the processing time. Therefore, $SPT = 6T_{proc}$.

II. The total time for message transmission in wired links(SMT_l)

Message transmission in the wired line states are in steps 5, 6, 10, 11, 16, and 17. In these cases, the total time for message transmission is

$$SPT_l = min\{T_l^5 \cdot H_{MAR,oAR}, T_l^5 \cdot H_{MAR,nAR}\} + T_l^6 \cdot H_{MAR,CN}$$
$$+ T_l^{10} \cdot H_{MAR,CN} + min\{T_l^{11} \cdot H_{MAR,oAR}, T_l^{11} \cdot H_{MAR,nAR}\}$$
$$+ T_l^{16} \cdot H_{MAR,nAR} + T_l^{17} \cdot H_{MAR,CN} \quad (2)$$

We assume that each T_l^i $(i = 5, 6, 10, 11, 16, 17)$ has a fixed value(T_l), and thus

$$SPT_l = 2min\{T_l \cdot H_{MAR,oAR}, T_l \cdot H_{MAR,nAR}\} + T_l\{3H_{MAR,CN} + H_{MAR,nAR}\}. \quad (3)$$

Table 1. System parameters.

Variables	Definitions	Values
T_{proc}	extra processing time	0.5 msec
B_l	bits rate of wired link	155 Mbps
B_w	bits rate of wireless link	144 Kbps
M_b	a control message size	50 bytes
β_l	wired link message propagation time	0.5 msec
β_w	wireless link message propagation time	2 msec
γ	message processing time	0.5 msec
T_{out}	message loss judgment time	2 msec
T_{but}	binding update time	5 msec
q	probability of message loss	0.5
$H_{a,b}$	number of hops between a and b	

III. Sum of transmission time in wireless links(SMT_w)
Message transmissions are in steps 4, 12, and 15 in wireless links. Here message transmission time is $SMT_w = T_w^4 + T_w^{12} + T_w^{15}$. From the equation T_w^i, SMT_w becomes $SMT_w = 3T_w(T_w = 2M_w + T_{out}) = 3(2M_w + T_{out})$.
IV. Sum of binding update time$(SBUT)$
The binding update time T_{but} is a system variable and given before the computation. $SBUT = T_{but}^8 + T_{but}^{19} = 2T_{but}$.

Therefore, we obtain the total required time for the handoff completion by the summation from step I to step IV.

$$\begin{aligned} T_{req} &= SPT + SMT_l + SMT_w + SBUT \\ &= 6T_{proc} + 2min\{T_l \cdot H_{MAR,oAR}, T_l \cdot H_{MAR,nAR}\} \\ &\quad + T_l\{3H_{MAR,CN} + H_{MAR,nAR}\} + 3(2M_w + T_{out}) + 2T_{but} \end{aligned} \quad (4)$$

4.3 Handoff Failure Rate

The T is a random variable of the time for MN staying in the overlapped area and the T_{req} is the time required for the handoff completion. Hence, the handoff failure rate is represented as $P = Prob(T < T_{req})$, where we assume that T is exponentially distributed. Thus,

$$P = Prob(T < T_{req}) = 1 - \exp(-\lambda T_{req}) < P_f \quad (5)$$

Here is λ the arrival rate of MN into the boundary cell and its movement direction is uniformly distributed on the interval $[0, 2\pi)$. So λ is calculated by the equation $\lambda = \frac{VL}{\pi S}$[11]. Here V is the expected velocity for MN that varies in the given environment and L is the length of the boundary at the overlapped area assuming a circle with radius l, i.e. $L = \frac{1}{6}2\pi l2 = \frac{2}{3}\pi l$. The area of the overlapped space S is $S = 2(\frac{1}{6}(\pi l^2 - \frac{\sqrt{3}}{4}l^2))$. Therefore, λ is induced by equations regarding as l. Hence we get the handoff failure rate by T_{req} in (5) and λ above.

$$l > \frac{4VT_{req}}{(2\pi - 3\sqrt{3})log(1/(1 - P_f))} \tag{6}$$

Similarly, when the desired handoff failure rate is given, then V is calculated. Thus,

$$\frac{l(2\pi - 3\sqrt{3})log(l/(1 - P_f))}{4T_{req}} > V \tag{7}$$

Consequently, the desired maximum velocity is acquired.

4.4 Experimental Results

Fig. 4 compares results for the probability of handoff failure between the existing scheme and the proposed one. It is obtained by using system parameters given in Table 1.

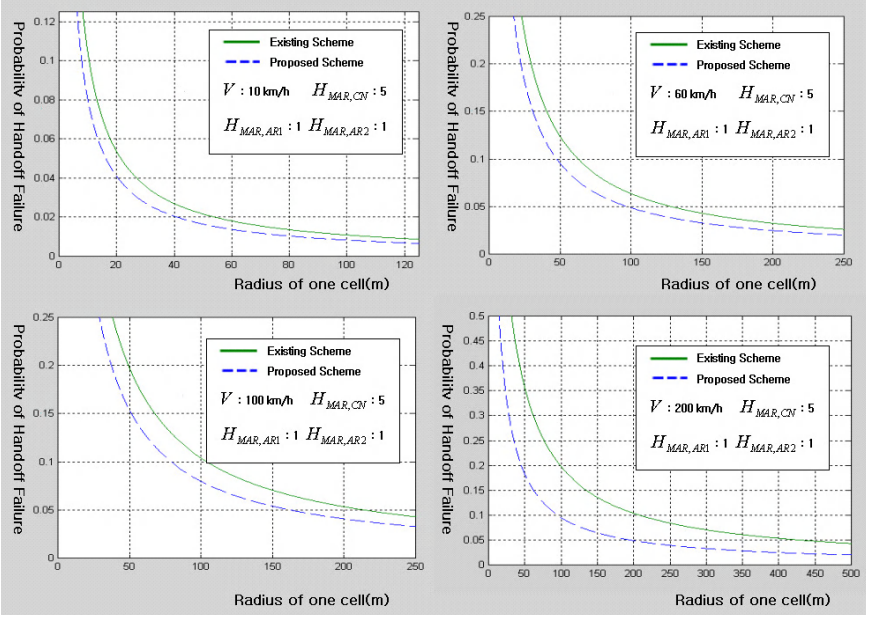


Fig. 4. By the increment of V

The probability of handoff failure is influenced by few factors that are the velocity of MN, hop count($H_{MAR,CN}$) from MAR to CN, hop count($H_{MAR,oAR}$) from MAR to oAR, hop count($H_{MAR,nAR}$) from MAR to nAR, and the radius of a cell(l). The increase of MN velocity V means the handoff should be completed within relatively short period of time. If MN moves faster than the regular speed, handoff time may not be sufficient and consequently the handoff is failed. In Fig. 4, for the various V, we investigate the probability of handoff failure in percentage with the cell radius 100m, and they are 1.09, 6.37, 10.39, and 19.71%. On the other hand, in our proposed scheme, they are 0.83, 4.87, 7.99, and 15.34%. The proposed scheme shows the relatively low handoff failure rate comparing to the previous one so it provides the more stable performance.

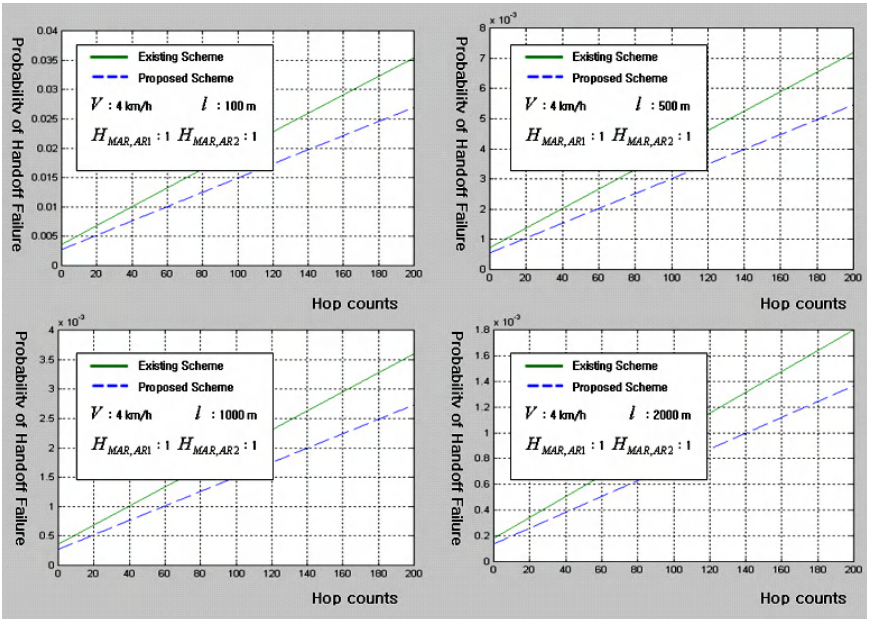


Fig. 5. The probability of handoff failure by the hop count

We assume in Fig. 5 that V is 4km/h, $H_{MAR,oAR}$ and $H_{MAR,nAR}$ are 1 each. We get the probability of handoff failure in terms of the number of hops. Increase of the hop count reflects the increment of handoff delay. We calculate the cell size when $H_{MAR,CN}$ is 5, the probability of handoff failure is 2%, and the velocity of MN is 10km/h. By the equation (6), the minimum cell radius of the existing scheme is 53.4m, meanwhile the proposed one is 40.4m. Therefore, the proposed scheme is more efficient than the existing one in terms of cell radius and the efficiency is improved up to about 24%. Similarly, by the equation (7), we are able to calculate the speed of MN. In case that $H_{MAR,CN}$ is 5, $H_{MAR,oAR}$ and $H_{MAR,nAR}$ are 1 each, the probability of handoff failure is 2% and the radius of l

is 500m. Then the maximum velocity of MN in the existing scheme is 25.5km/h while the proposed one is 33.7km/h. As a result, the proposed scheme is overall superior to the existing one and even it supports more stable environments.

5 Conclusion

In this paper, we propose the new MIPv6 based handoff scheme which supports QoS guaranteed real-time data service called partial dual unicasting(PDU). The PDU scheme is basically cooperated with RSVP signaling protocol, and resolves buffering problem and problems occurred by the ping-ponging effect. This can be accomplished by concurrently maintaining CoAs of old and new networks in binding cache of CN. According to the analytical model and the comprehensive experimental results, the handoff execution time on MNs is relatively longer comparing to the previous approach and the handoff failure rate is significantly decreased up to about 24% with almost negligible signaling overhead. Neighbor casting mechanism based on the proposed PDU scheme is underway.

References

1. C. Perkins(ed.), "IP Mobility Support," RFC 2002, October 1996.
2. R. Braden(ed.), "Resource ReSerVation Protocol(RSVP)-Version 1 Functional Specification," RFC 2205, September 1997.
3. B. David, C. Perkins, and J. Arkko, "Mobility Support in IPv6," IETF draft, Internet Draft draft-ietf-mobileip-ipv6-17.txt, May 2002, work in progress.
4. W. Woo and C. M. Leung, "Handoff enhancement in mobile-IP environment," IEEE International Conference on Universal Personal Communications, vol. 2, pp. 760–764, 1996.
5. C. Perkins and K. Wang, "Optimized smooth handoffs in Mobile IP," IEEE International Symposium on Comupters and Communications, pp. 340–346, 1999.
6. IETF old Working Group, "http://www.ietf.org/html.charters/old/intserv-charter.html"
7. J. McNair, I. F. Akyildiz, and M. D. Bender, "An inter-system handoff technique for the IMT-2000 system," IEEE INFOCOM, vol.1, pp. 208–216, 2000.
8. J. McNair, I. F. Akyildiz, and M. D. Bender, "Handoffs for real-time traffic in mobile IP version 6 networks," IEEE GLOBECOM, vol.6, pp. 3463–3467, 2001.
9. J. Xie, and I. F. Akyildiz, "An optimal location management scheme for minimizing signaling cost in mobile IP," IEEE ICC, vol.5 , pp. 3313–3317, 2002.
10. A. K. Talukdar, B. R. Badrinath, and A. Acharya, "MRSVP: A Resource Reservation Protocol for an Integrated services network with mobile hosts," Wireless Networks, vol.7 pp. 5–19, 2001.
11. R. Thomas, H. Gilbert, and G. Mazziotto, "Influence of the moving of the mobile stations on the performance of a radio mobile cellular network," in Proceedings of the 3rd Nordic Seminar, 1988.

Simulations on Batching in Video-on-Demand Transmissions*

Juan Segarra and Vicent Cholvi

Departament de Llenguatges i Sistemes Informàtics,
Universitat Jaume I, 12071 Castelló, Spain
{jflor,vcholvi}@lsi.uji.es

Abstract. One of the methods for taking advantage of multicast services is the use of batching. With this method, several request of the same video are grouped and transmitted together, using only the bandwidth required for one transmission. This method is commonly used in transmission of streamed data. In this paper we analyze the system performance with explicit constant batching, and demonstrate that a system without explicit batching performs better in terms of delays. We also propose a dynamic batching policy which improves the system performance both in mean and in maximum serving times.

1 Introduction

With the advancement of broadband network technology, Video-on-Demand (VoD) services are becoming commonplace, specially in local residential areas. A VoD system is typically implemented by a client-server architecture supported by certain transport networks. Among the many issues that have attracted the attention of researchers, optimizing the network channel demand is considered a major one. In order to solve that problem, VoD systems usually serve multiple clients at the same time by means of artificially delaying several requests for the same video [5,6,11,13] and serving them together. This method is known as *batching*.

However, these artificially introduced delays may not be always necessary or even convenient. For instance, in a system with low load, all video requests could be served immediately, and the use of batching would be clearly inefficient, since it only introduces an unnecessary delay. Moreover, the amount of bandwidth used in a high load period is much bigger than in low ones, and since these networks have to afford this requirement, in low load periods this resource is mostly underutilized despite of offering Internet connections or other similar services. Thus, bandwidth is not always a critical resource.

In this paper we focus on the convenience of using explicit batching. We will evaluate, by means of some simulations, how the use of batching improves (or reduces) the overall performance of the system described in Section 2. We

* This study is partially supported by the CICYT under grant TEL99-0582 and Bancaixa under grant P1-1B2000-12.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 453–462, 2003.
© Springer-Verlag Berlin Heidelberg 2003

consider the *serving time* (the time between a video request and the beginning of its transmission) as the metric to evaluate the system performance.

First, we use a *Constant Explicit Batch-Time Policy* with different delays. Whereas this policy is the most widely used, we show that, in general, it provides improvements on bandwidth, but it does not provide any delay improvement against a system without explicit batching (but still with implicit batching). Then, we introduce a new batching policy which varies dynamically following the system load. Even though such a policy has a better performance in starting delays than previous ones, it only provides a low improvement when the system is highly saturated.

The rest of our paper is organized as follows. In Section 2 we describe our video-on-demand delivery system and the considered scenario. In Section 3 we show our comparison between constant batching policies and in Section 4 we present our approach and results with dynamic batching. Finally, our conclusions are presented in Section 5.

2 System

For our study, we use a quite simple network architecture consisting of a central server, storing all videos. Users are connected to that server by means of a common link which has a bandwidth of 1.5 Gbps downstream. A moderate bandwidth is also needed upstream in order to send user requests to the central server. We consider that this is a VoD dedicated network[1], therefore our goal is to obtain the best results using the available resources. We use a configuration consisting of 1 000 videos and 15 000 users. Each video information has been constructed making variations over the example presented in [12], resulting in videos of a mean duration of about 85 minutes and a mean bandwidth requirements of about 1 850 Kbps in fragments ranging from 96 to 4 608 Kbps. With this configuration we ensure that the link will become *saturated* (i.e. it will reach an utilization of 100%) at some period of time, being non-saturated during the remaining time. That will allow us to analyze the system behavior in both situations.

On the other hand, transmitting real-time VBR flows is not a trivial matter because a different amount of bandwidth will be needed during the transmission [3,8]. One of the methods to improve these transmissions is the *smoothing* [7, 9,12] of streams before their transmission. This way peaks and rate variability requirements are minimized. We use an approach similar to [12], which develops a *transmission plan* consisting of time periods so that, in each period, the transmission may be performed by using a constant-bit-rate (CBR) network service. Thus, each video is characterized by a collection of tuples $\langle time, rate \rangle$, whose first parameter denotes a time period and whose second one is the CBR rate associated with such a time period.

[1] Usually these networks offer more services, but dedicating a concrete amount of bandwidth to each service is also usual.

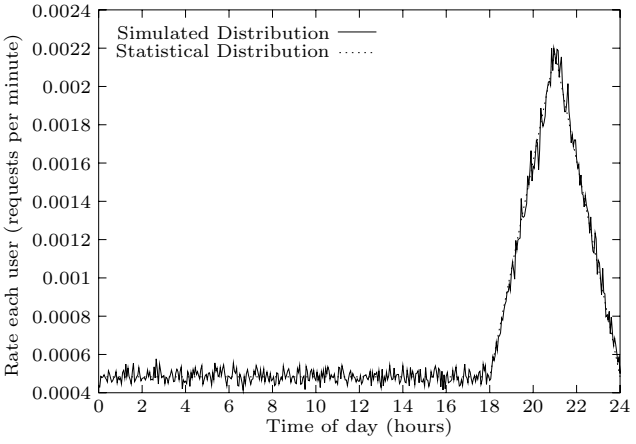


Fig. 1. Request rate distribution in a 24 hour period.

In our approach to the video transmission problem we study the performance in a 24 hour period. Previous studies on video demand rates [2,10] have determined the behavior of these rates in this period, and our simulations work with a rate distribution according to these studies. We also consider that a user makes a request only if he or she is not waiting nor receiving a video (i.e. nobody can receive more than one transmission at a time). In Fig. 1 we can see the statistical request rate and the request rate distribution obtained in a simulation having one request per day for each user. This is what we assume in our study (actual VoD trials show that the average number of user request per week is between 2 to 3).

Furthermore, each video has a *popularity*, which represents the probability of being requested according to Zipf's law [4]. This distribution has been found to statistically fit video program popularities estimated through observations from video store statistics [1].

In order to guarantee that, once a video is accepted (maybe after some start-up delay), it will be delivered without interruption to the final user(s), it is necessary to use a *reservation algorithm* to manage video requests. It works as follows: when a video request is received, we test (for each video-part) if it is already planned for transmission at the same time. If so, that request is added to the multicast transmission of this video-part without any additional bandwidth requirements. Otherwise, we check if there is enough bandwidth during the slots of the transmission. In that case, bandwidth is reserved by adding the new video-part rate to the bandwidth used. See Fig. 2 for details.

As it can be seen, this reservation algorithm guarantees that all requests will be served, maybe after some start-up delay. Moreover, users can know the exact time when the transmissions will begin, so we assume that there are no cancellations.

Step #1: A user requests a video.
Step #2: If that video has been already requested and it has not started yet, the start time of the new request will be the same as the previous one. Otherwise, the new request will start after an explicit batch-time.
Step #3: The system calculates the exact time when each video-part has to be transmitted.
Step #4: For each video-part do
If there is a reserve for this video-part at the same time, add our request as a multicast. Otherwise, reserve the bandwidth needed for this video-part during the required time slots.
Step #5: Accept video for transmission or go to Step #3 in the next time slot (in which case, all reserves and multicasts for this request performed in Step #4 are canceled).

Fig. 2. Algorithm for the acceptance of video requests.

An important feature of the reservation algorithm is the fact that, in saturation periods, all new request will have to wait to be served. So, even in the case where the there is not explicit batch-time, the reservation algorithm will still induce an *implicit batching* that, as we will show in the next sections, provides a very good system's performance.

3 Constant Batching Policies

Using a constant explicit batching scheme is currently the most used policy maybe because of its simplicity. Basically it consists of taking a constant batch-time delay throughout the whole system execution.

In this section, we analyze the performance of our system when taking explicit batch-time values of 0, 10, 20 and 30 minutes (despite we have tested more values, for clarity we only present results for these four values, since we found the rest follow the same pattern). Fig. 3 shows the starting delays in a 24 hour period.

First of all, it can be readily seen that, when the request rate is low (from approximately 0h to 19h), the more we increase the batch-time the more the starting delay increases. On the other hand, Fig. 3 also shows that during the time period when the request rate is high (from approximately time 19h to 24h) starting delays are all following the high audience peak independently of the batch-time used. This behavior can be explained if we look at the reservation algorithm in Fig. 2. This algorithm tries to allocate transmissions as soon as possible after their batch-time. However, in the time interval where the system is saturated, the delay necessary to start transmitting is higher than the batching delay. Consequently, since the system gets saturated in a short time interval (a few minutes), at the very end, all policies operate in the same way most of this time period, confirming what has been observed in Fig. 3.

In Fig. 4 we can see that, in the low load period, bandwidth usage never gets saturated. Therefore, since there is enough bandwidth to serve immediately all

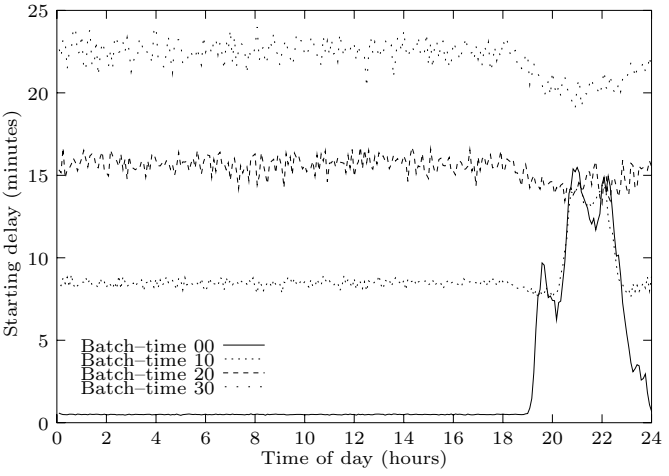


Fig. 3. Mean starting delays during a 24 hour period.

requests, adding a batch-time will only increase their starting delays. In turn, taking smaller batch-times will increase the used bandwidth. However, here we are assuming that the whole bandwidth is completely dedicated to our video-on-demand system. On the other hand, it can be seen that with very high explicit batch-times it is possible to avoid saturation. However, it also prevents from obtaining better starting delays.

4 Dynamic Batching

We have demonstrated, in the previous section, that a policy without explicit batching performs better in delay terms than with constant batch-times. In this section we present a dynamic batching approach to the problem. Such an approach consists of obtaining a batch-time that adjusts itself according to the current system load.

Our idea is the following. Think that the system receives a video request in the instant t, and the objective is to compensate the load we will have in the instant $t + \Delta t$. This compensation is done using batching over the requests in the current time t. The batch-time we assign to a request on time t is defined as follows:

$$batch_t = load_{t+\Delta t}\, adjust$$

where $load_{t+\Delta t}$ represents the number of videos awaiting for transmission in the time slot $t + \Delta t$ and $adjust$ is a function that will be used to adjust the load with the batch-time. In order to make load independent of the system, $load_{t+\Delta t}$ has been normalized so that a value of 1 indicates the saturation point.

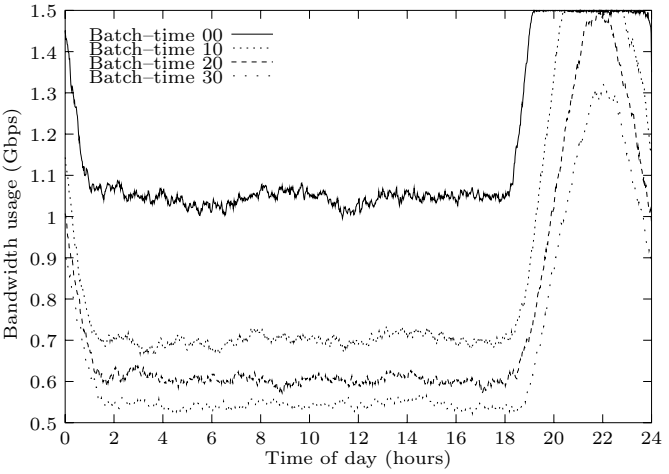


Fig. 4. Bandwidth usage during a 24 hour period.

To define *adjust*, think in a system which is continuously receiving requests and immediately serving those requests at the same rate. This system has, in addition, a list of video requests waiting for being served. Therefore, when a request for a given video is received it may happen that that video is also in the list waiting for being served. Thus, both requests would be transmitted as a multicast and the additional load list would decrease in one video. If this procedure is used until all waiting videos in the list are transmitted, at that time we would have eliminated the additional load list. Let us call $EliminationTime$ the time where the additional load is eliminated.

$$EliminationTime = AdditionalLoad \frac{1}{ProbCoincidence}$$

where $AdditionalLoad$ is the additional load list (normalized as above) we have to eliminate, and $ProbCoincidence$ is the probability of having a coincidence when requesting a video for transmission, that is, the probability of the requested video being in the load list.

For our work, we choose $batch_t \equiv EliminationTime$. Since we have that $load_{t+\Delta t} \equiv AdditionalLoad$, then $adjust \equiv \frac{1}{ProbCoincidence}$. Therefore,

$$batch_t = \frac{load_{t+\Delta t}}{ProbCoincidence}$$

A detailed description of $load_{t+\Delta t}$ and $ProbCoincidence$ can be found in the Appendix.

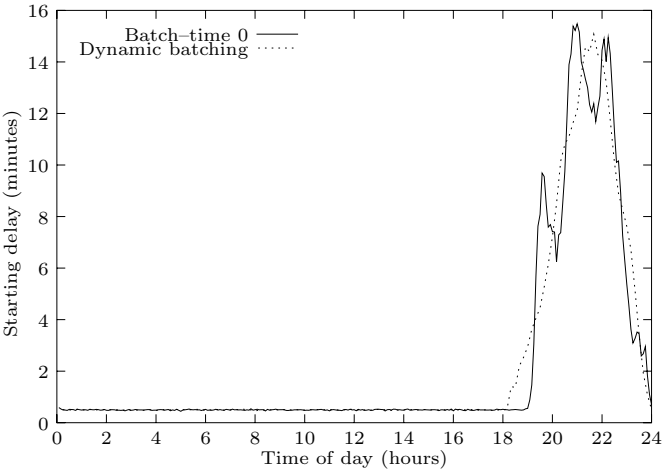


Fig. 5. Starting delays during a 24 hour period.

Results

Now, we show a comparison between this policy and one without explicit batching (which has been shown in the previous section to provide better results than using any explicit constant batching policy).

Fig. 5 shows the starting delays both without explicit batching and with a dynamic batching policy. During the low load period, both policies perform very similarly and between 0 and 1 minute. When the high load period begins, the dynamic policy responds adding artificial delays for grouping requests. This action produces a sooner increase in starting delays but this prevents starting delays from increasing so much as without explicit batch-time when these delays are higher. That is because the dynamic policy has a look ahead component, whereas the batch 0 policy has no information about the future system load.

The mean starting delay during a 10 days period with the dynamic policy in this case is 4 min 21 sec, and its maximum delay has been 27 min 22 sec. These values are about a 0.4% and a 4.7% lower than without explicit batch-time.

Bandwidth usage with these configurations is presented in Fig. 6. Whereas the bandwidth usage is almost the same, with the dynamic policy the added batching produces a delay in saturation. The reason is the same as above, grouped requests are transmitted together in multicast transmissions and they use less bandwidth.

5 Conclusions

In this paper we have studied the effect of using batching in the transmission of video on demand. Instead of using cost functions, this study presents results

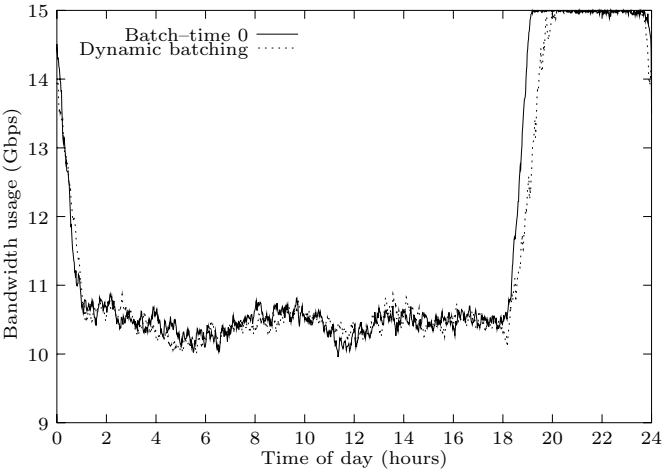


Fig. 6. Bandwidth usage during a 24 hour period.

based on the delays between video requests and the beginning of their transmission. Firstly, we have demonstrated that, without using explicit batching techniques, the overall system performs better than using a constant explicit batch-time. Furthermore, to improve the system performance, we have proposed a method for statistically calculate the system load, and use the resulting value to adjust the batch time dynamically. With this method we have reduced around 0.4% the mean starting delay and around 4.7% the maximum starting delay compared with a system without batching.

These results demonstrate that, unless a constant explicit batch-time is needed for some specific reason, it is better not using it. In case of needing a more optimized system, we also demonstrate that our dynamic batching method is better than using a batch-time 0 policy. However, the improvements are relatively small and require additional computation, so a policy without explicit batching could also be considered as a good option due to it performs well and it is the simplest batching policy. The reasons of the good performance are that without explicit batching there is no artificial delay introduced in the system, and this policy has a good adaptation to the current load, since it allocates the transmissions as soon as possible.

References

1. A. Dan and D. Sitaram and D. Shahabuddin. Dynamic Batching Policies for an On–Demand Video Server. *Multimedia Systems*, 4:112–121, 1996.
2. Bell Atlantic. Fact Sheet: Results of Bell Atlantic Video Services. Video–On–Demand Market Trial. Trial Results, 1996.

3. L. Berc, W. Fenner, R. Frederick, and S. McCanne. Rpt payload format for jpeg-compressed video. Request for Comments 2035, Network Working Group, October 1996.
4. Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of the INFOCOM '99 conference*, March 1999.
5. Shueng-Han Gary Chan and Fouad Tobagi. Providing on-demand video services using request batching. *IEEE Int. Conf. Communications (ICC'98)*, pages 1716–1722, June 1998.
6. Asit Dan, Dinkar Sitaram, and Perwez Schahabuddin. Scheduling policies for an on-demand video server with batching. *ACM Multimedia*, pages 15–23, 1994.
7. W. Feng, F. Jahanian, and S. Sechrest. An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video. *Multimedia Systems*, 5(5):297–309, 1997.
8. D. Gall. Mpeg: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.
9. Jeniffer Rexford and Don Towsley. Smoothing Variable–Bit–Rate Video in an Internetwork. *IEEE/ACM Transactions on Networking*, pages 202–215, April 1999.
10. PG de Haar et al. DIAMOND Project: Video–on–Demand System and Trials. Eur Trans Teleccommun (8)4: 337–244, 1997.
11. Juan Segarra and Vicent Cholvi. Distribution of video–on–demand in residential networks. 8^{th} *International Workshop on Interactive Distributed Multimedia Systems*, September 2001. Lecture Notes in Computer Science 2180.
12. Arun Solleti and Kenneth J. Christensen. Efficient transmission of stored video for improved management of network bandwidth. *International journal of network management*, 10:277–288, 2000.
13. Constantinos Vassilakis, Michael Paterakis, and Peter Triantafillou. Video placement and configuration of distributed video servers on cable TV networks. *Multimedia Systems*, 8:92–104, 2000.

Appendix: Description of Dynamic Batching Parameters

Definition 1 *We define the* real load *in a time slot t as:*

$$rload_t = \frac{|(Awaiting_{t-1} \cup Requested_t) - Served_t|}{ServingCapacity}$$

where $Served_t$, $Requested_t$ and $Awaiting_t$ respectively denote the set of different videos served, requested and awaiting for transmission in the time slot t, and $ServingCapacity$ is the mean number of different videos served in a time slot using all bandwidth.

Nevertheless, this definition is not usable in practice, because it needs both which videos are awaiting for transmission and which of them will be served in each time slot. Obviously, it is not known in advance which videos will be requested, nor which ones will serve the system since that depends on the batch-time assigned to each request. Therefore, we need to obtain a (statistical) load value independent of the batching algorithm. All probability operations below are done using the Zipf's distribution, which offers probability values based on popularity.

Definition 2 *We define the* statistical load *in a time slot t as:*

$$sload_t = max\left(0, sload_{t-1} + \frac{NewRequests_t - ServingCapacity}{ServingCapacity}\right)$$

In this equation, we calculate the load in a time slot t by adding the load generated by the new requests to the load of the previous time slot and subtracting the amount of load transmitted, which is 1 because of the normalization[2].

The $NewRequests_t$ parameter is the number of new requests in the time slot t, and it is calculated as:

$$NewRequests_t = ProbNewReq(ReqInSlot(t), sload_{t-1}\, ServingCapacity)\, ReqInSlot(t)$$

where $ReqInSlot(t)$ is the number of requests in the time slot t obtained statistically using the request distribution and $ProbNewReq(new, awaiting)$ is the probability of that *new* requests not being included in *awaiting* requests. Statistically $ProbNewReq(new, awaiting)$ is calculated as $P(Videos(new) \mid Videos(new) \nsubseteq Videos(awaiting))$, being $Videos(x)$ a set of x $VideoID$s obtained statistically from the repository using Zipf's law.

However, obtaining this statistical load value has a computation cost of $O(MaxVideos^{new})$ in the step $ProbNewReq(new, awaiting)$, so an approximation is needed. We obtain this approximation supposing that the proportion of different requests in each time slot is a constant k, and the $sload_{t-1}$ videos awaiting for transmission are the most popular ones. The first simplification is reasonable when the number of requests in each time slot is not very different and in our case it is less than one order of magnitude. The second one is reasonable having in mind that Zipf's law popularity implies exponential probabilities of request, so the most popular videos are much more requested than the others. The usable $ProbNewReq(new, awaiting)$ definition would be the probability of not requesting the *awaiting* most popular videos. Statistically: $k\, P(x \mid x > awaiting)$, where k is the proportion of distinct elements in a group of $VideoID$s requested in each time slot.

The remaining definition is $ProbCoincidence$, but it is exactly the opposite of $ProbNewReq()$, so its calculation is immediate.

The simulation parameters can be easily obtained. $ServingCapacity$ is obtained dividing the link capacity by the product of the mean bandwidth usage of all videos each minute and the mean video duration. The resulting value was 9.42 videos. The parameter k is obtained using the mean value from a complete simulation. The resulting value was $k = 0.77$.

To obtain the best value of Δt, we have simulated our system with Δt ranging from 0 to 10 hours. Depending on the pursued objective this value can be adjusted closely, in our case we take a value of 60 minutes, which offers good results both for mean delay and maximum delay.

[2] Actually, we do not know the value representing the transmitted videos, so we assume the system is serving at full capacity (1 because of the normalization) and then use the *max* operation to prevent the system from getting load values lower than 0.

The Design of XML-Based Internet Security Integrated System Architecture

Kwang H. Kim, Tae-Kyung Kim, Dong S. Kim, and Tai M. Chung

Real-Time Systems Laboratory,
School of Electrical and Computer Engineering,
Sungkyunkwan University,
300 Cheoncheon-dong, Jangan-gu, Suwon-si, Gyeonggi-do, 440-746
{byraven, tkkim, dskim, tmchung}@rtlab.skku.ac.kr

Abstract. To protect networks from intrusions and attacks, many vendors have developed various security systems. Managing these systems individually demands much work and high cost. To decrease these overheads, we have developed an Internet Security Integrated System(ISIS). This system is composed of several security systems and applies the security policy to each security system to establish the security zone. But we have some problems in standard forms of platform interfaces, operating methods, log formats, and message structure required in integrating security systems. So, we propose a message exchange scheme among the security systems using XML. This approach presents significant advantages in terms of increasing simplicity and scalability, supporting ease of maintenance, and providing fast information retrieval.

1 Introduction

Today, the Internet is connected to almost everyone all around the world. Despite the usefulness of the Internet, the security issues are emerging from intrusions and attacks such as computer viruses, denial of services, and hackings to destroy information, which have been considered to be seriously threatening the Internet. And the number of intrusions and incidents is increasing every year[18]. So using firewalls, intrusion detection systems, virtual private networks, and vulnerability scanners in network, we protect the information systems from those threats. But the uses of various security systems require not only highly advanced management knowledge but also many administrators to handle the different user interfaces and functions. The most important problem is that it is very difficult to maintain the consistency of the security policies among different security systems. However, we can raise the security level, using the security events of some security systems. Also, it is necessary to integrate heterogeneous security systems to raise the security level by cooperating with other security systems.

Our Integrated Security Management System(ISMS) is suggested as a solution to these problems[5]. The ISMS integrates user interfaces and methods of parameter

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 463–472, 2003.
© Springer-Verlag Berlin Heidelberg 2003

setting, and it is very helpful in maintaining the consistency of security policy. And the ISMS applies a policy globally among the different security systems. But the ISMS has some problems in different interface protocols, variety of factors, and different platforms among the heterogeneous systems. Most of all, we must solve the problem of message exchange among the security systems. The message exchange scheme has an influence on performance of the system. So we suggest a message exchange method to be applied by using XML among the security systems. We use this message exchange method for setting configuration, distributing policy, alerting intrusion detection, and logging.

In this paper, we suggest an Internet Security Integrated System(ISIS) Architecture. The ISIS is one of the Internet Security Management Systems and uses XML-based message exchange scheme. We expect the ISIS to be capable of providing the fundamental methods of integration of different security systems.

This paper is organized as follows. In section 2, we show several security systems that use the XML and the problems of systems. In section 3, we explain the architecture and function of the ISIS. In section 4, we evaluate the performance of the ISIS whether the XML message exchange scheme is useful or not. Conclusions and future works are discussed in section 5.

2 Related Works

XML has been developed successively since W3C had announced XML 1.0 recommendation[1]. XML is applied in various fields of industry, but it is rare for security area or products to use XML. But these days XML is used in the security area because security systems need the characteristics of extendibility, flexibility and formality.

XML has the following characteristics : (i) Simplicity - XML makes it easier to provide metadata, or descriptive data about the information. XML is easily readable and understandable to both people and computer. Also, it is easily processed by computer and is easy to learn, so it can reduce the development time. (ii) Powerful - XML can represent not only complex data structure but also security sensitive data. (iii) Scalability - Even though we use a text-based format, it can be also supported from XML. (iv) Exchangeability - XML allows most computer systems to manipulate XML without major modifications of existing infrastructure. (v) Independency - XML has separated content from design. So XML allows the look of the document to change without changing the document.

Intrusion Detection Working Group(IDWG) of IETF has developed the Intrusion Detection Message Exchange Format(IDMEF) to define the data formats and exchange procedures for sharing the information of interest. The IDMEF describes a data model to represent the information exported by intrusion detection systems, and explains the rationale for using this model. Currently, IDWG has published four internet drafts - Intrusion Detection Message Exchange Requirements, Intrusion Detection Message Exchange Format Data Model, and Extensible Markup Language (XML) Document Type Definition, the TUNNEL Profile, and the Intrusion Detection Exchange Protocol (IDXP)[3]. Although IDMEF provides flexibility of alert data format,

it is insufficient to maintain the policies and the configurations of security systems. But XML provides more suitable solution than SMI in the interactions of systems [3].

DAXFi(Dynamic XML Firewall) is a Python package that helps to configure several different kinds of firewalls in a consistent way. Rules can be described with XML files, XML strings, or generated directly by the code. It comes with a Python package, useful in building other applications aimed at manipulating different firewalls in a uniform way. DAXFi has offered the integrating methods of firewall systems. But its extensibility was restricted to some firewalls. It supports iptables, ipfilter, ipchains and ipfwadm[19].

Snort is a lightweight network intrusion detection system, and it can perform real-time traffic analysis and packet logging on IP networks. Snort uses the flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. The XML plug-in enables snort to log in Simple Network Markup Language(SNML) to a file or over a network. We can use this plug-in with one or more snort sensors to log on to a central database and create highly configurable intrusion detection infrastructures within networks. But Snort is restricted with narrow uses of XML in alert or log.

There are many efforts to use the XML in security systems. But most of them use the XML in the same products at system log or alert message. Therefore, we integrate security systems using XML. And we show the efficiency of the ISIS, which uses the XML, comparing it to the conventional ISMS by simulations.

In the next section, we will suggest the architecture of the ISIS using XML.

3 Architecture of the ISIS

3.1 Overview of the ISIS

The ISIS is composed of several security systems and applies the security policy to each security system to establish the security zone. The ISIS is required to solve the problems in the following manner: Firstly, the more we hire a security system, the more we suffer from managing the security systems. The security systems have many configuration parameters, so management skill is required to manage them. Secondly, individual security policies cause problems of security policy conflict. The lack of consistency in security policies can bring serious problems. Thirdly, distributed management of each security system requires much cost such as manpower, time, and money. Therefore, we designed the ISIS to diminish the Total Cost of Ownership(TCO).

Figure 1 shows the conceptual architecture of the ISIS. The ISIS has integrated the security systems as agents to control accesses, inspect vulnerabilities, and detect intrusions. The ISIS consists of four elements – the ISIS Client, the ISIS Server, the ISIS Agent, and the ISIS database(ISISDB)[5,8]. The ISIS agents installed on each security system are distributed in large networks to collect the logs and runtime events, but also control the systems. The ISIS Server processes the request of the client and collects the information of logs, performances, configurations, and policies. And the collected

data are sent to the ISIS client to manage the ISIS. The ISISDB has stored the user information, system configurations, security policies, performance data, security events, and logs.

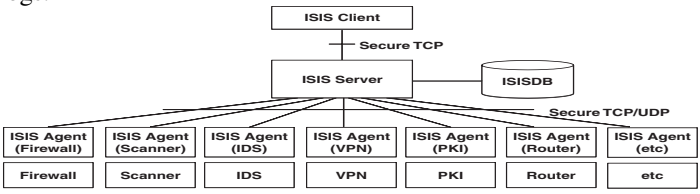


Fig. 1. A conceptual architecture of the ISIS.

3.2 Design and Implementation of the ISIS

Architecture of the ISIS follows a 3-tier model(Client, Server, Agent), with the specific packages designed for each tier. Figure 2 shows the main module of the ISIS. The functions of each module are as follows.

CMM(Configuration Management Module) : This module manages the configuration information of the security systems. CMM offers basic information to the others modules. We can obtain or change the configuration properties. Whenever the security system boots up, the agent sends the configuration information to the server. The server maintains configuration information in database and uses as basic information for other modules.

PMM(Policy Management Module) : PMM manages the security policies concerned the security systems of the ISIS. It operates the distribution, modification and deletion of security policies. It also checks the validity of the policies and provides the method of checking an authorization to perform the operations.

PeMM(Performance Management Module) : This module measures the performance of the ISIS. PeMM has the performance configurations of the security systems and the ISIS Server, and performs the response action when the performance gets over the threshold value.

UMM(User Management Module) : UMM manages the user authentication and authorization. UMM maintains Access Control List(ACL)s and checks the authorization of user using ACLs. According to the user access level, the operations are determined whether it is permitted or not.

SOMM(Security Object Management Module) : The network objects concerning the security policies in the ISIS are called the Security Object(SO)s. SOs are composed of security systems, routers, switches, and hosts. SOMM manages the objects, which are the subjects of the operation concerning with the configurations, the policies, and events.

EMM(Event Management Module) : EMM is in charge of alarm to client. It notifies the security events from security systems to the administrator and reports the event logs to compile a statistical report.

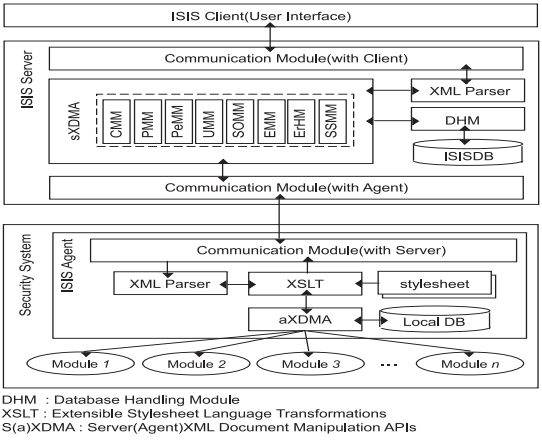


Fig. 2. The system Architecture of the ISIS

ErHM(Error Handling Module) : ErHM collects the system errors, and responds to errors, and reports their analysis. It can report the conflict of security policies, the violations of authentication or authorization, and system faults.

SSMM(Security Service Management Module) : SSMM maintains the available service lists in the ISIS. The administrator has to select the service names to use.

These modules are operated using the message exchange with the clients or the agents. And the ISIS must prepare the extension of messages caused by adding new security systems. The message structure has an influence on the scalability of the system. We have guaranteed the system scalability by using XML message exchange scheme. In the next section, we suggest a message exchange scheme in detail.

3.3 Message Definition of the ISIS

We had developed the ISIS MIB, and used it in past. The ISIS MIB had managed objects, policies, services, users, monitoring or performance information and et al. The ISIS MIB had been developed using the BER Encoding rule[20]. These information is similar to current message structure. When we had developed the ISIS, we have faced to unexpected difficulty. The difficulty is that we had problems about maintaining the different message formats of security systems entirely, and we took much time to develop the agent of security systems. So, we suggested the XML based integrated systems to offer the convenience of appending the additional security systems[3].

There are five types of message. Each type has a subclass corresponding to each security system; Firewall, IDS, VPN, Scanner, etc.

Currently, the data model representing the information must be flexible to accommodate different needs[10]. Figure 3 shows the relationship between the principal

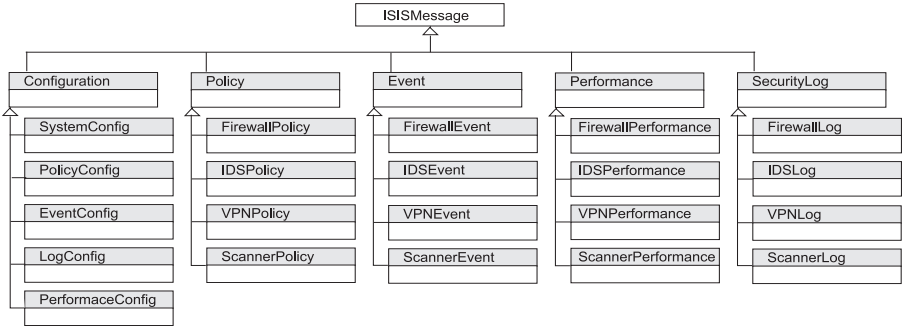


Fig. 3. The ISIS Message Architecture.

components of the data model. The top-level class of all the ISIS messages is the ISISMessage; every message is a subclass of this top-level class. Five types of message are defined - Configuration, Policy, Event, Performance, and Security Log. Within the message, subclasses are used to provide the detailed information.

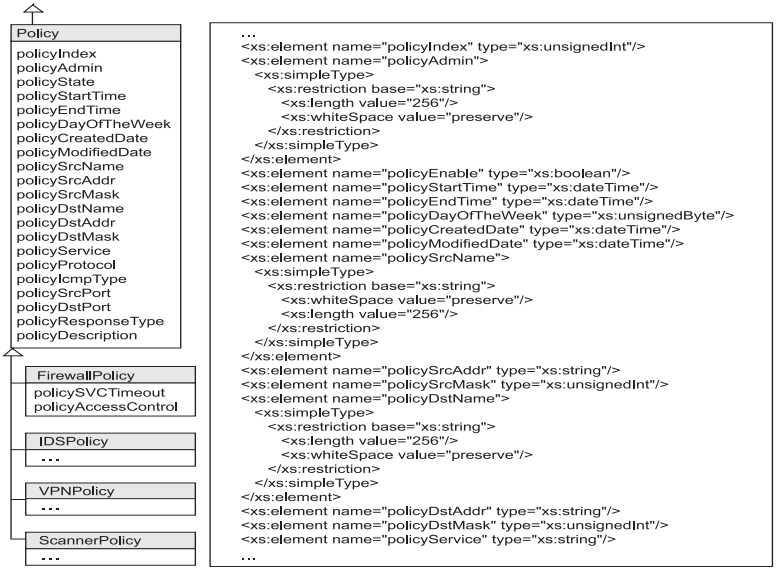


Fig. 4. A Part of XML Schema of interchange Message in the ISIS.

The Communication Module operates by using text-based scheme. Figure 4 shows a part of XML schema of interchange messages in the ISIS. The policy class contains the attributes generated by security systems. We define elements and attributes using element and attribute schema elements, and structures by using the simpleType and the complexType schema elements. The XML schema increases the reusability on the

basis of the object-oriented concepts and customized data types. We can omit the validation codes to delegate validation of data type to the parser applying the XML schema. But XML uses a text format and tags to delimit the data. Therefore, the XML format is larger than binary format[14].

Figure 5 shows an example of setting the security policy. The administrator commands the ISIS server to set the security policies. The ISIS server enables the parser to carry out the task of validating messages. If messages are valid, the PMM receives the next task of conflict checking. The PMM checks whether the policies are in conflict with the existing policies by using the conflict check algorithm [8]. Then, the server transmits the policies to the ISIS agent. The ISIS agent translates the received message to the format of each security system. Each security system may have its own format; the parameters, the tag names, the structures, etc. The XSLT handles the translation process. The XSLTs of the agents are important because they take charge of conversion to the understandable messages. Actually, the development of XSLT takes long time in adding an agent. The Agent:XDMA processes the remaining works by using the translation result. The ISIS agent returns the result of policy set. The Server stores the policies, or logs the error according to the result.

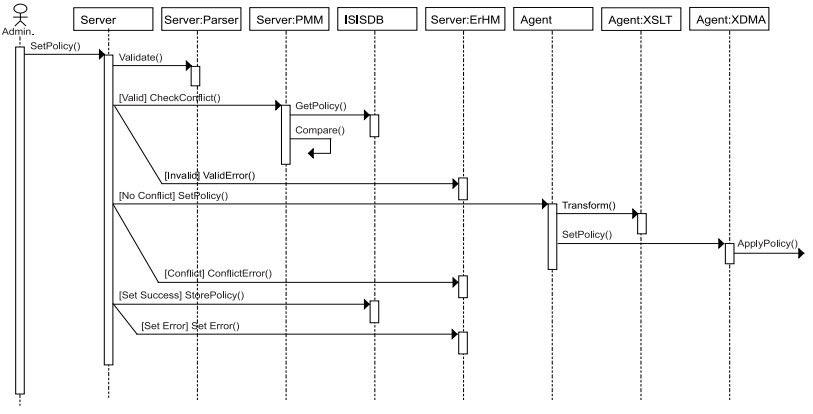


Fig. 5. Sequence Diagram about the Policy Set

4 Comparison of Performance

We have examined the usefulness of XML based scheme. In this simulation, we did not take the propagation latency in network into consideration. And we only estimated the message parsing time of the server and the agent because each security system uses its own particular message format after the message parsing.

The simulation was carried out using the policy set module to both the IDS and the firewall agent. The IDS and the firewall agent were installed in two machines, Heaven and Ixion. In each host, we installed the ISIS server with an agent to decrease the

network delay. Thus, we measured the message processing time – from one message generation to another message generation. We increased the policy count from 1 to 100 and performed 10 thousands times for each system. Figure 6 shows the result of our simulation. Figures 6 (a) and 6 (c) indicate the processing time in the past method using the BER encoding. As the count of policies increased, the processing time increased slowly and stably. Figure 6 (b) and (d) show the result using the XML message format. Compared with the previous system, the elapsed time has taken longer when the XML was used. Though the XML message scheme takes more time than the BER message scheme, the elapsed time was under 100ms.
Simulation circumstance was as follows.

```
Oracle XML Developer's Kit for C 9.2.0.4.0
Heaven(IDS Agent) :
    OS : Solaris7
    CPU : UltraSparc II 450Mhz
    Memory : 512MB
Ixion(Firewall Agent) :
    OS : Solaris7
    CPU : UltraSparc IIi 440Mhz
    Memory : 512MB
```

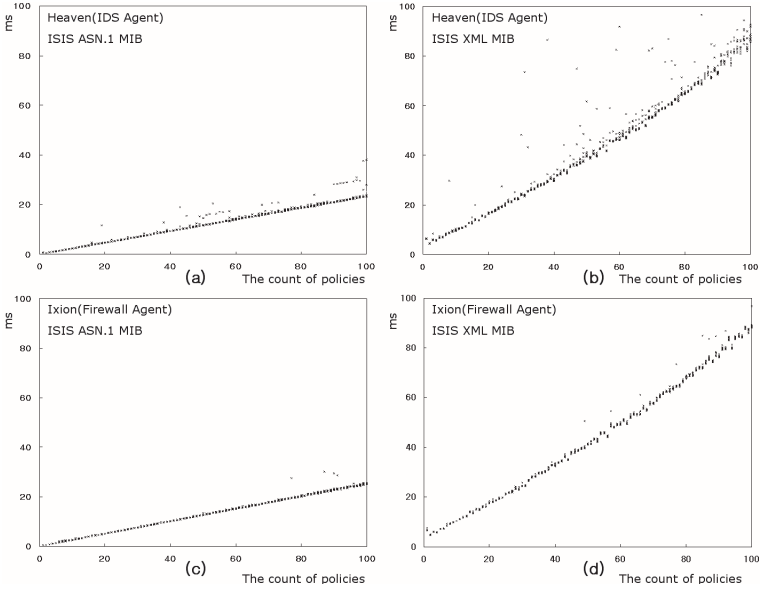


Fig. 6. The Parsing Result of Policy Set Message. (a) Processing Time of ASN.1(BER) Message for IDS Agent. (b) Processing Time of XML Message for IDS Agent. (c) Processing Time of ASN.1(BER) Message for Firewall Agent. (d) Processing Time of XML Message for Firewall Agent

Recently, security systems are being considered as real time systems. Therefore, it is very important to respond quickly to abnormal activities. To overcome the slow Process time of XML, many studies are in progress to develop an accelerator of the XML process. [21].

5 Conclusion and Future Works

Nowadays, security systems run independently. And this independence has brought the problems of difficulty in responding to increasing security threats. We have proposed the ISIS to solve the Internet threats using Integrated Security Management. The ISIS has improved the security intensity through collaborated works among the security systems. The ISIS Architecture has a 3-tier architecture, and the agents of the ISIS are customized to operate security tasks, taking the characteristics and the operation of each security system into consideration. The ISIS uses the XML-based message exchange as a method of collaboration among security systems. Using the XML, we increase the scalability and integrity of the ISIS with a minimum effort.

Comparison of the response time of the security system using BER encoding method and the ISIS using the XML method shows that response time of the ISIS is somewhat slower, but it can be acceptable. Through further study of the XML processing methods, we will continue to enhance the performance of the ISIS.

In the future work of this study, we are planning to optimize the ISIS for highly efficient throughput. We will study the optimization of XDMA, XSLT or DHM, ISISDB, and database query. A more quantitative approach will be needed to solve this issue.

References

1. W3C XML Working Group: Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/REC-xml, 2000. 10. 6
2. W3C XML Schema Working Group, XML Schema Part 0,: Structures, http://www.w3.org/TR/2001/REC-xmlschema-0-20010502, 2001. 5. 2
3. IETF IDWG, The Intrusion Detection Exchange Protocol (IDXP), http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-07.txt, 2002. 10. 22
4. IETF IDWG, Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition, http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt, 2001. 1. 30
5. Dong, Y. L., Dong, S. K., Seung, S. H., Tai, M. C.: A Development of Web-based Integrated Security Management System for Firewalls, The Transactions of KIPS, Vol.7, KIPS, NO10, pp. 3171–3181, 2000
6. Samuel, C. M.: XML and Information Security, Information Security Management Handbook(4th), Vol.3, CRC Press LLC, pp.551–361, 2002
7. Xavier, E.: XML based security for e-commerce applications, Engineering of Computer Based Systems, pp. 10–17, 2001
8. Yoon J. K., Kwang H. K., Hyun H. C., Tai M. C.: Policy Conflict Probing for Integrated Security Management, APNOMS2001, pp. 377–387, 2001

9. Roy, J., Ramanujan, A.: XML schema language: taking XML to the next level, IT Professional, Volume: 3 Issue: 2, 2001
10. Widergren, S.; deVos, A.; Jun Z.: XML for data exchange, Power Engineering Society Summer Meeting, Vol. 2, pp.840–842, 1999
11. Daniel, E., Robert, L.: Structured Information Retrieval using XML, ACM SIGIR2000 Workshop On XML and Information retrieval, Athens, Greece, 2000
12. Takeshi, I., Blair, D., Ed, S.: XML Encryption Syntax and Processing, W3C, http://www.w3.org/TR/xmlenc-core, 2002. 12. 10
13. Mark, B., John, B., Barb, F., Brian, L., Ed, S.: XML-Signature Syntax and Processing, W3C, http://www.w3.org/TR/xmldsig-core, 2002. 2. 12
14. Takashi, I., Hiroshi, M.: Mapping between ASN.1 and XML, Applications and the Internet Symposium. 2001
15. Joseph, F., Francis, P., Chris, B.: Converting Relational Database into XML Document, Database and Expert Systems Applications, Proceedings, pp.61–65, 2001
16. Flesca, S., Greco, S., Zumpano, E.: Modeling and querying XML-data, Database Engineering and Applications Symposium, pp. 275–283, 2000
17. Jason, M., Jennifer, W.: Query Optimization for XML. Proceedings of the Twenty-Fifth International Conference on Very Large Data Bases, Edinburgh, Scotland, September 1999
18. CERT/CC Statistics 1988-2002, http://www.cert.org/stats/cert_stats.html
19. The Dynamic XML Firewall, http://daxfi.sourceforge.net,
20. Steedman, Douglas., Abstract Syntax Notation One (ASN.1) – The Tutorial and Reference, Technology Appraisals, 1990
21. Larry Seltzer, “XML on a chip?”, http://techupdate.zdnet.com/techupdate, 2002. 12. 17

Performance Analysis of a Mobility Support Scheme between Subnetworks in Differentiated Services*

Jongoh Kim[1], Jonghyun Choi[2], and Youngsong Mun[2]

[1] Samsung Electronics Co. Ltd.,
Suwon Si, Gyeongi Do, Korea,
Fido@samsung.com

[2] School of Computer Science, Soongsil University,
Sangdo 5 Dong, Dongjak Gu, Seoul, Korea
wide@sunny.ssu.ac.kr, mun@computing.ssu.ac.kr

Abstract. The Differentiated Services architecture does not have any provision to support the mobile host. This paper analyzes the performance of a scheme to support the mobility service in the Diff-Serv using the Mobile IP. As results, it is shown that the cost varies around 2 times of that of the Mobile IP. Premium service shows higher costs than Assured service does. It is also shown that packet drop probability for mobile hosts may be managed negligible if the proposed scheme is used.

1 Introduction

Internet Service Provider (ISP) traditionally has provided the same level of service to all customers, namely the best-effort service. The increased internet usage resulted in a performance hit of highly mission critical applications. At the same time, newer applications that need a better service quality have emerged.

The Differentiated Services (Diff-Serv) [1] is an effective approach for Quality of Service (QoS) support in the internet. The Diff-Serv is expected to provide a better QoS control with scalability than existing schemes today. However, the Diff-Serv in its current form does not have any provision for the mobility services.

Recent advances in the wireless network interfaces and the implementation of the global networks make host mobility an issue of interest both in wireless and wired networks. Thus, in this paper a scheme to support the mobility service in the Diff-Serv is proposed.

The paper is organized as follows. In section 2, related works are briefly overviewed. Proposed mobility scheme in the Diff-Serv architecture is described in Section 3. The network architecture is described and operations for supporting the mobility service in the Diff-Serv using the Mobile IP [2] are proposed. In section 4, the

* This work was supported by Korea Research Foundation Grant (KRF-2000-041-E00266)
Dr. Mun is the corresponding author.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 473–481, 2003.
© Springer-Verlag Berlin Heidelberg 2003

performance analysis is performed. Results of the performance analysis are given in section 5. Finally, the conclusion is given in section 6.

2 Related Works

The Mobile IP is proposed to enable nodes to change their point of attachment to the internet without changing their IP address [2]. R. Jain *et al.* performed the performance analysis of the Mobile IP [3]. However, they did not consider the cost incurred by the packets from the mobile host (MH) to the correspondent host (CH).

S. Black *et al.* proposed Diff-Serv [1]. Diff-Serv defines the layout of the TOS byte (DS field) [4] and a base set of packet forwarding treatments (per-hop behaviors, or PHBs) [5]. By marking the DS fields of packets differently and handling packets based on their DS fields, several differentiated services becomes essentially a relative-priority scheme.

In order for a customer to receive the differentiated services from its ISP, he/she must have a service level agreement (SLA) with the ISP. The Premium service and its corresponding Expedited Forwarding PHB provides a guaranteed bandwidth, low delay and low loss service that show the same characteristics as a virtual leased line [6]. The Assured service proposed in [7] only permits a statistically guaranteed rate. It permits to use additional available capacity while providing a base rate. Packets that exceed the negotiated rate are either marked as the best-effort traffic or dropped. The overall dropping probability of Assured service packets is considerably lower than for best-effort packets.

M. May proposed a simple model for the performance analysis of Diff-Serv [8]. Each service class using queue models was analyzed.

3 Proposed Scheme

The network architecture premised in this paper is the same one defined in [9] with the bandwidth broker (BB) [10]. In a domain, there may be more than two subnetworks. For the mobility support there should be agents defined in the Mobile IP in each subnetwork. The leaf router marks packets. The BB located in a domain is responsible for allocating the preferred service to users as requested, and for configuring the network routers with the correct forwarding behavior for the defined service. A BB is associated with a particular trust region, one per domain. Only BB can configure the leaf routers to deliver a particular service to flows. The configuration request might come from another region's BB. The BB is used for allocation of the service profile of MH.

The range of the movement of MH in the Diff-Serv may be categorized into four groups such as the movements between subnetworks in a same domain, between customer domain, between provider domains, and returning to home domain. When MH moves from one subnetwork to another, the leaf router which MH will be connected to is not yet configured according to its service profile. Thus the QoS is not guaranteed.

The following describes the registration procedure for the movement between subnetworks as shown in Fig. 1 [11].

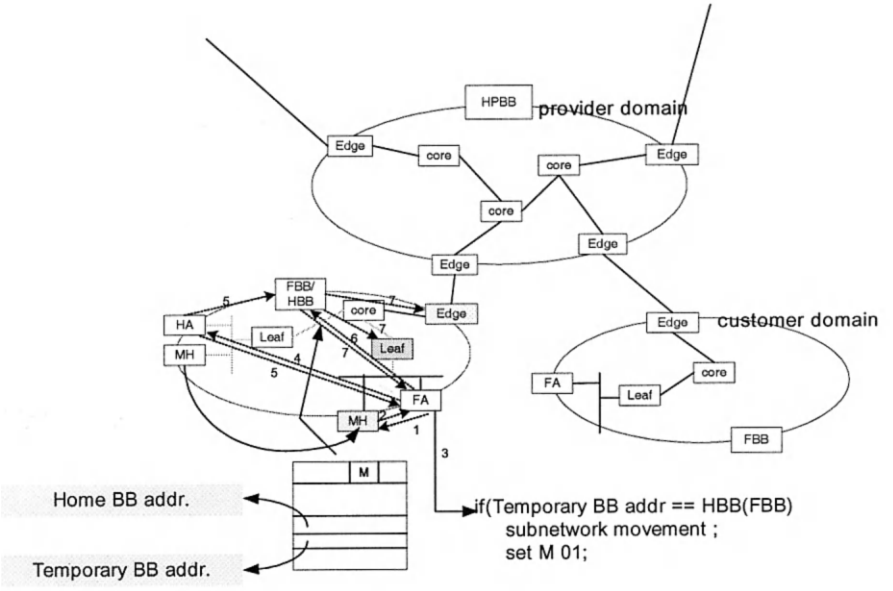


Fig. 1. Registration procedure for the movement of MH

① After MH moves, it detects its movement by the received advertisement messages.

② MH forwards a registration message to a foreign agent (FA). At this time, MH uses a modified registration request message having the address fields of the home BB and the temporary BB. It is assumed that the address of the home BB is already known to MH and the first temporary BB address is set as the home BB address. To obtain the temporary BB address for MH, the FA receiving the registration reply message writes the address of its domain's BB in the message and forwards it to MH. MH records this address and do not update it until receiving another registration reply message which contains a different temporary BB address.

③ If the BB address which is known to the FA is the same as the temporary BB address, a FA determines that MH has moved between subnetworks.

④ If the FA determines that MH has moved between subnetworks, it forwards the registration request message with which the flag *M* is set to value 01 to the HA. The 2 bits of the code field in the registration request/reply message are used for flag *M*. If the flag *M* is not supported in the Diff-Serv, MH forwards the registration request message, where home and temporary BB address fields are filled up zeros, to the FA. Then, the FA compares these two addresses. If they are all zeros, flag *M* is set to 00 and then the general Mobile IP operations are triggered. If it is 10, the movement between (customer/provider) domains is assumed. In case of the movement between subnetworks, MH's profile is already enrolled in the BB. Thus, it is not necessary to acquire MH's profile.

⑤ After the home agent (HA) sent the registration reply message to the FA, to keep CH's service in tunneled path, following procedure is needed. The HA sends the service request to the home BB for the path reconfiguration in the tunneled route. Then, the home BB relays the service request to the home provider BB. If the service request reaches to the BB that is located in a domain where the FA is, the BB makes an admission control decision and comes to configure the classification and policing rules on the leaf and boarder routers. If the request is denied, an error message is sent back to the provider BB. After this procedure is done successfully, the foreign provider BB will configure the classification, policing and reshaping rules on the egress and ingress routers. Then, the home BB and home provider BBs are also set the policing and reshaping rule on the leaf and boarder router.

⑥ If the FA receives the registration reply message to grant the request, it sends the service request message to the foreign BB in place of MH. Then the foreign BB and foreign provider BBs will configure the leaf and boarder routers.

⑦ When the FA receives the reply to grant the service request from the foreign BB, it sends the registration reply message with an address of temporary BB to MH.

4 Performance Analysis

The distances between the various entities involved in the proposed scheme are shown in Fig. 2. This system model is for the performance analysis in the movement between subnetworks.

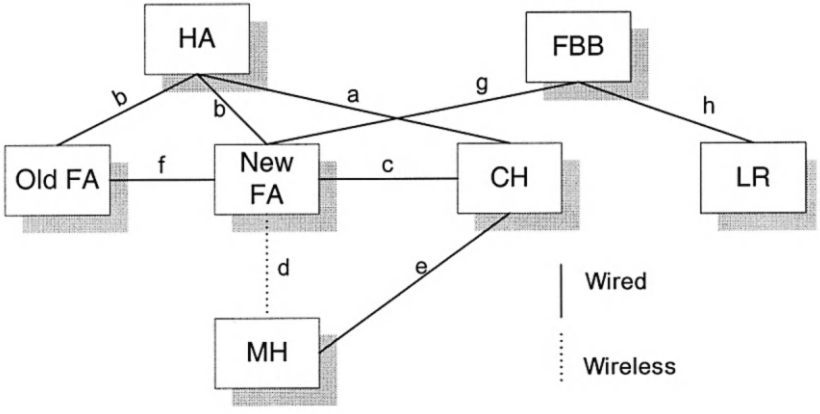


Fig. 2. System model of the proposed scheme

For the proposed scheme, during the time interval when the MH moves between the subnetworks in a domain, the total cost incurred, C_{SMDS} is given by (1).

$$C_{SMDS} = C_{rg} + C_{oldFA} + C_{newFA} + C_{pkt} \tag{1}$$

The cost of MH registering at the new subnetwork, C_{rg} is given by (2). Besides the cost of MH registering in the Mobile IP, the distances and processing cost that takes place in reconfiguring the leaf router must be added.

$$C_{rg} = 2(b+d) + 2(g+h) + 5r + 4r \tag{2}$$

The cost of data packets lost by being delivered to the old FA during the registering delay is given by (3).

$$C_{oldFA} = \alpha_1 \times t_{rg_SMDS} \times C_{dl} \tag{3}$$

C_{dl} in the Mobile IP is $l(a+b+d)+2r$ which means the cost of a single data packet delivered from CH to MH via tunneling at the HA [3]. However, in the proposed scheme, the overhead of traffic conditioning in the boarder router r_c should be added as given in (4). Here, n and r_c mean the number of boarder routers between CH and MH and the average processing cost for traffic conditioning in the boarder router, respectively.

$$C_{dl} = l(a+b+d) + 2r + nr_c \tag{4}$$

For the registration delay, the maximum time among two delivery times $(t_b, t_g + t_h)$ must be added because the delivery of registration message to a HA coincides with the delivery of service request.

$$t_{rg_SMDS} = t_d + \max(t_b, t_g + t_h) + 5t_r + 2t_r \tag{5}$$

The cost of data packets delivered to the MH via a new FA is the same as defined in the Mobile IP. Thus, C_{newFA} becomes same as in the Mobile IP.

The cost of data packets delivered from MH to CH, C_{pkt} is given as follows.

$$C_{pkt} = \rho_2 \times e \times D_{Diff-Serv} \tag{6}$$

The expected delay, $D_{Diff\text{-}Serv}$ and the drop probability in each hop are dependent on the service class, namely Assured and Premium service. In the next, these two parameters are obtained according to the service class.

4.1 Expected Delay and Drop Probability in Assured Service

We assume Random Early Detection with In/Out (RIO) as the buffer management scheme. Assume that the packets arrive in the queue according to a Poisson process with rate λ. The packets are *In* packets with probability p. The packets require a service exponentially distributed with the parameter μ. The RIO accepts all packets until the queue size is equal to $K/2$ and drops a packet with the probability that increases linearly to 10% for tagged packets, and to 90% for non-tagged packets.

$$\alpha^T(n) = \begin{cases} 1 & n \le \dfrac{2}{K} \\ 1 - \dfrac{0.1(2n-K)}{K} & \dfrac{2}{K} \le n \le K-1 \end{cases} \tag{7}$$

We can also derive $\pi(n)$ using the stationary distribution of buffer contents.

$$\pi(n) = \pi(0)\left(\frac{\lambda}{\mu}\right)^n \prod_{i=0}^{n-1} \alpha(i) \tag{8}$$

Let π_{drop}^{T} be the drop probabilities for tagged packets. The drop probability means that the queue has no room for the packets. The probability of rejecting a packet is equal to 1 minus the probability of accepting it in the queue.

$$\pi_{drop}^{T} = 1 - \sum_{n=0}^{K} \alpha^{T}(n)\pi(n) \tag{9}$$

The expected delay in the queue for Assured service is given by (10).

$$D_{Assured} = \frac{1}{\mu} \sum_{n=0}^{K-1} (1+n)\pi(n)\alpha(n) \tag{10}$$

4.2 Expect Delay and Drop Probability in Premium Service

The Premium Service can be modeled as follows. The router includes two separate queues, one with the finite size *K* accessible only to tagged packets. If the tagged queue is full, arriving tagged packets are discarded.

We assume that the input stream is a Poisson with rate λ, that the arriving packets are tagged with probability *p*. The input stream of the high and low buffers are an independent Poisson process with the rate $\lambda_1 = \lambda_p$. Let $\phi_1 = \lambda_1 / \mu$ be the load factor of the tagged queue.

$$\pi(n) = \pi(0)\phi_1^{\,n} \qquad (n \le K) \tag{11}$$

For the high priority customers the response time $D_{Premium}$ is the time of customers in a M/M/1/K queue with arrival rate λ_1. $D_{Premium}$ is obtained using Little's law [12].

$$D_{\mathrm{Pr}emium} = \frac{1}{\mu - \lambda_1} \frac{1 - (K+1)\phi^K + K\phi_1^{K+1}}{1 - \phi_1^{K+1}} \tag{12}$$

We also obtain the loss probability of the high priority class.

$$\pi_1 = \phi_1^K \frac{(1-\phi_1)}{1-\phi_1^{K+1}} \tag{13}$$

5 Results

As results of the performance analysis, C_{MIP}, C_{SMDS}, C_{NMDS}, and the drop probability in the leaf router are obtained. Costs of the proposed scheme are shown as normalized to the cost of the Mobile IP. For the roundtrip time calculation in wired and wireless

links, equations in [3] are used. For parameters, *a=4, b=c=g=h=y=2, d=1* and *e=6* are assumed.

The cost ratio, C_{SMDS} to C_{MIP}, is shown in Fig. 3. It is showing the cost ratio of the proposed scheme to the Mobile IP in case of the movement between subnetworks. Cost of Assured service is approximately 1.95 times higher that that of the Mobile IP. Cost of Premium service is more than 2 times higher. Since the proposed scheme requires the additional operations to the Mobile IP, it is natural that the cost of the proposed scheme becomes higher than that of the Mobile IP. The cost of Premium service is approximately 10% higher than that of Assured service as shown in the Fig. 3. As Packet to Mobility Ratio (PMR) [3] is increasing, the cost is decreased. This is because the amount of the data packet relative to the control packet such as the registration and reply packets is increasing as PMR is increasing.

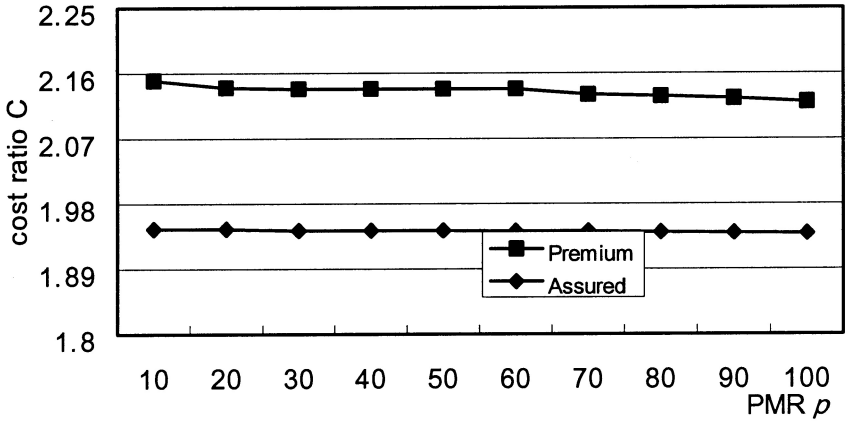


Fig. 3. Cost ratio for the movement of MH

Next, the packet drop probability in the leaf router is shown according to the system load in Fig. 4. The buffer size of 50 and the utilization factor of the queue of 0.5 are assumed. In the plain Diff-Serv architecture, MH's packets will be heavily dropped in the leaf router since the Diff-Serv alone does not support the mobility. However, if the proposed scheme is used, the packet loss may be significantly down. Actually, in Fig. 4, it is confirmed that there is negligible packet drop probability in the practical system traffic load.

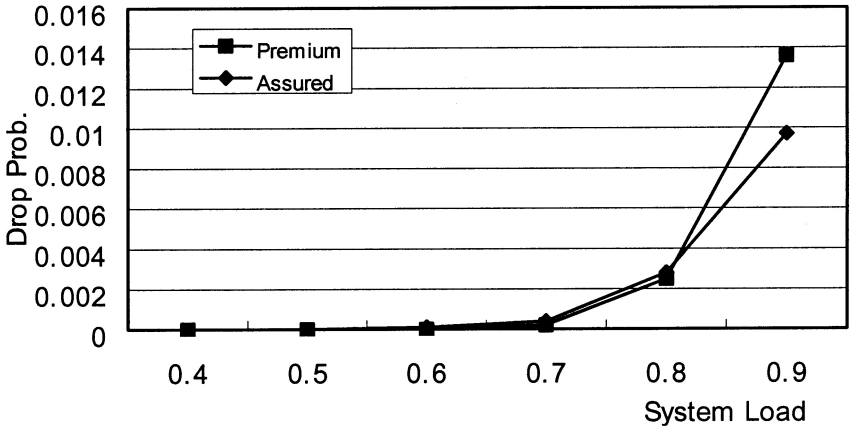


Fig. 4. Packet drop probability

6 Conclusions

Over the past several years there has been a considerable amount of research to provide QoS for the real time and multimedia data service in the internet. Among these researches Differentiated Service model is considered as the practical choice having scalability. However, since this model only considers fixed hosts, mobility service can not be supported.

To support the quality of service and the mobility service in the Internet simultaneously we propose a new scheme, namely the proposed scheme. It is basically a mechanism to support the Mobile IP in the Diff-Serv architecture. Performance analysis is performed to see the cost of the proposed scheme when compared with the cost of the Mobile IP without the Diff-Serv. Through the performance analysis it is shown that the cost varies around 2 times of that of the Mobile IP. Premium service shows higher costs than Assured service does. Packet drop probability shows that there is little packet drop probability for mobile communications in the practical system load if the proposed scheme is used.

References

1. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss: An Architecture for Differentiated Services, RFC 2475, Dec. 1998.
2. C. Perkins: IP Mobility Support, RFC 2002, Oct. 1996.
3. R. Jain, T. Raleigh, C. Graff and M. Bereschinsky: Mobile Internet Access and QoS Guarantees using Mobile IP and RSVP with Location Registers, in Proc. ICC'98 Conf., pp. 1690–1695, Atlanta.
4. J. Postel: Service Mappings, RFC 795, Sept. 1981.

5. K. Nichols et al.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, Dec. 1998.
6. V. Jacobson and K. Poduri: An Expedited Forwarding PHB, RFC2598, June 1999.
7. J. Heinanen, F. Baker and J. Wroclawski: Assured Forwarding PHB Group, RFC 2597, 1999.
8. M. May, J.-C. Bolot, A. Jean-Marie and C. Diot: Simple Performance Models of Differentiated Services Schemes for the Internet, in Proc. INFOCOM'99, pp. 1385–1394, New York.
9. K. Nichols, V. Jacobson and L. Zhang: A Two-Bit Differentiated Services Architecture for the Internet, Internet draft draft-nichols-dif-srv-arch-00.txt, Nov. 1997.
10. X. Xiao and L.M. Ni: Internet QoS: A Big Picture, IEEE Network Magazine, March/April, 1999.
11. Y. Mun and J. Kim: Mobility Support between Subnetworks in Differentiated Services, in Proc. PDPTA 2000, pp. 2079–2085, Las Vegas.
12. L. Kleinrock: Queueing Systems, vol.1. J. Wiley & Sons, 1975.

Performance Analysis for Real-Time Grid Systems on COTS Operating Systems

Eui-Nam Huh[1] and Y. Mun[2]

[1] Seoul Women's University, School of Computer Science, Seoul, Korea
huh@swu.ac.kr
[2] Soongsil University, School of Computer Science, Seoul, Korea
mun@computing.ssu.ac.kr

Abstract. Computer technology for communication has become an integral aspect of daily operation. The exponential growth of internet services with dynamic inquiries in such areas as manufacturing, business, air traffic control and mission critical systems demands that there be quick, reliable and safe use of services. Each service must contain QoS metrics to assure security, performance and accuracy. A new paradigm of resource management middleware techniques is in this paper presented which can provide QoS for dynamic, distributed real-time systems on Common Off The Shelf (COTS) operating systems. Accommodation of dynamic environments enables the middleware to carefully consider an efficient design of resource profiling, resource needs estimation, resource unification, and performance analysis (or compliant with schedulability analysis) infrastructure providing significant benefits for QoS management on COTS operating systems. First, the use of low-cost COTS systems is extended to real-time computing without changing the operating system. Further, experiments for response time analysis confirm that the worst-case analysis poorly utilizes computational resources. Finally, it is shown that the new method of middleware design employing scalability of software and hardware system can be easily applied to legacy systems to manage resources efficiently for quick, reliable services and accurate QoS.

1 Introduction

Computer Technology for communication has become an integral aspect of daily operation. The exponential growth of internet services with dynamic inquiries in such areas as manufacturing, business, air traffic control and mission critical systems demands that there be quick, reliable and safe use of services. Each service must contain QoS metrics to assure security, performance and accuracy.

Generally, the real-time system is designed to predict response times of applications before they are executed in order to meet the QoS requirement in the metric of timeliness. The Rate Monotonic Analysis (RMA) introduced by [1] is used primarily to determine schedulability of an application by using *a priori* Worst-Case Execution Time (WCET) and the priority of the application. The priority of the application to be applied to RMA is dependent upon arrival patterns and rates. The

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 482–490, 2003.
© Springer-Verlag Berlin Heidelberg 2003

application, which has a higher arrival rate of task or needs to be executed more frequently, has a higher priority level than any other applications.

However, as has been noted in [2], and [3] resources are poorly utilized if the average case is significantly less than the worst case. Another drawback of RMA is that it cannot *efficiently* accommodate high-priority jobs that have relatively low rates. It must, however, be noted that RMA can be made to work in such cases, by transforming low-rate, high-priority jobs into high-rate jobs – but this can be extremely wasteful in terms of resources.

It is stated in [4], and [3] that accurately measuring the WCET is often difficult, and is sometimes impossible. Puschner and Burns in [5] consider WCET analysis to be hardware dependent, making it an expensive operation on distributed heteogeneous clusters.

Statistical RMA in [6] considers tasks that have variable execution times and allocate resources to handle the expected case. The benefit of the approach is the efficient utilization of resources. However, there are shortcomings. First, applications which have a wide variance in resource requirements cannot be characterized accurately by a time-invariant statistical distribution; and secondly, deadline violations occur when the expected case is less than the actual case. Similarly, real-time queuing theory [7] uses probabilistic event and data arrival rates for performing resource a1location analysis. On the average, it provides good utilization of resources. It must be noted, however, that applications which have a wide variance of resource requirements cannot be characterized accurately by a time-invariant statistical distribution

Use of distributed systems appears to provide strong integration for quick services. However, distributed systems have heterogeneous resources. That is, they have different capacities. Thus, when data or tasks are distributed to heterogeneous systems to be processed concurrently, the load of data should be balanced. A major difference between the traditional load-balancing techniques demonstrated by [8] and the dynamic QoS-based resource management services lies in the overall goals. While load-balancing systems[1] attempt to achieve *system* performance goals such as minimized response time or maximized throughput, the dynamic QoS-based resource management service strives to meet the QoS requirements of each application being managed. Another major difference between these systems is their workload models. Traditional load balancing systems assume *independent* jobs with *known resource requirements*. In dynamic resource management systems, the workload requirements of applications are *dependent* (communicate with each other) and can vary, based on environmental conditions.

Presently, there is an increasing amount of research focused on QoS management and resource management techniques using distributed real-time systems. However, most of the Commercial Off The Shelf (COTS) systems use general purpose operating systems such as Windows, Unix, and Linux rather than real-time operating systems such as MACH, RT-Kernel, and Lynx.

Therefore, use of distributed systems with COTS operating systems to handle the QoS and resource management for the dynamic real-time system becomes a new

[1] See Load-Leveler (IBM 1993), Globus (http://www.globus.org), Beowulf (Becker et al 1995), and Condor (http://www.cs.wisc.edu/condor/manual).

challenge to handle complex problems. The COTS operating systems, especially, has become widely used in many application fields (especially in internet based applications) supporting QoS.

2 Contention Analysis for Tasks with Same Priority Level

The benefit of use of COTS operating systems can be extended, if they can be utilized for real-time tasks. With the small number of priority levels in COTS operating systems, tasks may often have the same priority which means they will use CPU time quanta (denoted TQ) equivalently.

Furthermore, dynamic environments with the unknown arrival times of processes make it difficult to predict the exact response time of the processes (or tasks). Thus, in the following section, the techniques which are the *worst-case* analysis and probabilistic analysis to predict response times of processes are studied.

2.1 Worst-Case

The *worst-case* of contention occurs when all tasks arrive at the same time and the target task starts last. The *worst-case* queuing delay of a task due to the same priority tasks can be computed. This approach considers the period of the tasks and the segments of execution times of all processes. The term, $MC(a_{xyz}, a_{ijk}, tl, H_q)$, is the modified execution time that an application of workload ($tl(a_{xyz})$) will contend with the target task (a_{ijk}) on the host (H_q). The queuing delay of a_{ijk} on H_q due to the same priority tasks, $SD_{pred}(a_{ijk},tl,H_q)$, is the approximated response time, minus execution time, as shown in formula (1).

$$SD_{pred}(a_{ijk},tl,H_q)=ar_i(a_{ijk})-C_{pred}(a_{ijk},tl,H_q),\ if\ ar_i(a_{ijk})=ar_{i+1}(a_{ijk}),\ \forall I \quad (1)$$

The i^{th} approximated response time, ar_i, starts with 0 and adds execution times of all same priority tasks within the period. The following steps show the computation process of the approximated response time.

Step 1. Compute the first approximation of response time (ar_0) by adding the time range of other tasks, $MC(a_{xyz},a_{ijk},tl2, H_q)$.

$$ar_0(a_{ijk})= C_{pred}(a_{ijk},tl,H_q) + \sum_x \sum_y \sum_z MC(a_{xyz},a_{ijk},tl2,H_q)$$

$$MC(a_{xyz},a_{ijk},tl2,H_q) = \begin{cases} SEG(a_{ijk},tl,H_q)*TQ(H_q), & if\ SEG(a_{xyz},tl2,H_q) > SEG(a_{ijk},tl,H_q) \\ C_{pred}(a_{xyz},tl2,H_q), & \text{if } SEG(a_{xyz},tl2,Hq) \le SEG(a_{ijk},tl,H_q) \end{cases}$$

$$where\ tl=tl(a_{ijk},\ c+1),\ Host(a_{xyz},\ t)= Host(a_{ijk},\ t),\ p(a_{xyz})=p(a_{ijk}),\ and\ x \neq i,\ \forall x,y,z$$

The conditions of other tasks (a_{xyz}) are as follows:

1) Should have the same priorities of tasks as the target task *($p(a_{ijk})=p(a_{xyz})$),*
2) Should run on the same host ($Host(a_{xyz}, t)= Host(a_{ijk}, t)$),
3) Should be on a different path from the target task ($x \neq i$).

From condition 1), it can be seen that the execution times of tasks need to be re-computed due to the equivalent chance of execution among other tasks, which have a different required execution time. Because the $MC(a_{xyz},a_{ijk},tl2,H_q)$ depends on the number of segments of the target task, $SEG(a_{ijk},tl,H_q)$ $(=C_{pred}(a_{ijk},tl,H_q))/TQ(p,H_q))$, the segments of other tasks become the same as the segment of the target task, even though original execution times of other tasks may be greater than that of the target. The extra segments of the other tasks do not affect the response time of the target task.

In Step 2, the next approximation is performed to determine additional execution time required. If the periods of the tasks are smaller than the previous approximated response time ($ar_i(a_{ijk})/T(a_{xyz}) \geq 1$), additional executions occur. Otherwise, the next period does not appear.

Step 2. Compute the next approximation of the response time by using the previous approximation.

$$ar_{i+1}(a_{ijk})=C_{pred}(a_{ijk},tl,H_q) + \sum_x \sum_y \sum_z \left\lceil \frac{ar_i(a_{ijk})}{T(a_{xyz})} \right\rceil * MC(a_{xyz},a_{ijk},tl2,H_q)$$

The ceiling function determines if the period of a task is within the previous approximation. If all task periods are smaller than the previous approximation, none of the tasks starts during the approximation. Thus, Step 3 performs is necessary for finding the answer.

Step 3. Determine if the approximation is the answer.

if $(ar_i(a_{ijk}) == ar_{i+1}(a_{ijk}))$, $SD_{pred}(a_{ijk},tl,H_q) = ar_i(a_{ijk}) - C_{pred}(a_{ijk},tl,H_q)$
else increase i and repeat step 2

2.2 Rate of Progress (RP)

Welch et al (1995) introduced two approaches for estimating the contention using progress rate during the least common multiple (LCM) of periods of tasks, and for resource usage. These approaches were modified and applied to the dynamic real-time systems.

2.2.1 The First Rate of Progress Technique (RP-1)

The following formula (2) called ***RP-1,*** shows how it is processed to compute the queuing delay of the target application (a_{ijk}) due to the same priority tasks.

$$SD_{pred}(a_{ijk},tl,H_q) = \left(\frac{SEG(a_{ijk},tl,H_q)}{prate\ 1(a_{ijk})} - SEG(a_{ijk},tl,H_q) \right) \times TQ(p,H_q) \quad (2)$$

In this approach, during LCM, the progress rate of the target task, *prate1*(a_{ijk}), is computed by considering the segments of the task that have the same priority. It is the ratio of execution segments of the target task versus other tasks (a_{xyz}), which will contend the target task. The *SEG*(a_{ijk},*tl*,H_q)/*prate1*(a_{ijk}) is the total number of segments considering contention. Hence, by subtracting required segments, *SEG*(a_{ijk},*tl*,H_q), the contended segments due to the other tasks are produced. Finally, the contended segments are converted into time unit ($SD_{pred}(a_{ijk},tl,H_q)$: queuing delay) by multiplying the time quantum (*TQ*(*p*,H_q)).

In Step 1, LCM is computed by using periods of all task on the same host with the same priority, and during the LCM, required resource segments, ($S(a_{xyz},tl,H_q)$) for the target task on H_q are calculated.

Step 1. Compute the resource requirement of the target task (a_{ijk}) during the interval, [0,LCM].

The required resource segments for other tasks on H_q ($S(H_q,a_{ijk})$) in [0,LCM] can also be computed as in Step 1. The *LCM*/*T*(a_{ijk}) computes the number of arrivals of a_{ijk} during the LCM.

$$S(a_{ijk}, tl, H_q) = SEG(a_{ijk}, tl, H_q) \times \frac{LCM}{T(a_{ijk})}$$

Step 2. Compute the resource requirement of the other tasks (a_{xyz}) during the interval, [0,LCM].

$$S(H_q, a_{ijk}) = \sum_x \sum_y \sum_z SEG(a_{xyz}, tl, H_q) \times \frac{LCM}{T(a_{xyz})}$$

$$where,\ tl=tl(a_{ijk},\ c+1),\ a_{xyz} \in AL(H_q,t),\ x \neq i,\ p(a_{xyz})=p(a_{ijk}),$$
$$and\ Host(a_{ijk},t) = Host(a_{xyz},t)\ \forall x,y,z$$

It must be noted that the conditions of other tasks (a_{xyz}) are the same as the *worst-case*. For condition 3) in section 4.1, the applications in the same path as the target task, a_{ijk}, are ignored, as a_{ijk} can start after successor (*succ*(a_{ijk})) is finished, and the predecessor of the target task, (*pred*(a_{ijk})) can start after a_{ijk} is finished. It can be said that they are execution dependent.

$S(H_q, a_{ijk})$ plus $S(a_{ijk},tl, H_q)$ should be less than LCM; otherwise, tasks can not meet time requirements. The resource demands for the LCM is computed at Step 1 and Step 2. In Step 3, rate of progress (*prate1*(a_{ijk})), a metric to account contention, is computed simply by comparing the relative ratio of required resources.

Step 3. Compute rate at which requests of a_{ijk} for resource H_p are serviced.

$$prate1(a_{ijk}) = \frac{S(a_{ijk}, tl, H_q)}{S(H_p, a_{ijk}) + S(a_{ijk}, tl, H_q)}$$

2.2.2 The Second Rate of Progress Technique

An improved approach called ***RP-2*** considers the probability of the target task being blocked by any other tasks. The basic comcept is the same as ***RP-1*** introduced in section 2.2.1. The improved progress rate, $prate2(a_{ijk})$, is presented in this section. This technique considers all possible probability that the target task could progress, including computation of progress rate within contention with other tasks. The following formula (3) shows the computation of the queuing delay of the target task, a_{ijk}, on H_q, ($SD_{pred}(a_{ijk},tl,H_q)$).

$$SD_{pred}(a_{ijk}, tl, H_q) = \left(\frac{SEG(a_{ijk}, tl, H_q)}{prate2(a_{ijk})} - SEG(a_{ijk}, tl, H_q) \right) \times TQ(p, H_q) \tag{3}$$

To compute $prate2(a_{ijk})$, utilization of the target task ($U(a_{ijk},tl,H_q)$) and utilization of other tasks ($U(H_q,a_{ijk})$) on resource H_q are computed at Step 1. Welch et al (1995) compute the utilization of tasks over the LCM, but in this study, utilization of a task is computed over the period ($T(a_{ijk})$) of the task. The results are the same.

Step 1. Compute utilization of the target and other tasks.

$$CUP(a_{ijk}, tl, H_q) = C_{pred}(a_{ijk}, tl, H_q) \times \frac{1}{T(a_{ijk})}$$

$$CUP(H_q, a_{ijk}) = \sum_x \sum_y \sum_z C_{pred}(a_{xyz}, tl, H_q) \times \frac{1}{T(a_{xyz})}$$

$$where,\ tl=tl(a_{ijk},\ c+1),\ x \neq i,\ p(a_{xyz})=p(a_{ijk}),$$
$$and\ Host(a_{ijk},t) = Host(a_{xyz},t)\ \forall x,y,z$$

The condition of other tasks are the same as the *worst-case* and the ***RP-1***. In Step 2, the improved rate of progress is computed by considering: 1) the probability of being with no usage of the resource ($cp_0(a_{ijk})$), 2) the chance that only a_{ijk} uses the resource ($cp_1(a_{ijk})$), and 3) the chance that there is a_{ijk} progress rate within contention ($cp_2(a_{ijk})$).

Step 2. Compute progress rate

$$prate2(a_{ijk}) = cp_0(a_{ijk}) + cp_1(a_{ijk}) + cp_2(a_{ijk})$$

$$cp_0(a_{ijk}) = 1 - (U(H_q, a_{ijk}) + U(a_{ijk}, tl, H_q)) = CIP(H_q, t)$$
$$cp_1(a_{ijk}) = U(a_{ijk}, tl, H_q)$$
$$cp_2(a_{ijk}) = U(H_q, a_{ijk}) \times prate1(a_{ijk})$$

3 Contention Analysis for Tasks at Different Priority Levels

Priority scheduling in distributed environments is a sub-problem of distributed resource management. At the present time, real-time priority services on COTS operating systems can provide powerful means to improve performance of tasks requiring time constraints. As noted in the literature review, different aspects of the priority management problem have been addressed in a number of studies. Especially, in [9], fixed-priority assignment and static-task allocations appear to be the most common approaches employed by resource management systems in distributed real-time systems. However, the problem of the queuing delay estimation for the dynamic real-time systems using real-time priority level on COTS operating systems in order to manage QoS has not heretofore been addressed before. This section, computation of queuing delays due to the higher real-time priority tasks, $HD_{pred}(a_{ijk},tl,H_p)$, is ignored due to the page limit. Shortly, as used in section 2, all formulas are identical as previous sections except tasks' priorities considered for computation of process contention.

4 Synthesizing Contention Analysis for Response Time Prediction

In this section, the final prediction techniques for COTS operating systems under dynamic environments by synthesizing previous approaches are presented. These techniques are used for same priority level as well as for different priority levels. Therefore, the final response time of a_{ijk} ($= \lambda_{pred}(a_{ijk},tl,H_q)$) is derived by the following formula (4):

$$\begin{aligned} \lambda_{pred}(a_{ijk},tl,H_p) &= C_{pred}(a_{ijk},tl,H_p) + D_{pred}(a_{ijk},tl,H_p) \\ D_{pred}(a_{ijk},tl,H_p) &= SD_{pred}(a_{ijk},tl,H_p) + HD_{pred}(a_{ijk},tl,H_p) \end{aligned} \tag{4}$$

The following Table 1 shows possible choices (s1 to s13) for synthesizing $SD_{pred}(a_{ijk},tl,H_p)$ and $HD_{pred}(a_{ijk},tl,H_p)$ computed by 5 different approaches for each. The same technique used for computing $SD_{pred}(a_{ijk},tl,H_p)$ and $HD_{pred}(a_{ijk},tl,H_p)$, shown in diagonal of Table 1, might be possible selections for synthesizing to compute total response time. Combination of the *worst-case* and other probabilistic approaches like RP-2 also might be a good candidate..

Table 1. The synthesized cases for the final response time prediction.

Same priority / *Different priority*	*Worst-Case*	*RP-1*	*RP-2*
Worst-Case	S1	S6	S7
RP-1	S10	S2	
RP-2	S11		S3

5 Experiments

In experiment, the filter application in DynBench [10] is also used as a target task examined on the dynamic and monotonic workload scenario. Several contention conditions derived by periods and workloads are considered in this experiment.

As Shown in Fig. 1 the combined scenario brings more variations in response time prediction, even though CPU is only used up to 65 percent. The "s1" overuses the CPU resource 29 percent in this experiment, while "s3" exceeds 2.8 percent only. Fig. 1 shows the comparison of each synthesized prediction technique.

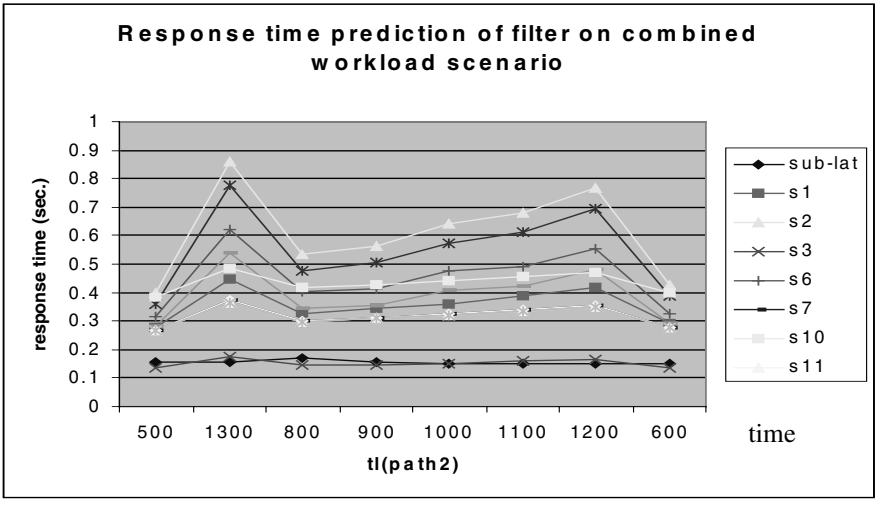


Fig. 1. Response time prediction by synthesized techniques on the combined workload

From the experiment presented, there is no measurement error under the 95% confidence interval. The “s3”, probabilistic contention analysis technique, RP-2, can accurately predict response time of an application on a COTS operating system, while the “s1” and the “s2” involving the *worst-case* analysis poorly predict response times. Therefore, these experiments for the dynamic real-time systems give strong analysis that the *worst-case* analysis poorly utilizes computational resources, and new approaches can predict response time on the COTS operating system with the dynamic environment constraints using current utilization. This suggest that schedulability of tasks can be predicted without using any real-time operating system components.

6 Conclusion

A new paradigm of the dynamic resource management technique was the focus of this research. This work mainly contributes to the area of QoS with an adaptive RM architecture, middleware and framework. This is accomplished with a mathematical

model for dynamic real-time systems, an execution time analysis, and a predictive response time analysis using real-time priority services on COTS operating systems for dynamic distributed real-time systems. The significant experiment results and solution procedures are deployed in analytical models. A statistical analysis tool was used to evaluate the accuracy of these new approaches compared with observed data and previous approaches.

Acknowledgement. This work was supported in part by the Korea Science and Engineering Foundation (R01-2001-000-00362-0) and by the Ministry of Information Communication of Korea, under the "Support Project of University Information Technology Research Center (ITRC)" supervised by KIPA.

References

1. Liu, C. L., and J.W. Layland, 1973. Scheduling algorithms for multi-programming in a hard-real-time environment. Journal of the ACM, Vol. 20: 46–61.
2. Ramamritham, J.A. Stankovic and W. Zhao, "Distributed scheduling of tasks with deadlines and resource requirements," *IEEE Transactions on Computers, Vol.* 38(8), 110–123, August 1989.
3. Abeni, L. and G. Buttazzo, 1998. Integrating multimedia applications in hard real-time systems. *Proceedings of the 19th IEEE Real-Time Systems Symposium,* IEEE Computer Society Press: 3–13.
4. Stewart, D.B. and P.K. Khosla, 1997. Mechanisms for detecting and handling timing errors. *Communications of the ACM , Vol.* 40(1): 87–93.
5. Puschner, Peter and Alan Burns, 2000. A Review of Worst-Case Analysis. *The International Journal of Time-Critical Computing Systems*, Vol. 18: 115–128.
6. Atlas, A., and A. Bestavros, 1998. Statistical rate monotonic scheduling. *Proceedings of the 19th IEEE Real-Time Systems Symposium,* IEEE Computer Society Press: 123–132.
7. Lehoczky, John P., 1996. Real-Time Queueing Theory, *Proceedings of IEEE Real-Time Systems Symposium*, IEEE CS Press : 186–195.
8. Shirazi, B., A. R. Hurson, and K. Kavi, 1995. *Scheduling and Load Balancing in Parallel and Distributed Systems*, EEE Press.
9. Lehoczky, J.P, 1990. Fixed Priority Scheduling of Period Task Sets with Arbitrary Deadlines. *Proceedings of IEEE Real-Time System Symposium*, Los Alamitos, CA. IEEE computer society press.
10. Welch, L. R. and B. Shirazi, 1999. A Dynamic Real-Time Benchmark for Assessment of QoS and Resource Management Technology. *IEEE Real-time Application System*.

Scalable Keyframe Extraction Using One-Class Support Vector Machine

YoungSik Choi[1] and Sangyoun Lee[2]

[1]Department of Computer Engineering, Hankuk Aviation University,
200-1 Hwajun-Dong Dukyang-Gu
Kyonggi-Do, Koyang-City, Korea
choimail@hau.ac.kr
[2]Service Development Laboratory, Korea Telecom,
17 Woomyeon-Dong Seocho-Gu
Seoul, Korea,
leesy@kt.co.kr

Abstract. In this paper, we present a scalable keyframe extraction method using one-class support vector machine. Keyframe extraction seeks to generate "good" images that best represent underlying video content and provide content-based access points. Criteria for "good" images play a major role for keyframe extraction process. Extracting "good images" can be viewed as detecting "novel images" among all the frames within a video. Therefore, keyframe extraction reduces to novelty detection problem. We describe how to efficiently solve the novelty detection problem using one-class support vector machine. We also present an algorithm of extracting keyframes in a scalable way so that one can access a video from coarse to fine resolution. We demonstrate the performance of our algorithm on several different types of videos.

1 Introduction

The advances in video data capturing, compression, storage, and communication technologies have made vast amounts of video data available in consumer and enterprise applications. This phenomenal growth of video data has brought the need for efficient video access mechanism. Efficient video access requires more than connecting with data banks and delivering data via networks. It requires improving the accessibility and usability of video data to the point that one can efficiently and effectively search, browse, organize, and manage video data as textual data. This challenge has attracted researchers from various disciplines and has formed a new research area, so called video content analysis and retrieval. Various applications involving education, entertainment, journalism, medical video libraries, and multimedia information services can benefit from this emerging technology.

Among various research areas in video content analysis and retrieval, video abstraction is one of the most important research topics [8]. Video abstraction is the process of creating an abridged version of video, which should be much smaller and

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 491–499, 2003.
© Springer-Verlag Berlin Heidelberg 2003

still preserve essential information about the original video content. This abstraction process is similar to extracting keywords or creating summaries in text document processing. Video abstraction is especially useful and important when even a short video of a few minutes' duration has the vast amount of data.

There exist several methods for abstracting video content: skims, highlights, and summaries. A video skim is a condensed representation of the original video, containing image and audio sequences [4]. Highlights only convey interesting parts of the video and thus involve detection of particular events in the video [8]. A video summary extracts structural and semantic information about the video, and normally represented by a collection of keyframes [1][2][3]. The temporal order of extracted keyframes can be visualized in a spatial domain so that one can quickly grasp the main content of video. Therefore, keyframe extraction plays an important role in a video summary. This paper will focus on the keyframe extraction schemes.

Keyframe extraction seeks to select good images that best represent the underlying video content and provide content-based access points to video content. The challenge is how to automatically determine which frames are most representative for a given video. The representational power of a video summary largely depends on the criteria for selecting keyframes. The criteria for keyframes vary with respect to the target video at which the video summarization methods aim. The process of keyframe extraction for long videos focuses on extracting semantic and structural information about the whole video content. Extracted keyframes are a collection of images, probably in a hierarchical order that represent events, scenes, or stories. On the other hand, the keyframe extraction process for relatively short videos seeks to select a small number of images that best represent dynamic visual content. Various applications can benefit from this approach. For example, generating thumbnail images for video clips in web-search services can eliminate the painstaking downloading of the entire video clips to check whether the retrieved video clips are the desired ones. Furthermore, condensing video message will be valuable especially in wireless multimedia messaging services.

In [13], the video is segmented into shots and then the first frame of each shot is determined as a keyframe. Although this approach seems a natural way of extracting keyframes, the number of keyframes for each shot is limited to one, regardless of the visual complexity of the shot. That is, a shot boundary image is not necessarily representative for the rest of images. Several algorithms have been proposed to overcome this limitation [11]. Another class of keyframe extraction is to choose the keyframes based on motion metric. In [10], the optical flow of each flow is first computed, and then a simple motion metric is computed. Finally, by analyzing the metric as a function of time, the frames at the local minima of motion are selected as the keyframes. In [14], a domain specific keyframe extraction method is presented using sophisticated global motion and gesture analysis. Mosaic-based keyframe generation is based on detecting specific camera motions [15]. In [9], temporal variations of feature vectors are used to select keyframes. First, several features are extracted from each frame, forming a feature vector trajectory. Then, the frames of local minimal and maximal curvatures in the feature trajectory are chosen as the keyframes. The curvatures are computed as the magnitude of the second derivative of the feature vector trajectory.

Selecting the representative frames can be viewed as detecting the novel frames that best describe visual content. The representatives are novel in that not all frames within a sequence are descriptive, but only some of them descriptive. Therefore, the

keyframe extraction problem becomes the novelty detection problem. In this paper, we propose a scalable shot-based keyframe extraction method based on one-class support vector machine [7]. We first define a measure for compatibility of a frame with its neighboring frames using the temporal variations of its visual features. We select the frames of locally maximal compatibility measure as the novel frames. We next extract the keyframes out of the selected frames according to the novelty of their visual features. Detection of novel features is made by one class support vector machine, which is well known for novelty detection problems [6]. One-class support vector machine can find the feature vectors on the surface of sphere with minimal radius, which encloses all the feature vectors. The feature vectors on the surface are called support vectors. We choose the frames as the keyframes whose features are mapped into the surface of the smallest sphere. Furthermore, we can peel off the surface and obtain the support vectors from yet another enclosing sphere, smaller than the previous. The new support vectors can be added up to the previously obtained support vectors. In this manner, we can obtain a series of keyframe sets, each of which represents video content in its own level of detail.

In the rest of the paper, we briefly describe the one class support vector machine and present the proposed method. We also present the experimental results of our algorithm on several different video clips.

2 One Class Support Vector Machine

A classical unsupervised learning is density estimation. Assuming that the unlabeled observations $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$ were generated i.i.d according to some unknown distribution, the task is to estimate its density. However, there are several difficulties to this task. First, a density need not always exist: there are distributions that do not possess a density. Second, estimating densities is known to be a hard task. In many applications it is enough to estimate the support of a data distribution instead of the full density. One class SVMs avoids solving the harder density estimation problem and concentrate on the simpler task [6], i.e. estimating quantities of the distribution, i.e. its support. So far there are two independent algorithms to solve the problem in a kernel feature space. They differ slightly in spirit and geometric notation [5][6]. For brevity, we will focus the approach of [5] as it is more in the line of this paper.

Suppose we are given a data set containing N points, $S = \{\mathbf{x}_j, j = 1, \ldots N\}$, with $S \subseteq \mathrm{X}$ and $\mathrm{X} \subseteq \mathrm{R}^d$. Using non-linear transform ϕ from X to some high dimensional feature-space, one-class SVM seeks the smallest sphere of radius R, enclosing all the points $\phi(\mathbf{x}_j)$. This is described by the constraints.

$$\left\|\phi(\mathbf{x}_j) - \mathbf{a}\right\|^2 \le R^2 \quad \forall j,$$

where $\|\cdot\|$ is the Euclidean norm and $\mathbf{a}$ is the center of the sphere. Introducing slack variable ξ_j forms soft constraints.

$$\left\|\phi(\mathbf{x}_j) - \mathbf{a}\right\|^2 \le R^2 + \xi_j \quad \forall j \tag{1}$$

with $\xi_j \geq 0$. One-class SVM solves the problem by minimizing the following objective function with inequality constraints (1).

$$R^2 + C\sum \xi_j \tag{2}$$

where C is a constant and $C\Sigma\xi_j$ is the penalty term. One can solve this problem introducing the Lagrangian multipliers $\beta_j \geq 0$ and $\mu_j \geq 0$ as follows.

$$L = R^2 - \sum_j (R^2 + \xi_j - \left\|\phi(\mathbf{x}_j) - \mathbf{a}\right\|^2)\beta_j - \sum_j \xi_j \mu_j + C\sum_j \xi_j \tag{3}$$

Setting to zero the derivative of L with respect to R, $\mathbf{a}$, and ξ_j, respectively, results in

$$\sum_j \beta_j = 1 \tag{4}$$

$$\mathbf{a} = \sum_j \beta_j \phi(\mathbf{x}_j) \tag{5}$$

$$\beta_j = C - \mu_j . \tag{6}$$

The KKT complementary conditions lead to

$$\xi_j \mu_j = 0 , \tag{7}$$

$$(R^2 + \xi_j - \left\|\phi(x_j) - a\right\|^2)\beta_j = 0 . \tag{8}$$

From equation (6) we have $0 \leq \beta_j \leq C$. If β_j is zero, μ_j is C and ξ_j is zero. From (8), $R^2 - \left\|\phi(\mathbf{x}_j) - \mathbf{a}\right\|^2 \geq 0$ and thus point $\mathbf{x}_j$ is inside or on the surface of the sphere. If $0 < \beta_j < C$, $0 < \mu_j <$ C and ξ_j is zero. From (8), $R^2 - \left\|\phi(\mathbf{x}_j) - \mathbf{a}\right\| = 0$, and therefore, point $\mathbf{x}_j$ is on the surface of the sphere. Such a point will be referred to as a support vector. If $\beta_j = C$, μ_j is zero and $\xi_j \geq 0$. Then, $R^2 - \left\|\phi(\mathbf{x}_j) - \mathbf{a}\right\| \leq 0$ from (8) and thus, point $\mathbf{x}_j$ lies outside the sphere. This will be called a bounded support vector.

One can reformulate equation (3) as a function of the variables β_j by substituting equations (4), (5), and (6) into L. This leads to

$$W = \sum_j \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_j)\beta_j - \sum_{i,j} \beta_i \beta_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) , \tag{9}$$

with $0 \leq \beta_j \leq C$, (4), and (5). One can replace the dot product in equation (9) by a Mercer Kernel. Throughout the paper, we use the Gaussian kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-0.5\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / \sigma^2) , \tag{10}$$

with a scale parameter σ^2.

One can solve equation (9) using the Quadratic Programming. However, we use the SMO-like (Sequential Minimal Optimization) method [7] since (9) has relatively

simple constraints. The SMO repeats the procedure of selecting two training points and updating the corresponding β_j's with box constraints [7] until all training points satisfy the KKT conditions. We next briefly describe the elementary updating rules and the overall optimization procedure.

Suppose that we chose two points whose β's are β_1^* and β_2^*, respectively. We want to update the two variables so as to optimize equation (9) while unchanging the sum of the two variables. That is, we want not to violate the linear constraint (4) while updating the variables. Therefore, equation (9) becomes a function of β_2. Note that the equation can be re-written as a function of β_2 using the fact that $\beta_1 + \beta_2 = \beta_1^* + \beta_2^*$ = constant. Setting to zero the derivative of (9) with respect to β_2 yields

$$\beta_2 = \beta_2^* + \frac{O_1 - O_2}{K_{11} + K_{22} - 2K_{12}}, \tag{11}$$

where $O_i = \sum_j \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$ and $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. SMO selects a first β for the elementary optimization step in one of the following two ways: (1) SMO scan over the entire data set until it finds a variable violating a KKT condition. Then, SMO choose the next variable β_j according to $j = \arg\max |O_i - O_j|$. (2) Same as (1) except that SMO scans over non-bounded support vectors. Refer to [7] for the detailed procedure.

3 Scalable Keyframe Extraction

The scalable keyframe extraction method proposed in this paper consists of three phases: (1) Selecting the locally representative frames based on a compatibility measure, (2) Detecting the novel frames among the frames obtained from phase (1) using one-class SVM, and (3) Constructing a scalable keyframe-based video summary.

3.1 Compatibility Measure

We denote a given frame sequence as $\{s_1, \ldots, s_N\}$ and its corresponding visual feature vector sequence as $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where s_i and $\mathbf{x}_i$ are i-th frame and its visual feature vector, respectively, and N is the total number of frames within a given sequence. We define the compatibility measure C_i of frame $\mathbf{x}_i$ with its neighbors as follows:

$$C_i = \left(\sum_{j \in W} \frac{1}{|W|} \phi(\mathbf{x}_j) \right) \cdot \phi(\mathbf{x}_i) = \frac{1}{|W|} \sum_{j \in W} K(\mathbf{x}_i, \mathbf{x}_j) \tag{12}$$

where W represents a set of s_i's neighboring shots, $|W|$ cardinality. Note that $K(\mathbf{x}_i, \mathbf{x}_j)$ is the Gaussian kernel function as defined in equation (10). Equation (12) computes the dot product of $\phi(\mathbf{x}_i)$ and the mean value of its neighbors in the Gaussian kernel space. That is, the compatibility of $\mathbf{x}_i$ with its neighbors is defined as the similarity to its

neighbors in the Gaussian kernel space. It is worthwhile to note that the Gaussian kernel function is very similar to the bell-shaped fuzzy membership function and therefore, our compatibility measure can be considered as a fuzzy compatibility measure [12]. We compute the compatibility for each frame in a given sequence. We select the frames of locally maximal compatibility measures, each of which best represents its neighbors in a compatibility sense in (12). Note that the visual feature in our compatibility definition could be obtained from color, texture, shape of the salient object in the frame, or the combination of the above. In this paper, we select the color histogram of a frame as our visual feature, although other visual features can be readily integrated into the compatibility measure. The color histogram used is a 6×6×6 RGB color histogram. Figures 1 shows the compatibility measures for a hand movement video clip of 380 frames obtained from the Web. Black dots represent local maxima in the Figures. We fixed the scale parameter σ^2 in equation (10) into the standard deviation of the feature vectors within a given video.

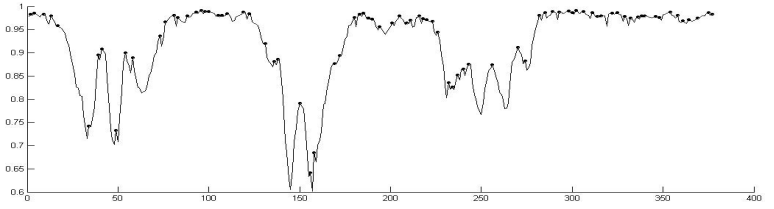


Fig. 1. Compatibility measurements for a hand movement video clip of 380 frames

3.2 Scalable Novelty Detection

One-class SVM described in section 2 has been successfully used in the novelty detection problems [6]. One can control the fraction of outliers and support vectors by varying the value of C. Let $C = 1/(nv)$ where n is the number of training samples and $v \in (0, 1]$. The following statements hold (1) v is an upper bound on the fraction of outliers and (2) v is a lower bound on the fraction of support vectors. These statements directly follow from $0 \le \beta_j \le C$ and $\sum \beta_j = 1$. If we set C to 1, then one-class SVM does not allow the outliers and finds the smallest sphere that encloses all the training data.

The proposed scalable keyframe detection algorithm sets C to 1 and finds the support vectors on the surface of the smallest sphere that encloses all the training points. The obtained support vectors form a set of keyframes. After finding the support vectors on the outer surface and removing them from the training points, the algorithm seeks a new set of support vectors from the rest of training samples. That is, we peel off the surface and obtain yet another enclosing sphere, smaller than the previous. The support vectors from the new enclosing sphere can be added up to the previously

obtained support vectors. In this manner, we can obtain a series of keyframe sets, each of which represents visual content in its own level of detail. Figure 3 shows the result on a synthesized data set. The line in Figure 3 represents the contour of the surface of the sphere in each level.

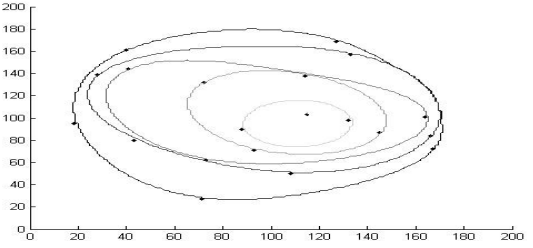


Fig. 2. Scalable Novelty Detection: Contours of the Surfaces of the Smallest Enclosing Spheres

4 Experimental Results

We have experimented with several video clips from the Web. However, we present one of them for the sake of space. In this experiment, we used an MPEG-1 video clip of 380 frames. The $6 \times 6 \times 6$ color histogram in RGB color space was used as a visual feature. The scale factor σ^2 in the Gaussian Kernel function was adaptively adjusted to the standard deviation of a given video clip. Filtering the original sequence with the compatibility measure made 80 frames left. Then, we applied one-class SVM to the remaining 80 frames. Figure 3 shows the result. In order to see the effectiveness of the result, we uniformly sampled every 20 frame from the original sequence. Figure 3(a) shows the uniformly sampled frame sequence. In Figure 3(a), the hand movements are not well captured because the hand movements occur in a burst way within the sequence. Figure 3(b), (c), and (d) show the results from our method. We can see the hand movements well captured even in the first level.

5 Conclusions

In this paper, we presented a scalable keyframe extraction method. Our method is based on the observation that the keyframe extraction problem can be interpreted as the novelty detection problem. For temporal outlier detection, we defined the compatibility measures. For novelty detection in visual features, we used one-class support vector machine. Moreover, we showed how to present a keyframe-based video summary in a scalable fashion. Several experiments on real video clips show the effectiveness of the proposed algorithm.

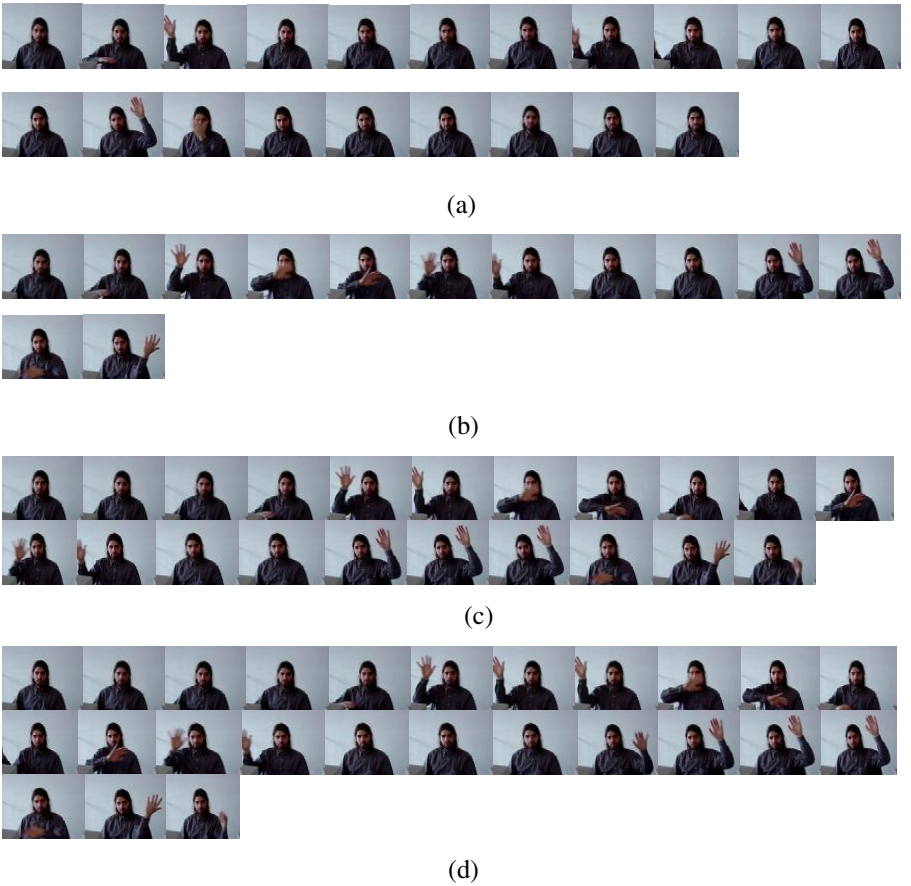


Fig. 3. (a) Result from sampling every 20 frame, (b) Result from our method at level 1, (c) Result from our method at level 2, and (d) Result from our method at level 3

Acknowledgement. This research was supported by IRC (Internet Information Retrieval Research Center) in Hankuk Aviation University. IRC is a Kyounnggi-Province Regional Research Center designatd by Korea Science and Engineering Foundation and Ministry of Science & Technology.

References

1. M Yeung and Boon-Lock Yeo, and Bede Liu “Extracting Story Units from Long Programs for Video Browsing and Navigation”, Proceedings of IEEE International Conference on Multimedia Computing and Systems1996, pp. 296–305.

2. Shingo Uchihashi and, et al, “Video Magna: Generating Semantically Meaningful Video Summaries”, Proceedings of ACM International Conference on Multimedia 1999, pp. 383–391.
3. A. Hanjalic and et al, “Automated High-Level Movie Segmentation for Advanced video-Retrival Systems”, IEEE Transactions On Circuits and Systems for Video Technology, Vol. 9, No. 4, June 1999, pp. 580–588.
4. Michael A. Smith and Takeo Kanade, “Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques”, Proceeding of IEEE, pp.775–781. 1997
5. Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik, “Support Vector Clustering”, Journal of Machine Learning Research 2 (2001) 125–137
6. Klaus Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf, “An Introduction to Kernel-Based Learning Algorithms”, IEEE Transactions on Neural Networks, Vol. 12, No.2, March, 2001, pp.181–202.
7. Nello Cristianini and John Shawe-Taylor, “An Introduction to Support Vector Machines and other kernel-based learning methods”, Cambridge University Press 2000.
8. Nevenka Dimitrova, Hong-Jiang Zhang, Behzad Shahraray, Ibrahim Sezan, Thomas Hunag, and Avideh Zakhor, “Applications of Video-Content Analysis and Retrieval”, IEEE Multimedia, July-September 2002, pp. 42–55.
9. A. D. Doulamis, N. Doulamis, and S. Kollias, “Non-sequential Video Content Representation Using Temporal Variation of Feature Vectors”, IEEE Transactions on Consumer Electronics, Vol. 46, No.3, August 2000, pp.758–768.
10. W. Wolf, “Key frame selection by motion analysis”, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing”, 1996, pp.1228–1231.
11. P.O. Grelsle and T. S. Hunag, “Gisting of video documents: A key frames selection algorithm using relative activity measure,” The 2nd Int. Conf. on Visual Information Systems, 1997.
12. YoungSik Choi, Sun Jeong Kim, and Sangyoun Lee, "Hierarchical Shot Clustering for Video Summarization," Computational Science – ICCS 2002, LNCS 2331.
13. Y. Taniguchi, “An intuitive and effiecient access interface to real-time incoming video based on automatic indexing”, Proc. of ACM Multimedia, Nov. 1995, pp.25–33.
14. S. X. Ju, et. al., “Summarization of video-taped presentations: automatic analysis of motion and gestures”, IEEE Transactions on CSVT, 1998.
15. Y. Taniguchi, et. al., “Panorama Excerpt: extracting and packing panoramas for video browsing”, Proc. of the 5th ACM International Multimedia Conference, pp. 427–436, Nov. 1997.

A Design and Performance Evaluation of 2-Level Call Admission Control Scheme in Mobile Multimedia Networks*

Myung Il Kim and Sung Jo Kim

Dept. of Computer Science & Engineering, Chung-Ang University,
221 Huksuk-Dong, Dongjak-Ku, Seoul 156-756, South Korea
{nicemi, sjkim}@konan.cse.cau.ac.kr

Abstract. Multimedia applications are much more sensitive to the quality of service(QoS) than text-based ones due to their data continuity. In order to provide a fast moving mobile host(MH) supporting multimedia applications with a consistent QoS, an efficient call admission mechanism is in need. This paper proposes the 2-level call admission control(*2LCAC*) scheme based on a call admission scheme using the priority to guarantee the consistent QoS for mobile multimedia applications. The *2LCAC* consists of the basic call admission and advanced call admission; the former determines call admission based on bandwidth available in each cell and the latter determines call admission by utilizing delay tolerance time(*DTT*), priority queue(*PQueue*), and upgrade queue(*UpQueue*) algorithms. In order to evaluate the performance of our scheme, we measure the metrics such as the blocking probability of new calls, dropping probability of handoff calls, and bandwidth utilization. The result shows that the performance of our scheme is superior to that of existing schemes such as complete sharing policy(*CSP*), guard channel policy(*GCP*) and adaptive guard channel policy(*AGCP*).

1 Introduction

The provision of QoS in mobile multimedia networks becomes more complex due to user mobility. The problem becomes even more challenging as recent wireless networks are constructed based on small-size cells(microcells or picocells). Small-size cells increase the number of handoffs, which results in complex and rapid handling in call admission control(*CAC*), thus increasing the handoff dropping rate.

It is important to reduce the dropping rate caused by lack of available bandwidth in the handoff target cell. If the cell's available bandwidth is not sufficient enough to satisfy a handoff call, this call has to compete with other calls to acquire the bandwidth. To solve this problem, several schemes have been proposed in literature for efficient channel assignment.

* This work was supported by grant No. R01-2001-000-00333-0 from the Korea Science & Engineering Foundation.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 500–509, 2003.
© Springer-Verlag Berlin Heidelberg 2003

One well-known scheme is priority-based *CAC* and channel assignment. From the user's point of view, dropping an ongoing call is more undesirable than blocking a new call[1],[2],[3]. As a result, many researches have been devoted to *CAC* and channel assignment using prioritized handoff requests. Another well-known scheme is bandwidth reservation based *CAC* which reserves bandwidth for handoff calls in all or some cells that MH might pass through, resulting in potentially lower bandwidth utilization and higher new call blocking probability.

In this paper, we propose the 2-level call admission control(*2LCAC*) scheme using the priority to guarantee the consistent QoS for mobile multimedia applications. A cell is assumed to be small and overlapped partially with its neighboring cells. MH calls are classified into new calls, hand-off calls, and QoS upgrading calls. The *2LCAC* consists of two stages: basic call admission control(*BCAC*) and advanced call admission control(*ACAC*). The former determines call admission based on available bandwidth in each cell and the latter determines call admission by utilizaing delay tolerance time(*DTT*), priority queue(*PQueue*), and upgrade queue(*UPQueue*) algorithms according to the type of call blocked at *BCAC*.

We compare the performance of *2LCAC* scheme with that of the other *CAC* schemes. Simulation results show that our scheme outperforms the others by satisfying the constraint on call blocking probability of new calls(*CBP*), call dropping probability of handoff calls(*CDP*) and bandwidth utilization.

This paper is organized as follows. We describe our *CAC* scheme in Section 2. Section 3 shows simulation results and compares the performance of our scheme with that of the other schemes under various scenarios. Finally, Section 4 contains the concluding remarks.

2 *2LCAC* Scheme Based on Priority Queue

Generally, it is assumed that a cell is hexagonal in mobile networks. However, in real world, a cell is close to a circle, and there are overlapped areas with neighboring cells as shown in Fig. 1. MHs residing in overlapped areas have higher possibility to move to another cell than MHs not residing in those areas.

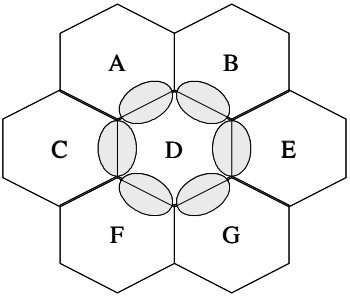


Fig. 1. Overlapped Cell Structure

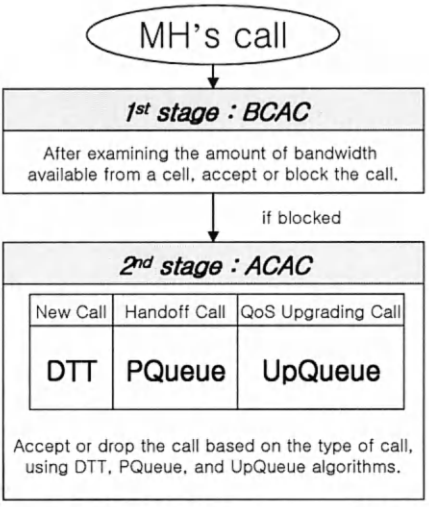


Fig. 2. *2LCAC* Procedure

As shown in Fig. 2, *2LCAC* consists of *BCAC* and *ACAC*. At the *BCAC* stage, each call(new, handoff and QoS upgrading) requests required bandwidth from its base station(BS). That is, *CAC* is performed at the *BCAC* stage when a call is requested for the first time from a specific cell. If available bandwidth of the BS satisfies each call's request, it assigns the bandwidth to the MH; otherwise, it blocks the call and passes its control to the *ACAC* stage.

Table 1. The Policies related to *BCAC*

Call Type	Triggered Time	Conditions		Actions
New	When MH_i requests	$B^{avail}{}_j \geq B^{new}{}_i$		Accept
		$B^{avail}{}_j < B^{new}{}_i$		Block and perform *ACAC* stage
Handoff	When MH_i enters overlapped area	$B^{avail}{}_j \geq B^{handoff}{}_i$		Accept
		$B^{avail}{}_j < B^{handoff}{}_i$		Block and perform *ACAC* stage
QoS Upgrading	When $B^{avail}{}_i$ is higher than Th_{up}	$B^{avail}{}_j \geq Th_{up}$	$B^{avail}{}_j \geq B^{up}{}_i$	Accept
			$B^{avail}{}_j < B^{up}{}_i$	Block and perform *ACAC* stage
		$B^{avail}{}_j < Th_{up}$		Block and perform *ACAC* stage

2.1 Basic Call Admission Control (*BCAC*)

Table 1 shows the policies related to *BCAC*, which are chosen by the type of calls. $B^{new}{}_i$ and $B^{handoff}{}_i$ represent MH_i's required bandwidth of new call and handoff call, re-

spectively. Whereas $B^{up}{}_i$ represents required bandwidth of QoS upgrading call, $B^{avail}{}_j$ represents the amount of bandwidth available in cell j. $B^{avail}{}_j$ can be calculated by Eq. (1):

$$B^{avail}{}_j = B^{total}{}_j - (\sum_{i=0}^{n} B^{handoff}{}_i + \sum_{k=0}^{m} B^{new}{}_k) \quad (1)$$

where $B^{total}{}_j$ is the total amount of bandwidth provided by cell j and n and m are the number of handoff calls and new calls currently residing in cell j, respectively.

In case of a new call, the *CAC* is triggered when MH requests the call admission. If $B^{avail}{}_j$ is larger than $B^{new}{}_i$, the call is admitted; otherwise, blocked. In case of a handoff call, the *CAC* is triggered when the MH enters an overlapped area. If the $B^{avail}{}_j$ is larger than $B^{handoff}{}_i$, the call is admitted; otherwise, blocked. In order to upgrade the current QoS level of an ongoing call, additional bandwidth can be requested by triggering *CAC*. If $B^{avail}{}_j$ is larger than Th_{up}, the call is accepted, where Th_{up} is the threshold used for determining the acceptance of QoS upgrading call; otherwise, blocked. The control of blocked new, handoff and QoS upgrading call will be treated at the *ACAC* stage.

2.2 Advanced Call Admission Control (*ACAC*)

Priority Queue and QoS Upgrading Queue

In *ACAC*, each cell requires space to store information about MH: priority queue(*PQ*) and QoS upgrading queue(*UpQ*). *PQ* stores information on an MH such as ID and the priority that has been blocked as a result of *BCAC* for its handoff. For example, when MH_1 and MH_2 move towards the overlapping area with cell *D* from cell *C* and *E*, respectively, as shown in Fig. 3, *BCAC* is triggered. If both MHs are blocked, the priorities of MH_1 and MH_2 are calculated by cell *D*'s BS and the information on those blocked MHs are inserted into *PQ*. Furthermore, if an MH_3 residing in cell *D* moves towards cell *B* through the overlapped area between the two cells and is blocked as a result of *BCAC*, the priority of MH_3 is inserted into cell *B*'s *PQ*.

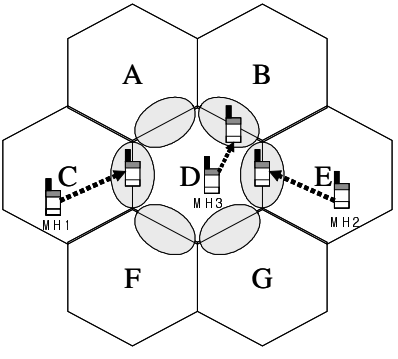


Fig. 3. A Snapshot of Cell Status

The call priority of MH is calculated using Eq. (2):

$$Call\ \ priority = w_1 DP + w_2 TP + w_3 VP \tag{2}$$

where *DP*, *TP* and *VP* are the distance priority, traffic priority and velocity priority, respectively, and w_1, w_2 and w_3 are weight factors. The rationale behind this equation is that an MH has the highest priority as it gets closer to the target cell(*DP*) at a high mobility speed(*VP*) and require the service with high *TP* like multimedia services.

QoS upgrading calls that have been blocked as a result of *BCAC* are inserted into *UpQ*. When the amount of bandwidth available in a cell is greater than Th_{up}, the bandwidth is allocated to the calls in *UpQ* in the FIFO order.

DTT, *PQueue* and *UPQueue* Algorithm

In *ACAC*, different *CAC* algorithms are adopted according to the type of each call; new, handoff and QoS upgrading calls are tested using *DTT*, *PQueue* and *UPQueue* algorithm, respectively. In case of a new call, it requests a call admission to BS periodically until *DTT* expires, where *DTT* is set when the MH is powered on. If available bandwidth of BS is not enough to satisfy the new call's request after *DTT*, it is dropped.

In case of handoff calls, call classification and bandwidth borrowing algorithms are invoked. Call classification algorithm as below is applied to MHs that have just entered the overlapped area. It classifies handoff calls into two classes: *high* and *low*. Class *High* contains handoff calls that have relatively higher priority than the others. Once being classified into this class, the calls will not be degraded into class *low* and bandwidth assignment to them will be also guaranteed. On the other hand, class *low* contains handoff calls that have relatively lower priority than the others. They have high chances of being dropped during *ACAC* and bandwidth assignment to them will not be guaranteed.

Pseudo Code for Call Classification Algorithm

```
Call_Classification(){
  Input  : B^avail_j , B^TPQ_j , n, BSP_j
  Output : Priority of MHs in PQ_j
  if(B^avail_j ≥ B^TQB_j){
    for(i=0; i≤n; i++)
      Priority of MH_i = High;}
  else{
    for(k=0; k<BSP_j; k++)
      Priority of MH_k = High;
    for(m=BSP_j; m≤n; m++)
      if(MH_m enters PQ_j for the first time)
        Priority of MH_m = Low;
      else
        Maintain the current Priority of MH_m}
}
```

In the pseudo code above, B^{TPQ}_j(Total Bandwidth of *PQ*) is the sum of bandwidth requested by all the MHs that reside in cell *j*'s *PQ*. Bandwidth satisfaction point(BSP_j) points out the right most MH in *PQ* of which bandwidth request can not be satisfied from cell *j*'s available bandwidth. To be specific, if B^{avail}_j is equal to or higher than B^{TPQ}_j, the priorities of all the MHs residing in *PQ* become *high*. However, if B^{avail}_j is lower than B^{TPQ}_j, only the priorities of MHs that are at the left-hand side of BSP_j become *high* and the others become *low*. The MHs that are at the right-hand side of BSP_j and have a *high* priority maintain its current priority. MHs residing in *PQ* are sorted based on their priorities.

When the priority of MH_i becomes *high* during its stay in the overlapped area, the required bandwidth is assigned to it if the amount of bandwidth available in the target cell is sufficient enough, or bandwidth borrowing algorithm[1],[3],[4] is triggered to assign the amount of bandwidth requested by MH_i. On the other hand, the algorithm is not triggered immediately if the priority of MH_i belongs to class *low*, and the triggering of the algorithm is delayed until its priority is upgraded to class *high*. A handoff call of MH_i is dropped, if the priority of MH_i cannot be upgraded to class *high* until it leaves the overlapped area.

In case of QoS upgrading call, an MH is stored into the QoS upgrading queue in order to upgrade the QoS level. This happens either when the MH wants to upgrade its service quality or to reclaim as much bandwidth as it has lent to the other MH. Only when the amount of bandwidth available in BS is above the threshold, the required bandwidth is assigned. Just because there is sufficient bandwidth in a cell, the QoS will not be upgraded immediately by *UPQueue* algorithm. To be specific, if the available bandwidth is below the threshold, MH_i is stored in the QoS upgrading queue. Next, only when the available bandwidth is above the specific threshold, the QoS of MH_i is upgraded in the FIFO order. Note that the available bandwidth below the threshold is assigned to new calls and handoff calls, because the QoS upgrade calls have the lower priority than both new calls and handoff calls.

3 Performance Evaluation

In this section, we compare the performance of proposed scheme with that of other schemes such as *CSP*, *GCP*, and *AGCP*. In *CSP* [5], new and handoff calls are not distinguished from each other and are admitted as long as available bandwidth of a cell can provide the required bandwidth to the MH. In other words, the same priority is given to both the new and handoff calls. This scheme has the low *CBP*, but high *CDP*.

GCP [5],[6],[7] reserves a fixed amount of bandwidth for multimedia handoff calls. This scheme reserves as much bandwidth as required by an MH requesting multimedia services and having high probability of entering a certain cell. The bandwidth is assigned later when the MH actually enters the cell. This scheme has an advantage of having the low *CDP* for multimedia handoff calls by assigning reserved bandwidth to them. However, since the available bandwidth is reserved for multimedia calls, new

and non-multimedia handoff calls have high *CBP* and *CDP*. Moreover, the reserved bandwidth may turn out not to be used, thus resulting in low bandwidth utilization.

AGCP [2],[8],[9] is a policy that adaptively reserves a portion of bandwidth for multimedia handoff calls depending on network condition. If the *CDP* of a cell is above the specific threshold (e.g., 0.05), the amount of reserved bandwidth is increased. On the other hand, if the *CBP* is below a certain threshold, a certain amount of reserved bandwidth is decreased only when the utilization of reserved bandwidth is below specific threshold (e.g. 0.5). This scheme has the lower *CDP* than *GCP*. When the number of new calls increases dramatically, however, the *CDP* is increased as well.

Table 2. Traffic Classes Used in the Simulation

Traffic Class	Bandwidth Requirement	Call Duration (minutes)	Typical Service
Class I	30Kbps	1~10	Voice Service
	256Kbps	1~30	Video-Phone
	1~6Mbps	5~300	Video on Demand
Class II	32Kbps	0.2~2	E-mail, Paging
	64~512Kbps	0.5~600	Remote Login & Data on Demand
	1~10Mbps	0.5~20	FTP

In this simulation, we suppose that the mobile multimedia network is comprised of 100 hexagonal cells, each of which diameter is 1 km. We also suppose that each cell provides the bandwidth of 30Mbps and communicates with its six neighboring cells. The client moves at a speed from 1 to 100 Km/h. New call reqeusts are generated according to a Poisson distribution with the arrival rate λ(calls/second) in each cell. A new call is assumed to be requested at any place in the cell with an equal probability. Table 2 shows six typical traffic classes used in the simulation[8].

Performance metrics considered in the simulation are *CBP*, *CDP* and bandwidth utilization. The metrics are measured with respect to *CSP*, *GCP*, and *AGCP* as the arrival rate of new calls vary.

We first attempt to find out the most reasonable rate of class I calls by performing our simulation under the situation as follows; the arrival rate of new calls is fixed at 0.1(MH/sec) and the proportion of class I calls is incremented by 10% from 10% to 90%. We find that the *CBP*, *CDP*, and bandwidth utilization showed the best performance when the proportion of class I call is 50%. We then perform our simulation under the following situation; the proportion of class I calls is fixed at 50% and the arrival rate of new calls is incremented by 0.01(MH/sec) from 0.01 to 0.1.

Fig. 4 shows the *CBP* for each of four schemes as the arrival rate of new calls increases from 0.01 to 0.1. According to the figure, *2LCAC* has the lowest *CBP* followed by *CSP*. It also shows that *AGCP* has the highest *CBP*. This is because *2LCAC* allows an MH to request the acceptance of its blocked new call repeatedly at the

BCAC stage until *DTT* expires without human-user's intervention. The *CBP* of *AGCP* and *GCP* are lower than that of *CSP*. The reason for that is that *AGCP* and *GCP* assign higher priority to handoff calls than new calls and reserve some bandwidth for handoff calls. As a result, as the amount of bandwidth assigned to new calls decreases, the *CBP*s of *AGCP* and *GCP* are higher than those of *2LCAC* and *CSP*. On the other hand, the *CSP* treats new and handoff calls equally and does not reserve bandwidth for handoff calls.

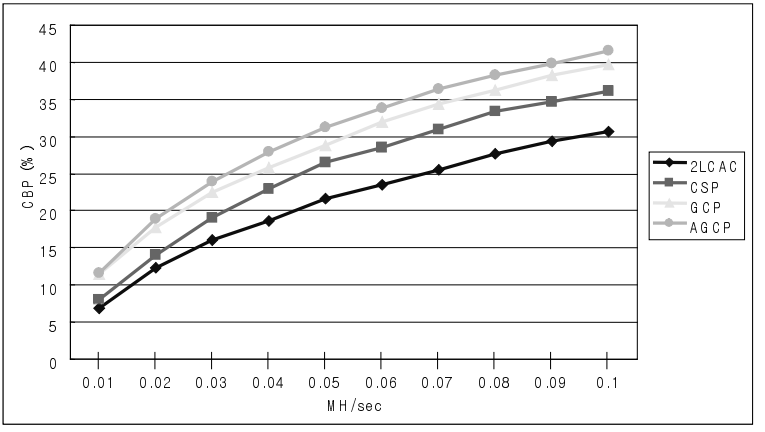


Fig. 4. Blocking Probability of New Calls

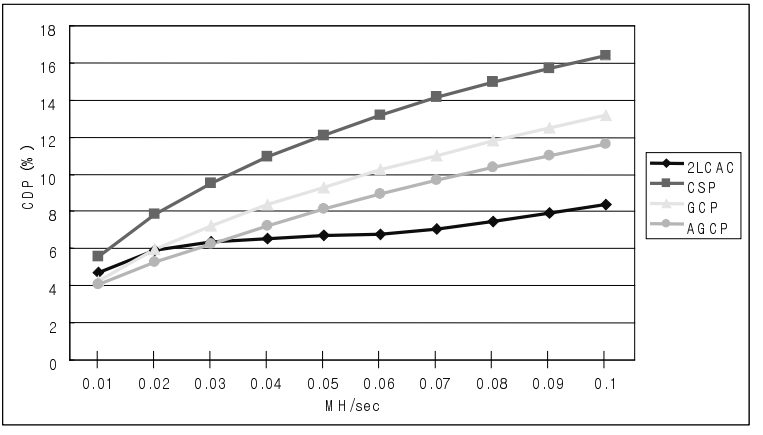


Fig. 5. Dropping Probability of Handoff Calls

Fig. 5 shows the *CDP* of each scheme as the arrival rate of new calls increases from 0.01 to 0.1. *2LCAC* has the lowest *CDP*, followed by *AGCP*, *GCP*, and *CSP*. This is because *2LCAC* gives a priority to both class I and class II handoff calls efficiently and fairly. Fair bandwidth assignment is guaranteed by *2LCAC* based on the type of calls, the speed of MH, and MH's distance from the target cell. Even when there is not

enough bandwidth available in a cell, handoff call admission is guaranteed for handoff calls with high priority by reclaiming bandwidth at the *ACAC* stage. If there are many new call requests, the *CSP* has the highest *CDP* because it does not assign higher priority to handoff calls than new calls. In case of *AGCP* and *GCP*, both have the high *CDP* because the reserved bandwidth might not be used at all and be eventually wasted.

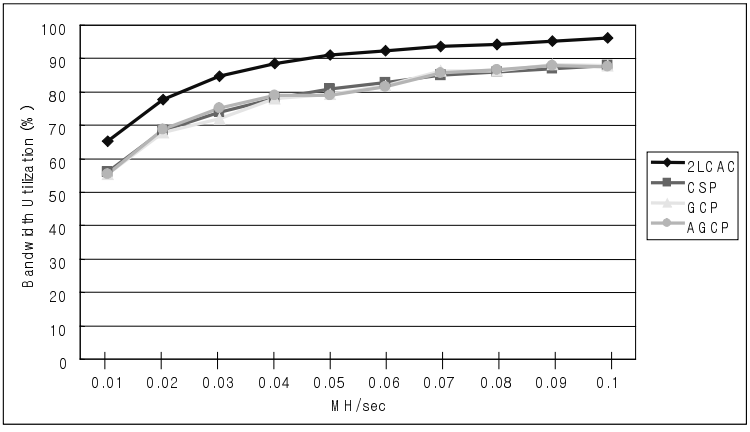


Fig. 6. Bandwidth Utilization

Fig. 6 shows the bandwidth utilization of each scheme. *2LCAC* has the highest utilization followed by *CSP*, *GCP*, and *AGCP*. This is because *2LCAC* uses available bandwidth of a cell more efficiently than other schemes. When there is available bandwidth in a cell, it is assigned to QoS upgrade calls. Since QoS upgrade calls of an MH reclaim bandwidth which has been previously lent to handoff calls with high priority, *2LCAC* manage to utilize the bandwidth efficiently. On the other hand, *AGCP* and *GCP* have low bandwidth utilization, because reserved bandwidth can be unused and be wasted.

4 Conclusion

For efficient and fair *CAC*, we proposed *2LCAC* scheme using priority queue. We classified MH calls into three types: new, handoff, and QoS upgrade call. The priority of each type is assigned in the order of handoff, new, and QoS upgrade call, and each type uses the different call admission policy. *2LCAC* consists of two stages: *BCAC* and *ACAC*. At the former stage, calls from an MH are admitted without distinguishing new and handoff calls just as the *CSP*, when the amount of bandwidth available in a cell satisfies the required bandwidth of the MH. In case of QoS upgrading calls, they are accepted if the available bandwidth is above the specific threshold; otherwise, the calls are blocked and *ACAC* is triggered. The latter stage tests whether the calls

blocked at the former stage can be accepted or not using *DTT*, *PQueue*, and *UPQueue* algorithms depending on the type of calls.

In order to evaluate the performance of our scheme, we measured the metrics such as *CBP*, *CDP*, and bandwidth utilization. We then compared the performance of *2LCAC* with that of three different schemes: *CSP*, *GCP*, and *AGCP*. Because *CSP* does not distinguish new calls from handoff calls, it has the low *CBP* and high *CDP*. Since *GCP* reserves a fixed amount of bandwidth, it has the low *CDP*, but the highest *CBP*. This scheme has the lowest bandwidth utilization because it cannot adaptively readjust the amount of bandwidth for handoff calls. The performance of *AGCP* is better than that of other schemes, but still worse than that of *2LCAC*. To be specific, *2LCAC* lowers its *CBP* and *CDP* by 6~11% and 5~8%, respectively, compared to *CSP*, *GCP* and *AGCP*. The bandwidth utilization of *2LCAC* is also better than that of other schemes by 7~8%. All in all, *2LCAC* utilizes the bandwidth of BS more efficiently than other conventional schemes.

References

1. W. Zhuang, B. Bensaou, and K. Chua, "Adaptive Quality of Service Handoff Priority Scheme for Mobile Multimedia Networks," *IEEE Transactions on Vehicular Technology*, Vol.49, No.2, pp.494–505, 2000.
2. C. H. Choi, M. I. Kim and S. J. Kim, "Call Admission Control Using Moving Pattern of Mobile Users for Mobile Multimedia Networks," *IEEE Conference on Local Computer Networks*, pp.431–440, 2002.
3. M. Ei-Kaid, S. Olariu and H. Abdel-Wahab, "Rate-Based Borrowing Scheme for QoS Provisioning in Multimedia Wireless Networks," *IEEE Transaction on Parallel and Distributed Systems*, Vol. 13, No. 2, pp.156–166, 2002.
4. A. Xhafa and O. K. Tonguz, "Dynamic Priority Queuing, Channel Borrowing, and First In First Out Handoff Schemes: A Performance Comparison," *IEEE Vehicular Technology Conference Spring*, Vol.2, pp.951–955, 2001.
5. B. Gavish and S. Sridhar, "Threshold Priority Policy for Channel Assignment in Cellular Networks," *IEEE Transactions on Computers*, Vol.46, No.3, 1997.
6. J. Lee, et al., "An Adaptive Resource Allocation Mechanism including Fast and Reliable Handoff in IP-based 3G Wireless Networks," *IEEE Personal Communications*, Vol.7, No.6, pp.42–47, 2000.
7. E. El-Alfy, Y. Yao, and H. Heffes, "A Learning Approach for Call Admission Control with Prioritized Handoff in Mobile Multimedia Networks," *IEEE Vehicular Technology Conference*, Vol.2, pp.972–976, 2001.
8. C. Oliveria, J. Kim and T. Suda, "An Adaptive Bandwidth Reservation Scheme for High Speed Multimedia Wireless Networks," *IEEE JSAC*, Vol.16, No.6, pp.858–874, 1998.
9. S. Choi and K. Shin, "Predictive and Adaptive Bandwidth Reservation for Hand-offs in QoS-Sensitive Cellular Networks," *Proceeding ACM SIGCOMM'98*, pp.155–166, 1998.

Performance Evaluation of a Relocation Method for Mobile Computing Maintaining Data Consistency

Byung-Kwan Lee [1] and Am-Suk Oh [2]

[1] Dept. of Computer Engineering, Kwandong Univ.
San-7, YangYangyup, YangYanggun, KwangwonDo, Korea,
bklee@kwandong.ac.kr

[2] Dept. of Multimedia Engineering, TongMyong University
of Information Technology. Yongdang-dong, Nam-gu, Busan, Korea
asoh@tmic.tit.ac.kr

Abstract. Recently, a method to use replicated data on a server in order to get new data without missing any information has been studied on wireless telecommunication networks and satellite services. Static Replica Allocation (SRA) that traditional scheme has been used for the replication method on the server. This replicates the data on the replica server after a moving host has been transferred to another cell. In case the number of users is smaller than that of cells, the network of the SRA is an efficient scheme in the standpoint of access cost, and if there are few moving users, no trouble will happen. But if there are no moving users in a cell, the data will not be shared. This paper proposes new method of relocation, that is, USRA (User Select Replica Allocation) based on data consistency scheme (called USRAC) after replicating the data to the cells for the users, and also analyzes access cost according to the moving rate of mobile users, according to the access rate of mobile hosts, and according to the number of cells of mobile users and mobile hosts.

1 Introduction

Mobile computers and wireless communication networks have made it possible to open up a new field in mobile computing. Mobile computing provides mobile users with the ability to access information regardless of the location of the user or the information. In traditional distributed database systems, various strategies have been suggested in order to process transaction and maintain consistency. But most solutions are restricted to mobile computing because of mobility, network failure and lower bandwidth. The replication of data allows mobile computers to quickly access data. But consistency among duplicated data should be maintained. The distributed database in the mobile computer environment for replicating data has used SRA(Static Replica Allocation). SRA is a scheme that when a mobile host moves to a cell, replicates data to the replica server within the cell. In case the network is stable and a number of mobile users smaller than cells, this scheme does not cause any problem with using data. However, after the mobile host moved to a cell, if no mobile user existed in the

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 510–517, 2003.
© Springer-Verlag Berlin Heidelberg 2003

replicated cell, replicated data become useless. And this scheme is not suitable for the mobile computing environment because of frequent disconnection, lower bandwidth, etc.

In this paper, we present our replica allocation called USRAC (User Selection Replica Allocation based on Consistency scheme) which makes up for defects of SRA scheme and is appropriate to the mobile computer environment. This scheme replicates data and then allocates it to the cell in which mobile users are numerous. We have conducted a detailed performance evaluation of SRA, USRA and USRAC scheme proposed in terms of access cost and communication cost based on access ratio of mobile user, access ratio of mobile host and variation of number of cells.

2 SRA (Static Replica Allocation)

Replication is the process that automatically updates data involved in a replica server maintains the same object and data. Whenever mobile users need to replicate data, they can do it. The replication is used to increase the speed of the query and information retrieval. SRA(Static Replica Allocation) is a strategy that has been currently researched. SRA is a method that replicates data to the replica server of moved cell if a mobile user requires data. The advantage of this method is that data replication is simple and easily relocates data. However, because mobile host replicates data to moved cell, in case mobile user does not exist in the moved cell, the replicated data can be deleted for consistency maintenance. And if mobile host leaves replicated cell, the access of data through route becomes difficult because of the frequent disconnection of mobile computing environment

The advantage of SRA strategy is to easily replicate transacted data and replication is simple. Therefore, this scheme is suitable when the network is stable and mobile users in the cell are smaller than mobile support stations. However, the first disadvantage of this scheme is that searching the route of the cell in which data transacted is replicated is difficult. Second, although location replicated is known, it is not easy to access a data through route since a frequent disconnections is occurred. Third, in case mobile user does not exist in the replicated cell, data transacted cannot be used. Fourth, if a mobile host moves, the data of replica server is deleted since the data of replica server cannot maintain consistency.

3 USRAC (User Selection Replica Allocation Based on Consistency Scheme)

3.1 USRA Process

Because SRA strategy is difficult to trace the location of a replicated data and use a data and maintain consistency, we propose a new replication scheme called USRA (user selection replica allocation). USRA strategy is a scheme that a replica server is

assigned to a cell in which mobile users are numerous and then relocates a data to the replica server. However, when access cost is computed to this scheme, cost to relocate replication is also added since this strategy additionally should relocate to the cell in which mobile users are more numerous.

USRA is strategy that determines the location relocated of replication at location server and assigns replica server. Replica server takes data information and location information and then provides the data which mobile user requires. This scheme resolves disadvantage that data is not used or the location route of a data has to be searched since selects cell which the user of a data is more numerous and replicates the data. If the relocation of replication is not determined, the loading of data replicated may increase.

3.2 Data Consistency Scheme

The data consistency among duplicated data in the mobile computing environment should be maintained. Thus, we present a data consistency scheme for USRA.

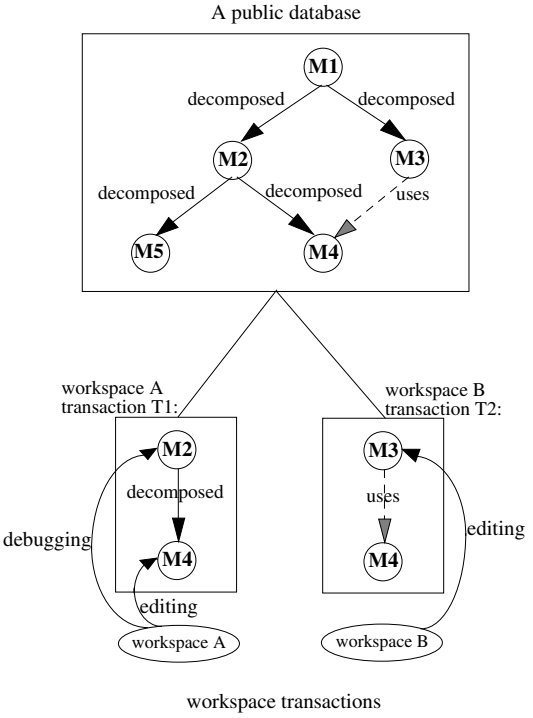


Fig. 1. A Transaction Model

3.2.1 Transaction Model

Fig. 1 shows a transaction model for data consistency. A transaction T1 develops two programs, M2 and M4, at workspace A. The other transaction T2 edits M3 and M4 at workspace B. Here we assume that M2 is **decomposed** into M4 and M5, where M4 is also used by M3. If M3 uses M4, we also say that M3 is a client of M4, since M3 requires the interfaces that M4 provides. The solid arrow denotes the relationship **'decomposed'**. The dashed arrow denotes the relationship **'uses'.**

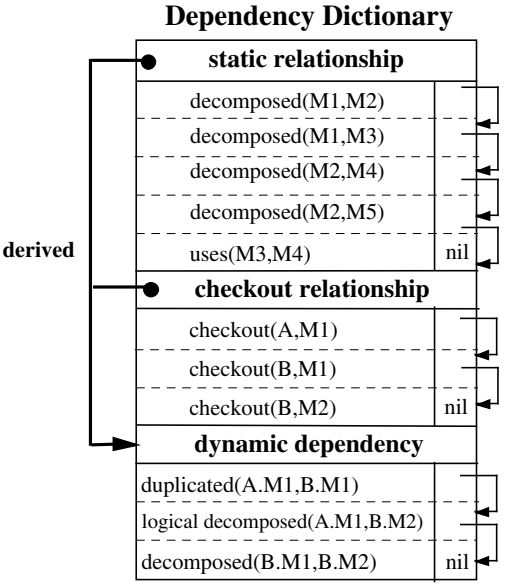


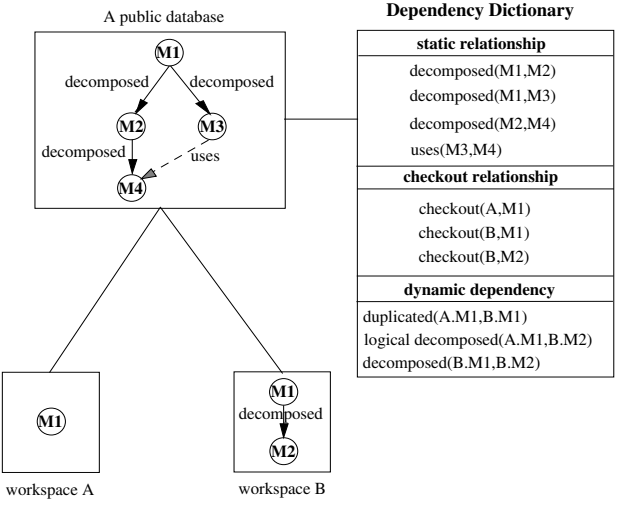
Fig. 2. Consistency Table

3.2.2 Consistency Table

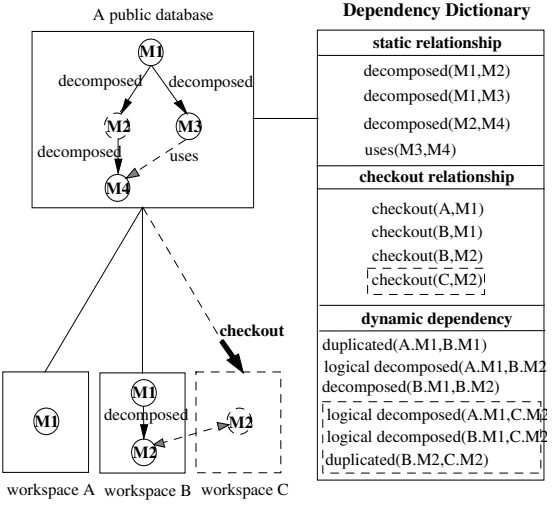
To represent the consistency scheme, we use the structure of linked lists, because the dynamic dependency grows and shrinks according to check-in/check-out operations performed. The consistency table shown in Fig. 2, consists of three lists: a list of static relationships, a list of checkout relationships, and a list of dynamic relationships. In Fig. 2, the static relationships represent those relationships among all objects in the public database. For instance, the relationship 'decomposed(M1,M2)' means that M1 is decomposed into M2. The list of static relationships is not influenced by any checkout operation. However, a new static relationship may be added to the list of static relationships by means of checkin operations.

3.2.3 Consistency Scheme

Fig. 3 is a scheme for data consistency. Fig. 3(a) represents the state before M2 is checked out to workspace C. Fig. 3(b) represents the state where M2 has been newly checked out to workspace C. The dashed rectangle within the consistency table in Fig. 3 (b) shows the new relationships generated after M2 is checked out into workspace C.



(a) before M2 is checked out



(b) after M2 is checked out

Fig. 3. Scheme for data consistency

4 Experiments and Performance Evaluation

We evaluated performance among SRA, USRA and USRAC strategy in the mobile computing environment by computing access cost and communication cost.

4.1 The Access Cost for Move Ratio of Mobile User

Fig. 4 illustrates the access cost of SRA, USRA and USRAC for the variation of access ratio(α) of mobile host. The move ratio of mobile user is generated from 0 to 1λ. We found out that if α is over 0.8, access cost maintains as fig. 4 regardless of variation of α value.

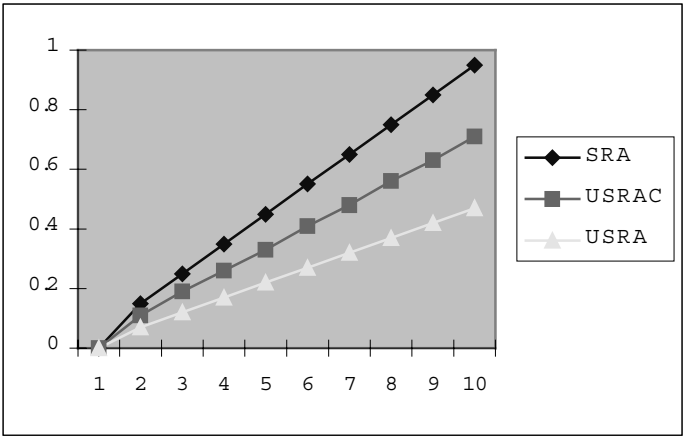


Fig. 4. The access cost of mobile user for α=0.8

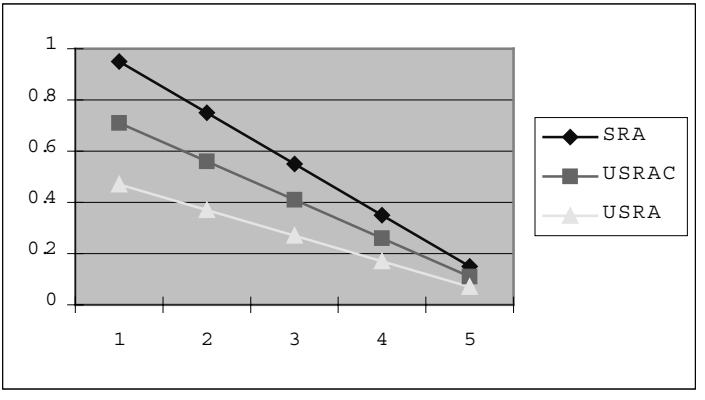


Fig. 5. The access cost of mobile user for λ=0.8

4.2 The Access Cost for Access Ratio of Mobile Host

Fig. 5 illustrates the access cost of mobile host as the access ratio(λ) of mobile user and the access ratio of mobile host varies from 0 to 1. When the access ratio(λ) of mobile user is over 0.8, the result is as fig. 5.

4.3 The Access Cost for The Number of Cells

Fig. 6 illustrates the access cost of SRA, USRA and USRAC for variation of the number of cells. We found out that the access cost of USRA and USRAC is more sensitive than SRA to the variation of the number of cells. Because the replicated cell also increases when the number of cells increases.

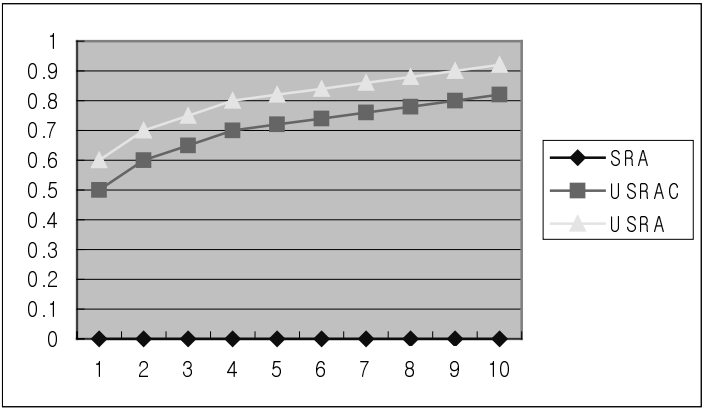


Fig. 6. The access cost for the number of cell to α=0.8, λ=0.8

5 Conclusions

In this paper we have considered USRAC scheme that relocates a replication for mobile computers. SRA(Static Replica Allocation) is a method that replicates a data to the replica server of moved cell if a mobile user requires a data. SRA takes the advantage that data replication is simple and can easily relocate data. However, if mobile user or mobile host is not existed in the moved cell then the replication data becomes useless and it makes move ratio is high. Because USRA selects and replicates the cell that the number of mobile user is most numerous, the move ratio of mobile user can be reduced. However, USRA scheme was not considered serious problem of data consistency in the mobile computing environments. In this paper we have investigated three schemes – SRA, USRA and USRC scheme and have evaluated performance in terms of cost based on schemes. Conclusively, though the USRAC scheme that we have proposed is medium cost, it makes maintain data consistency in the mobile computing.

References

1. A. Chandrasekaran and R. W. Broderson, "A portable multimedia terminal for personal communications," IEEE Communications Magazine, pp. 64–75, December 1992.
2. A. V. Bakre and B. R. Badrinath, "Implementation and performance evaluation of indirect TCP," IEEE Trans. on Computer, vol. 46, no. 3, pp. 260–278, March 1997.
3. D. Duchamp, S. K. Feiner, and G. Q. Maguire, Jr., "Software technology for wireless mobile computing," IEEE Nerwork Magazine, pp. 12–18, November, 1991.
4. D. J. Goodman, G. P. Pollini, and K. S. Meier-Hellstern, "Network control for wireless communications," IEEE Communications," IEEE Communications Magazine, pp. 116–124, 1992.
5. P. Evaggelia et al., "Revising Transaction Concepts for Mobile Computing", proc. IEEE Workshop on Mobile Systems and Applications, Dec 1994.
6. F. Teraoka, K. Claffy, and M. Tokoro, "Design, implementation, and evaluation of virtual internet protocol," IEEE Distributed Computing Systems DCS'92, pp. 170–177, 1992.
7. G. H. Forman and J. Zahorjan, "The challenges of mobile computing," IEEE Computer, pp. 38–47, April 1994.
8. K. Lim and Y. H. Lee, "Optimal partitioning of heterogeneous traffic sources in mobile communications networks," IEEE Trans. on Computer, vol. 46, no. 3, pp. 312–325, March 1997.
9. K. H. Choi, "Data Relocation Using User Selecting Replica Allocation in Mobile Computing Environment," Phd Thesis, University of Kwandong, 1999.
10. T. Imielinski and B. R. Badrinath, "Mobile wireless computing," Communications of the ACM, vol. 37, no. 10, pp. 18–28, October 1994.
11. T. Imielinski and B. R. Badrinath, "Data Management for mobile computing," the ACM SIGMOD RECORD. vol. 22, no.1, pp. 34–39, 1993.
12. Y. Huang, P. Sistla, and O. Wolfson, "Data replication for mobile computers" Proc. of the 1994 ACM SIGMOD Intl. Conf. on Management of Data, pp. 13–24, 1994.

Reliability Problem on All Pairs Quickest Paths

Young-Cheol Bang[1], Hyunseung Choo[2,*], and Youngsong Mun[3]

[1] Department of Computer Engineering, Korea Polytechnic University, Kyunggi-Do, KOREA
ybang@kpu.ac.kr
[2] School of Information and Communication Engineering, Sungkyunkwan University, Suwon 440-746, KOREA
choo@ece.skku.ac.kr
[3] School of Computing, Soongsil University, Seoul, KOREA
mun@computing.ssu.ac.kr

Abstract. The reliability problem of the quickest path deals with the transmission of a message of size σ from a source to a destination with both the minimum end-to-end delay and the reliability of the path over a network with bandwidth, delay, and probability of fault free on the links. For any value of message size σ, we propose $O(n^2m)$ time algorithm for all-pairs quickest most reliable path problem, where n and m are the number of nodes and the number of arcs or links in the network, respectively.

1 Introduction

We consider point-to-point computer networks represented by a graph $G = (V, A)$ with n nodes and m arcs or links. Any node can be reached from any other node in this network, and two nodes are connected by at most single directed link in each direction. Each directed link $l = (i, j) \in A$ has a bandwidth $B(l) \geq 0$, delay $D(l) \geq 0$, and probability of fault free $0 \leq \pi(l) \leq 1$. A message of σ units can be sent along the link l in $T(l)\,\sigma / B(l) + D(l)$ time with reliability $\pi(l)$ as in [6].

Consider a simple path P from i_0 to i_k given by (i_0, i_1), (i_1, i_2), …, (i_{k-1}, i_k), where $(i_j, i_{j+1}) \in A$, for $j = 0, 1, 2, \ldots, (k - 1)$, and all $i_0, i_1, \ldots, i_k$ are distinct. Subsequently, a simple path is referred to simply as a path. The delay of this path P, denoted by $D[P]$, is given by $\sum_{j=0}^{k-1} D(l_j)$, where $l_j = (i_j, i_{j+1})$. The bandwidth of this path is defined as $B(P) = \min_{j=0}^{k-1} B(l_j)$. The reliability of P, denoted by $R(P)$, is $\prod_{j=0}^{k-1} \pi(i_j, i_{j+1})$. The end-to-end delay of the path P in transmitting a message of size σ is $T(P) = \sigma / B(P) + D[P]$ with reliability $R(P)$.

---

* Corresponding Author

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 518–523, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The path P from s to d is the most reliable (MR) if $R(P)$ is the maximum among all paths from s to d. The path P from s to d is the quickest for message size σ if $T(P)$ is the maximum among all paths from s to d. The path P is the quickest most reliable if it is the quickest path to transmit σ units among all MR paths from s to d .

Determining the end-to-end delay paths is very critical part of achieving QoS (Quality of Service) in wide area network. One of the well-known routing algorithms to support QoS is the quickest path algorithm in networks with bandwidth and delay constraints. The quickest path problem was first introduced by Moore [3] and formulated by Chen and Chin [1]. Due to its importance in the computer network, the problem received strong attention in the area of operations research and was extensively studied by Rosen et al. [5], Rao and Batsell [4], D.T. Lee [2], and Bang et al. [7]. In real-life computer networks, communication links may fail, and hence it is important to determine quickest paths that are reliable as well. Recently, such reliability aspects have been studied by Xue[6], and $O(rm + rn\log n)$ time algorithms were proposed for computing both QMR and the most reliable quickest (MRQ) path from a source to destination when a message size σ is given, where r is the number of distinct link bandwidths [6].

In this paper, we consider all-pairs QMR paths of which it is very difficult to derive an efficient algorithm in terms of time-complexity. Since MRQ is relatively easy, we do not consider the QMR algorithm. The all-pairs version of the classical quickest path problem was solved in [2] with time complexity of $O(mn^2)$. By applying the algorithm of Xue [6] to compute all-pairs QMR algorithm, the time complexity for the algorithm is $O(rnm + rn^2\log n)$ with n nodes, m links, and r distinct bandwidth in the network. In this paper, we present $O(mn^2)$ time algorithm to compute all-pairs QMR paths, which match the best-known complexity for the all-pairs classical quickest path problem.

The rest of paper is organized as follows. In section 2, we present details of algorithm to compute the QMR paths for all pairs of nodes in a given network. In section 3, summary of our research is presented.

2 The Quickest Most Reliable Paths

As introduced, the QMR path P is quickest among all MR paths so that the simplest method is to compute all MR paths from source s to destination t and then select the quickest path among such paths. Since the number of all MR paths is possibly exponential, the most reliable path network (MRPN) was introduced in [6] to derive an efficient polynomial time algorithm to compute the QMR path. A MRPN, originally called shortest path network (SPN) in [6], is a directed and acyclic subnetwork of G that contains all MR paths from s to any other $t \in V$.

A MR path can be computed by invocating the shortest path algorithm with new weight π' such that $\pi'(l) = \log(1/\pi(l))$, where the weight of a path is the sum of the weights of its links. However there can be more than one MR paths from which QMR path may be chosen. To account for such paths, we should modify a classical shortest

path algorithm such as Dijkstra's shortest path algorithm that can compute at most one MR path.

Fig.1 illustrates one-to-all MRPN. Let a be the source in G. Then, the subnetwork illustrated in (b) is the MRPN rooted at a. As shown in (b), any path from a is the MR path. However, this property does not hold for all-pairs MR paths. Figure 2 represents the all-pairs MRPN (AMRPN) after merging all MR paths for all pairs of nodes in G. As shown in Figure 2, a path $P(a, b)$ is the MR path from a to b and a path $P(b, c)$ is the MR path from b to c, but $P(a, b, c)$ is not the MR path. We call the path like $P(a, b, c)$ as the redundant path (RP). To solve the problem caused by RPs, we introduce the labeling technique such that links on any MR path are labeled with source-destination pair. Since single link may be included in multiple MR paths, it may contain multiple labels. Thus, every link on a QMR path has at least one same label.

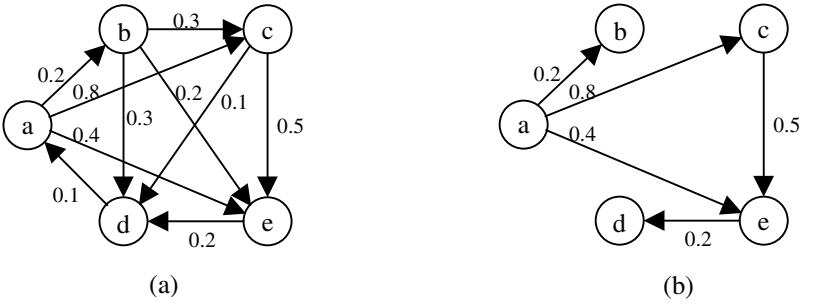


Fig. 1. (a) Network G (b) Subnetwork after merging all the most reliable paths from a to every other node is G; Number in each link represents $\pi'(e)$.

Once AMRPN is constructed by merging all labeled links of all the MR paths, All-pairs Quickest Path algorithm (AQP) [2] is applied to AMRPN. As shown in Fig2, since AMRPN may consist of redundant paths, the classical AQP may compute the quickest paths that are not the MR paths. Thus, AQP should be modified to ensure that only edges on appropriate MR paths are considered in computing QMR paths

Our algorithm consists of following three summarized steps.

1. Compute all the MR paths for each pair of nodes i, $j \in V$, and merge such paths to construct AMRPN.
2. Label each link in AMRPN to distinguish redundant paths
3. Compute all pairs quickest paths of AMRPN with enhancements.

In order to compute all the MR paths, let us define the predecessor data structure as a two dimensional array $pred[i, j]$ for $i, j \in V$, where each element of the array points to a linked list. If $pred[i, j] \rightarrow p \rightarrow q$, then there are two paths from i to j with one of them passing through the link (p, j) and the other through the link (q, j). Thus the data structure $pred$ can store all MR paths for every pair of nodes $i, j \in V$.

Let $\Phi[i, j] = \sum_{e=0}^{k-1} \pi'(l_e)$, where $l_e = (i_e, i_{e+1})$. Let $\tau[i, j]$ be minimum among all

$\Phi[i, j]$. Then we first compute $\tau[i, j]$ for all pairs of node (i, j) with $i, j \in V$. As shown in algorithm AQMR, whenever we find $\Phi[i, j]$ such that $\Phi[i, j] = \tau[i, j]$, new predecessor of node j is added to the list of $pred[i, j]$. Let it call elm. Since at least one of the MR paths from s to j passes through a link (elm, j), the MRPN can be constructed by merging links (elm, j) for all $j \in V$.

The labeling process can be easily done by backtracking each MR path from every $t \in V$ to s. If a link (i, j) is on the MR path from s to t, then $L_{ij}[s, t]$ is marked, where L_{ij} is the n x n matrix with $n = |V|$. By doing so, we may distinguish each link used by different MR paths. Once all links in each MR path are labeled, AMRPN can be constructed by merging all MR paths with labeled links. The following algorithm is to construct AMRPN with labeled links.

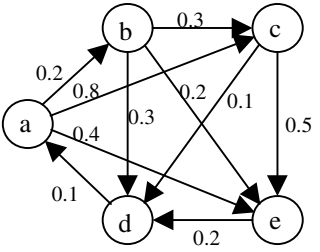


Fig. 2. Subnetwork after merging all the most reliable paths for all pairs of nodes in G; Number in each link represents $\pi'(e)$.

Algorithm AMRPN

Comment : Suppose $\tau[u, v]$ are pre-computed for all pairs of nodes (u, v) with $u, v \in V$

1. $\Phi[u, v] = \infty$ for all pairs of nodes (u, v) with $u, v \in V$
2. **for** each node $s \in V$ **do**
3. $\Phi[s, s] = 0$
4. $pred[s, v]$ = null for all $v \in V$
5. $Q \leftarrow V[G]$
6. **while** $Q \neq \varnothing$ **do**
7. $u \leftarrow$ a node with minimum of $\Phi[s, v]$ from Q
8. **for** each vertex $v \in Adj[u]$ **do**
9. **if** $\Phi[s, v] \geq \Phi[s, u] + \pi'[u, v]$ **then**
10. $\Phi[s, v] = \Phi[s, u] + \pi'[u, v]$
11. **if** $\Phi[s, v] = \tau[s, v]$ **then**
12. $pred[s, v] \rightarrow elm = u$
13. **for** each node $u \in V$ **do**
14. **for** each node $v \in V$ **do**
15. backtrack every path from v to u.
16. **if** a link (i, j) is on a path from s to t **then**
17. mark $L_{ij}[u, v]$

18. **for** each node $u \in V$ **do**
19. **for** each node $v \in V$ **do**
20. **while** $pred[u, v] \neq$ Null **do**
21. add link ($pred[u, v] \rightarrow elm, v$) to AMRPN

Lines 1–12 are modifications of Dijkstra's shortest path algorithm to compute all-pairs MR paths. In each iteration, line 11 checks if $\Phi[s, v] = \tau[s, v]$. If so, a predecessor of v is added to list of $pred[s, v]$. This process can be computed in $O(nm + n^2 \log n)$ by using Fibonacci heap [8]. Once all-pairs MR paths are determined, each MR path is labeled with (s, t) pair in lines 13-17 by backtracking each path from t to s. Since there are at most m links to be traveled for backtracking from each destination to single source, it takes at most $O(nm)$ to label. Thus, the time-complexity of this process for all pairs of nodes is $O(n^2 m)$. Lines 18-21 are the process to compute AMRPN by merging links of all-pairs MR paths. Thus the algorithm AMRPN can be constructed in $O(n^2 m)$.

To determine all-pairs QMR paths, we use $L_{ij}[s, t]$ which ensures that link (i, j) is on a MR path from s to t. If $L_{ij}[s, t]$ is marked then link (i, j) is on the MR path from s to t. Thus, (i, j) can be considered in computing a QMR path from s to t. Otherwise, (i, j) is ignored.

The following algorithm AQMR is to compute all-pairs QMR path. Let $d[u, v]$ and $D(u, v)$ be the delay of the quickest path from u to v and the delay of link (u, v), respectively at any iteration of the algorithm. The links of AMRPN are considered in the non-increasing order in lines 4-11; top operation of line 5 returns the top element and removes it from the heap. In each iteration, the chosen link (i, j) is checked if it is on some MR path from u to v in line 8. If yes, the algorithm is identical to that in [3]: it checks it the path via (i, j) is quicker than the quickest path computed so far, and replaces the latter if so. If not, this link (i, j) is not considered further.

Algorithm *AQMR*(AMRPN)

1. arc_heap ← top-heavy heap of all $l \in E$ according to $B(l)$
2. **for** pair $u, v \in V$ **do**
3. $d[u, v] = \infty$;
4. **while** arc_heap $\neq \varnothing$ **do**
5. (i, j) = top(arc_heap); let $l = (i, j)$;
6. **if** $d[i, j] > D(l)$ **then**
7. **for** each pair $u, v \in V$ **do**
8. **if** $L_l[u, v]$ is marked **then**
9. **if** $d[u, i] + D(l) + d[j, v] < d[u, v]$ **then**
10. $d[u, v] = d[u, i] + D(l) + d[j, v]$;
11. update QMR path from u to v
12. return all-pairs QMR paths

Lines 1–12 of algorithm AQMR are identical to that of [3] except for the condition in line 8 which ensures that only the links on MR paths are considered in computing the quickest paths, and hence its correctness directly follows. Since each link in

AMRPN has n x n matrix to keep the labeling information, $O(n^2m)$ space is required. Therefore, the following theorem is as follows.

Theorem 2.1 *The all-pairs quickest most reliable paths for any size of message can be computed by algorithm AQMR with time complexity $O(n^2m)$ and space complexity $O(n^2m)$.*

3 Conclusion

In this paper, we introduced the quickest the most reliability paths for all pairs of nodes in a given network G for any message size σ. These algorithms match the best known computational complexity for the classical all-pairs quickest path problem, namely without the reliability considerations. It would be interesting to obtain all-pairs algorithms for other variations of the quickest path problem such as general bandwidth constraints [9], random queuing errors [10], and dynamic bandwidth constraints [11]. Another future direction is the on-line computation of quickest paths so that path's quality can be traded-off for computational speed.

References

1. Y. L. Chen, Y. H. Chin, "The quickest path problem", *Computers and Operations Research* vol. 17, no. 2, pp. 153–161.
2. D. T. Lee and E. Papadopoulou. "The all-pairs quickest path problem", *Information Processing Letters*, 45(5): 261–267, 1993.
3. J. F. Mollenauer, "On the fastest routes for convoy-type traffic in flow-rate-constrained networks", *Transportation Science*, vol. 10, pp. 113–124, 1976.
4. N. S. V. Rao and S. G. Batsell, "Algorithm for minimum end-to-end delay paths", *IEEE Communications Letters*, vol. 1, no. 5, pp. 152–154.
5. J. B. Rosen, S. Z. Sun, and G. L. Xue, "Algorithms for the quickest path problem and the enumeration of quickest paths", *Computers and Operations Research*, vol. 18, no. 6, pp. 579–584.
6. Guoliang Xue, "End-to-End data paths: Quickest or most reliable?", *IEEE Communications Letters*, vol. 2, no. 6, pp. 156–158.
7. Y. C. Bang, S. Radhakrishnan, N. S. V. Rao, and S. G. Batsell, "On update algorithms for quickest paths", *Computer Communications*, vol. 23, pp. 1064–1068.
8. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms", New York: McGraw-Hill, 1990.
9. N. S. V. Rao and N. Manickam. General quickest paths and path-tables. *Computer Systems: Science and Engineering*, 17(4/5):235-239, 2002.
10. N. S. V. Rao. LetLets for end-to-end delay minimization in distributed computing over Internet Using two-paths. *International Journal of High Performance Computing Applications*, 16(3), 2002.
11. W. C. Grimmell and N. S. V. Rao. On source-based route computation for quickest paths under dynamic bandwidth constraints. *Journal of Foundations of Computer Science*, 1002, in press.

A Concurrency Control Algorithm for Firm Real-Time Database Systems

Seok Jae Lee, Jae Ryong Shin, Seok Il Song, Jae Soo Yoo, and Ki Hyung Cho

Department of Computer & Communication Eng., Chungbuk National University, 48 Gaesin-dong, Cheongju, Chungbuk, Korea, 361-763
{sjlee, jrshin, prince}@netdb.chungbuk.ac.kr,
{yjs, khjoe}@cbucc.chungbuk.ac.kr

Abstract. Unlike a conventional database system, whose main objective is to provide fast average response time, Real-time database systems (RTDBS) may be evaluated based on how often transactions miss their deadline, the average lateness or tardiness of late transactions, the cost incurred in transactions missing their deadlines. Therefore, in RTDBS, transactions should be scheduled according to their criticalness and the tightness of their deadlines, even if this means sacrificing fairness and system throughput. And it always must guarantee preceding process of a high priority transaction (HPT) as the 2PL-HP (two phase locking with high priority) method. 2PL-HP resolves a conflict through aborting or blocking of a low priority transaction (LPT). If HPT is eliminated in a system because of its deadline missing, an unnecessary aborting or blocking of LPT is occurred. To resolve the problem, AVCC (alternate version concurrency control) algorithm that outperforms 2PL-HP was proposed. However, AVCC must always create the alternative version and have additionally a technique to manage complex alternative versions. In this paper, we propose a new efficient scheduling algorithm, called Multi-level EFDF that combines EFDF (earliest feasible deadline first) and Multilevel Queue scheduling algorithm very ably, and a concurrency control algorithm, called 2PL-FT that prevents wastes of needless resources and eliminates an unnecessary aborting or blocking of LPT. We show through the performance evaluation that our algorithm achieves good performance over the other existing methods proposed earlier.

1 Introduction

Real-time database systems (RTDBS) are database systems whose transactions are associated with timing constraints such as deadlines. Therefore transaction needs to be completed by a certain deadline. Besides meeting transaction timing constraints, RTDBS need to observe data consistency constraints as well. That is to say, unlike conventional database systems, whose main objective is to provide fast average response time, RTDBS may be evaluated based on how often transactions miss their deadline, the average lateness or tardiness of late transactions, the cost incurred in transactions missing their deadlines.

Therefore, in RTDBS, transactions should be scheduled according to their criticalness and the tightness of their deadlines, even if this means sacrificing fairness and system throughput. And it always must guarantee preceding process of a high priority

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 524–532, 2003.
© Springer-Verlag Berlin Heidelberg 2003

transaction (HPT) as 2PL-HP (two phase locking with high priority). 2PL-HP resolves a conflict through aborting or blocking of a low priority transaction (LPT). However, if HPT is eliminated in a system because of its deadline missing, an unnecessary aborting or blocking of LPT is occurred. To resolve the problem, AVCC (alternate version concurrency control) that outperforms 2PL-HP in simulations for firm real-time transaction was proposed. However, the algorithm must always create the alternative version and have additionally a technique to manage complex alternative versions.

The objective of this paper is to present a new concurrency control algorithm, called 2PL-FT. In addition, we propose a new scheduling algorithm, called Multi-level EFDF that combines EFDF (earliest feasible deadline first) and Multilevel Queue scheduling algorithm very ably.

The rest of the paper is organized as follows. In Sect. 2, we review related works and Sect. 3 presents our proposed scheduling and concurrency control algorithms. And Sect. 4 shows experimental results. Finally, Sect. 5 gives concluding remarks.

2 Related Works

Concurrency Control algorithms for RTDBS are in general some extensions or combinations of traditional concurrency control techniques, i.e. two phase locking (2PL), optimistic concurrency control (OCC) or timestamp ordering (TO), which guarantee a serialization order among conflicting transactions. Most of these algorithms were developed only for firm deadline RTDBS which is the simplest environment [4]. The most important problem faced in RTDBS concerns the degradation of system performance due to aborts and restarts of transactions. These restarts are caused by concurrency control protocols trying to resolve conflicts between transactions[2].

In non real-time (conventional) database systems all transactions have the same priorities. Blockings and restarts are the usual approaches to resolve access conflicts. 2PL and OCC are representative techniques used for blocking and restart to resolve data conflicts, respectively. It is clear that 2PL-HP has wasted restart and wasted wait due to the semantics of the firm deadline while OCC has wasted execution. A wasted restart happens if HPT aborts LPT and then HPT is discarded as it misses its deadline. In other words, a transaction which is later discarded can cause restarts. A wasted wait happens if LPT waits for the commit of HPT and later HPT is discarded as it misses its deadline. In other words, a transaction which is later discarded can cause wait of a conflicting LPT. And a wasted execution happens when LPT in the validation phase is restarted due to a conflicting HPT which has not finished yet.

AVCC proposed in [3] always outperforms 2PL-HP in simulations for firm deadlines. Because HPT can proceed without aborting conflicting LPTs, they use a deferred update policy which updates on local copies of a data item and makes them global at commit time. It only needs to stop LPT until the completion (commit or abort) of the conflicting HPT. If HPT is discarded by missing its deadline it can execute the stopped LPT by resuming it. This is what it refers to as the stop/resume deferred restart policy. But AVCC always have to maintain the stopped version and

initiate the restarted version of LPT. Also each transaction might have multiple DR (deferred restart) versions and a single IR (immediate restart) version.

3 The Proposed Concurrency Control Algorithm

In this section, we introduce architecture of a new real-time database model and describe a feasibility test method, scheduling and concurrency control algorithms in detail.

3.1 Architecture

Fig. 1 shows architecture of a new real-time database model. The model is composed of a module to initialize information that is required to test feasibility of transactions, a real-time scheduling module, and a real-time concurrency control module. In an application, the number of supplied functions and transactions is not infinite. Therefore, it is possible to classify transactions into multiple types. In the proposed model, we separated transactions according to the types, and assigned a real-time property such as firm or soft and criticalness to each transaction.

In the init module, it acquires a real-time property and criticalness according to the type of a new transaction and decides the expected computation time by the average of total computation times of the transaction and the average of total system loads. New transactions are classified in a level by criticalness and their order is scheduled by the deadline in each level. The CPU scheduler allocates the CPU to the transaction with the earliest feasible deadline in the highest level. Therefore, it guarantees that important transactions are executed in the first place.

All firm real-time transactions that are released are dependent on the feasibility test. Through the test, a firm real-time transaction that has already missed or is about to miss its deadline is eliminated in a system. And the firm HPT that is real-time transaction with the higher priority at the time of conflict is dependent on the feasibility test. If the firm HPT is not feasible, the infeasible HPT is eliminated from the system. Therefore, it prevents wastes of needless resources and eliminates an unnecessary aborting or blocking of LPT that is a transaction with the lower priority.

3.2 Feasibility Test

The feasibility test is that it compares an expected computation time (Ei) of a transaction Ti, with its deadline (Di). For example, the feasibility test of a transaction Ti is executed as follows:

$$(D_i - t) \geq (E_i - C_i) \tag{1}$$

Where t is the current time and C_i is the real computed time of T_i until now. The E_i is computed based on the average computation time C_{avg}, the minimum computation time

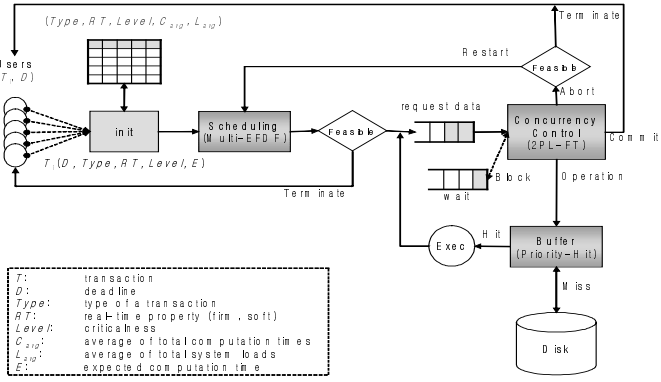


Fig. 1. Architecture of our model

C_{min}, the average system load L_{avg} and the current system load L_i. We propose two methods for computation of E_i.

$$E_i = \frac{(C_{\min} + C_{avg})}{2} \times \frac{L_i}{L_{avg}} \quad \ldots (Mid\text{-}type) \tag{2}$$

$$E_i = \max\left(C_{\min}, C_{\min} \times \frac{L_i}{L_{avg}} \right) \ldots (Min\text{-}type) \tag{3}$$

In the first method, called *Mid-type*, we use the intermediate value of C_{min} and C_{avg}. In the second method, called *Min-type*, we use a value that is less than or equal to C_{min}. In each method, we use (L_i/L_{avg}) for reflecting fluctuation rates of the system load. In case of *Mid-type*, transactions that its C_i is less than E_i can be damaged though the number of transactions is very few. But *Min-type* method overcomes that problem though the elimination ratio for infeasible transactions is decreased. Therefore we use *Mid-type* method for non-critical transactions and *Min-type* method for critical transactions. Note that non-critical transactions and critical transactions can be classified by a level of transactions.

3.3 Multi-level EFDF

The proposed Multi-level EFDF scheduling algorithm that combines EFDF (earliest feasible deadline first) and Multilevel Queue scheduling algorithm very ably eliminates a transaction that is about to miss its deadline and allocates CPU to the firm real-time transaction with the earliest deadline in the highest level. Accordingly transactions in lower levels must be in the ready queue until higher levels are empty. And if the priority of a new transaction is higher than the priority of the currently running transaction, CPU can be preempted by the new transaction. Fig. 2 shows the architecture of Multi-level EFDF.

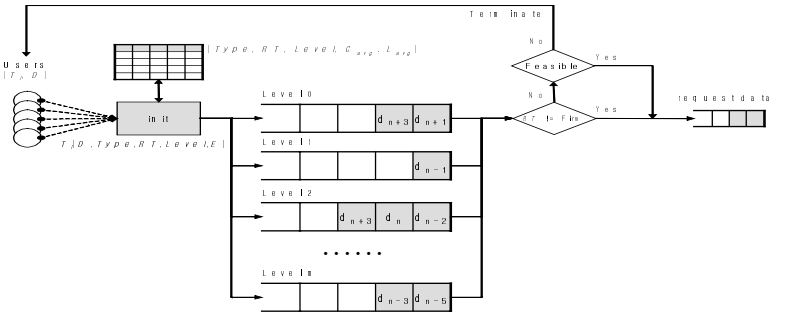


Fig. 2. Architecture of Multi-level EFDF

3.4 2PL-FT (Two Phase Locking with Feasibility Test)

The proposed real-time concurrency control algorithm, called 2PL-FT, eliminates an unnecessary aborting or blocking of LPT. And then it prevents wastes of needless resources and reduces the number of missed deadlines. To this end, the process of our algorithm is as follows:

(1) If a conflict occurs between a transaction T_H that already is locking on a data object O and a transaction T_R that is requesting the lock, it chooses as candidate transaction T_C the HPT between T_H and T_R.
(2) It decides whether T_C is about to miss its deadline or not (It is feasibility test for T_C).
(3) If T_C is infeasible, T_C is immediately eliminated in the system and LPT is continuously executed or acquires the lock. Table 1 shows the plan of the conflict resolution in detail.
(4) Otherwise T_C acquires the lock or is continuously executed. In contrast, LPT is restarted or blocked.

Table 1. Method to solve the conflict

Conditions		T_C is feasible	T_C is infeasible
$T_C = T_H$	T_H	Continuously executed	Eliminated
	T_R	Blocked	Acquires the lock
$T_C = T_R$	T_H	Restarted	Continuously executed
	T_R	Acquires the lock	Eliminated

For example, if T_C that is requesting the lock (T_R) is infeasible, the transaction is immediately eliminated in the system and LPT (T_H) is executed continuously. And if T_C with the lock (T_H) is infeasible, the transaction is eliminated and LPT that is requesting the lock (T_R) is executed.

Fig. 3 shows architecture of 2PL-FT, respectively.

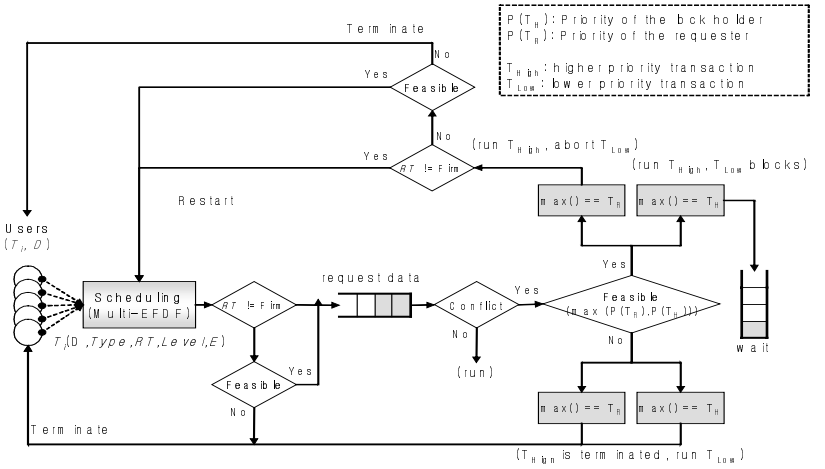


Fig. 3. Architecture of 2PL-FT

Fig. 3 shows that all firm real-time transactions that are released and firm HPT that is conflicted with LPT are dependent on the feasibility test. Through the test, (1) a firm real-time transaction that has already missed or is about to miss its deadline is eliminated in the system. (2) And if a firm HPT is infeasible, the HPT is eliminated too from the system. In contrast, LPT is not terminated or blocked but is executed continuously or acquires the lock.

4 Performance Evaluations

In this section we evaluate the performance of the proposed 2PL-FT algorithm with respect to restarting ratio and deadline missing ratio of transactions. The experiments were performed on a Pentium-III with 256Mbytes of main memory. The input and design parameters are shown in Table 2. These parameters are based on that of AVCC.

Table 2. The input and design parameters

Parameters	Description	Value
DBSize	The number of pages in the databases	1,000 pages
CPUtime	The processing time of a page	10 ms
NumLevels	The number of levels	20 levels
NumTypes	The number of transaction types	100 types
TransSize	The number of pages accessed per transaction	8~24 pages
Slackfactor	The tightness/slackness of deadlines	100~600 %

First of all, we evaluate an efficiency of feasibility test method with *Mid-type* or *Min-type* that are method for computing the expected computation time (E_i). Fig. 4 shows transaction ratio with missed deadline according as transaction arrival ratio is increased. In Fig. 4, *Mid-type* uses the intermediate value of C_{min} and C_{avg} to compute E_i and *Min-type* uses a value that is less than or equal to C_{min}. The "Average only" uses C_{avg} only.

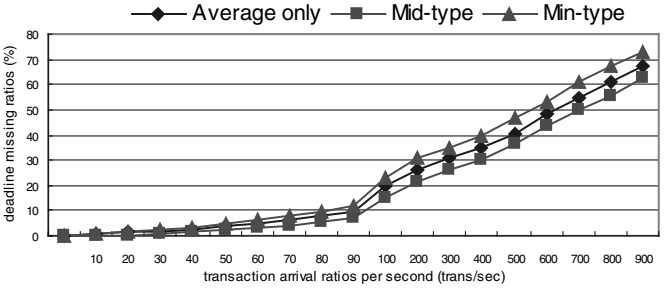


Fig. 4. Deadline missing ratios by feasibility test methods

The *Mid-type* and Average only show better performance than *Min-type* method. Because *Min-type* method applies the minimum execution time to the feasibility test, it cannot eliminate well firm real-time transactions that are about to miss their deadline.

Therefore it cannot fully prevent wastes of needless resources and eliminate an unnecessary aborting or blocking of LPT. Nevertheless we use *Min-type* for critical transactions because it is wastes of needless resources that the firm real-time transaction with the execution time less than the minimum value reflected fluctuation rate of the system load is executed uselessly. On the other hand *Mid-type* decreases the deadline missing ratio of firm real-time transactions very well. The reason is that many of firm real-time transactions that cannot be committed normally in their deadline are discarded.

From now on, we investigate restarting transaction ratios and deadline missing ratios of 2PL-FT and AVCC algorithms. In these experiments we use the feasibility test with *Mid-type* method only. Fig. 5 shows that the restarting ratio of 2PL-FT is about 38% less than that of AVCC, because AVCC always have to maintain the stopped version and initiate the restarted version of LPT conflicted with HPT. And then, we can see that the heavier resource competition, the more increased the restarting transaction ratio of AVCC. However, in 2PL-FT, LPT is restarted in case HPT is only feasible. Finally, deadline missing ratios of two algorithms are shown in Fig. 6 The deadline missing ratio of 2PL-FT is about 27% less than that of AVCC. The reason is as follows:

(1) Unnecessary restarting transactions of AVCC obstruct harmonious execution of normal transactions. On the other hand, infeasible transactions eliminated by 2PL-FT help an execution of the other transactions.

(2) AVCC with stop/restart version of LPT prevents an unnecessary aborting of LPT, whereas 2PL-FT with feasibility test of HPT prevents not only an unnecessary aborting of LPT but a blocking of LPT.

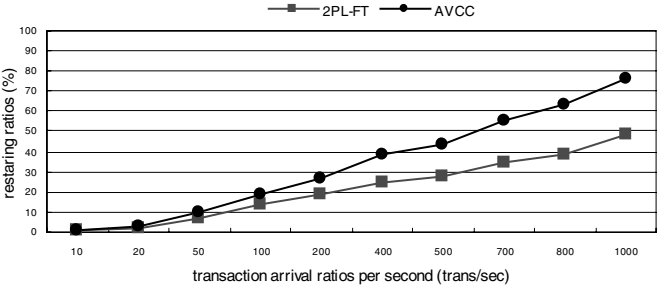


Fig. 5. Restarting transaction ratios

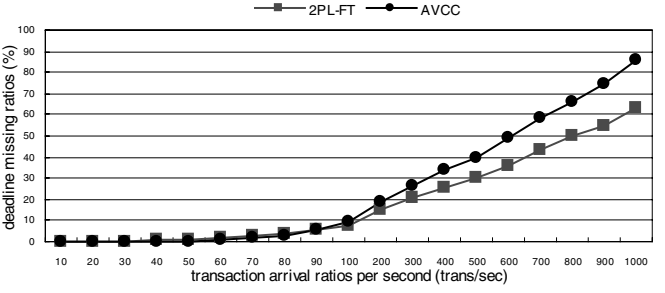


Fig. 6. Deadline missing ratios

5 Conclusions

In AVCC, HPT can proceed without aborting conflicting LPTs. They use a deferred update policy which updates on local copies of a data item and makes them global at commit time. But AVCC always have to maintain the stopped version and initiate the restarted version of LPT. Also each transaction might have multiple DR versions and a single IR version. In addition, AVCC must have additionally a technique to manage complex alternative versions.

In this paper, we have proposed a new efficient real-time concurrency control algorithm, called 2PL-FT that need not have alternative versions and a scheduling algorithm, called Multi-level EFDF that assigns the highest priority to the transaction with the earliest deadline in the highest level. And we have proposed the feasibility test to prevent wastes of needless resources. We have showed through the performance evaluation that the deadline missing ratio of 2PL-FT is about 27% less than that of AVCC. In the future research, we will adapt our idea to optimistic concurrency control algorithms such as OPT-BC, OPT-Sacrifice, OPT-Wait, Wait-50, Wait-X.

Acknowledgment. This work was supported by Korea Research Foundation Grant (KRF-2002-074-DS2501).

References

1. Abbott, R., and Garcia-Molina, H.: Scheduling Real-Time Transactions: A Performance Evaluation, Proceedings of the 14th Conference on Very Large Database Systems (1988)
2. Ben Kao and Hector Garcia-Molina: An Overview of Real-Time Database Systems, in Sang H. Son, editor, Advances in Real-Time Systems, chapter 19, Prentice Hall (1995)
3. D. Hong: Alternative Version concurrency Control Method for firm real-time database systems, Korea Information Processing Society, Vol. 5, No. 6 (1998) 1377–1389
4. D. Hong, Sharma Chakravarthy, Theodore Johnson: Locking Based Concurrency Control for Integrated Real-Time Database Systems, RTDB '96 (1996) 138–143
5. Haritsa, J. R., Carey, M., Livny, M.: Data Access Scheduling in Firm Real-Time Database Systems, Journal of Real-Time Systems, No. 4 (1992) 203–241
6. Haritsa, J. R., Carey, M., Livny, M.: Value-Based Scheduling in Real-Time Database Systems, The VLDB Journal, Vol. 2, No. 2 (1993) 117–152
7. Haritsa, J. R., Livny, M., Carey, M.: Earliest Deadline Scheduling for Real-Time Database Systems, Proceedings of IEEE Real-Time System Symposium (1991) 232–242
8. Haritsa, J. R., Livny, M., Carey, M.: On Being Optimistic about Real-Time Constraints, Proceedings of the 9th ACM Symposium on Principles of Database Systems (1990)
9. Huang, J., Stankovic, J., Ramamritham, K., Towsley, D.: Priority Inheritance in Soft Real-Time Database, Journal of Real-Time Systems, Vol. 4, No. 3 (1992) 243–268
10. Piotr Krzyzagorski: Concurrency Control in Real-Time Database Systems, EDBT Ph.D. Workshop (2000)
11. Ulusoy, O.: Processing of Real-Time Transactions in a Replicated Database Systems, Journal of Distributed and Parallel Database, Vol. 2, No. 4 (1994) 405–436

Secure Firewall Traversal in Mobile IP Network

Jung-Min Park[1], Min-Jung Jin[2], and Kijoon Chae[2]

[1] Intelligent System Control Research Center, KIST, Korea
pjm@amadeus.kist.re.kr
[2] Department of Computer Science and Engineering, Ewha Womans Univ., Korea
{mjjin,kjchae}@ewha.ac.kr

Abstract. With recent advances in wireless communication technology, mobile computing is an importance research area. Mobile IP is designed to provide IP services to roaming nodes. Mobile users take advantage of this protocol to obtain the services as if they were connected to their home network. In many cases mobile users is connected through a wireless link and is protected by corporation's firewall in virtual private network. In order to have a successful deployment of Mobile IP as an extension of a private network, security services should be provided as if the mobile node were attached to its home network. In this paper, we propose the security mechanism of combining Mobile IP and IPSec tunnels, which can provide secure traversal of firewall in a home network. The simulation results show that the proposed mechanism provides the secure and efficient communication.

1 Introduction

Mobility in IP networks is a significant issue due to the recent increase of many portable devices such as notebook and PDA and the development of wireless network interface. Many popular applications such as E-commerce and remote access require transmission of highly sensitive information, often over wireless links. Mobility implies higher security risk than static operation in fixed network, because the traffic may at times take unexpected network paths with unknown or unpredictable security characteristics. Technologies should be developed both IP security and mobility over wireless links.

Mobile IP has been designed with the IETF [1] to provide robust communication of mobile users as they roam from place to place [2][3][4]. With growth of public networks and the deployment of security protocols, enterprises around the world are trying to securely extend their network over the public backbones, establishing what are called Virtual Private Network (VPN). Practical applications of Mobile IP are likely to occur where a private network is extended over Internet. In many cases mobile users is connected through a wireless link and is protected by corporation's firewall in VPN. In order to have a successful deployment of Mobile IP as an extension of a private network, mobile users must be authenticated safely when roaming

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 535–544, 2003.
© Springer-Verlag Berlin Heidelberg 2003

from own network to another and only authorized users must be allowed the access to the private network. In Mobile IP, a tunnel is built up from a Home Agent (HA) to the Mobile Node's (MN's) Care-of-Address (CoA). Since this tunnel can be traverse the home network's firewall, the tunnel endpoints have to be administrated carefully with regard to security. Network operators are not likely to accept any tunnels through their firewalls. Additionally, the firewall has to inspect both outer and inner header of tunneled packets to ensure the packets are delivered only if their exact destination address is the home address of a registered MN, which may cause efficient problems. Since an MN abroad is logically still a part of its home network, security associations providing at least data origin authentication are necessary between each MN and a HA or a firewall in home network. Data encryption is required to keep confidential data transmitted to the MN as secure as it would be inside the home network. Another problem regarding Mobile IP and firewall is the use of topologically incorrect source addresses for packets sent by an MN. Few firewalls will allow such packets to exit a foreign network or to enter the MN's home network if the destination address is located there.

The IETF has also developed the suite of IPSec protocols [5] to provide capabilities to secure communications across the Internet at the network layer. With IPSec, security mechanisms are directly applied to IP packets. Additional headers are added to an IP packet to transport security related information and to identify protected packets.

In this paper we propose the secure tunneling mechanism of combining Mobile IP and IPSec tunnels, which can provide secure traversal of firewall in home network while mobile nodes are roaming abroad. This paper is organized as follows. Section 2 introduces the existing proposal for firewall over Mobile IP. Section 3 describes the our proposed mechanism. Section 4 presents our simulation environment and simulation scenarios and evaluates the simulation result. Finally, Section 5 summarizes and concludes this paper.

2 Related Works

Zao and Condell [8] describe the scheme to negotiate the use of IPSec on the Mobile IP tunnels and a procedure to establish these tunnels. IPSec is used to replace IP-in-IP tunneling, which is basic tunneling mechanism in Mobile IP. They defined the six possible unidirectional tunnels(HA-MN, HA-FA, MN-FA, and vice versa). The tunnel establishment has to be done by an appropriate protocol such as ISAKMP. The approach is required minimal overhead, but it is quite inflexible.

Binkley and Ricardson [9] propose a security model using IPSec for mobile user and networks providing access-points to these mobile nodes. They propose the bidirectional IPSec tunnels between a HA and a MN. Since they consider secure mobility problem as special case of ad-hoc network, it may be too complex in Mobile IP network.

Montenegro and Gupta [10] describe the scheme of combining Mobile IP and IPSec tunnels which can protect the home network when mobile nodes are roaming abroad. They propose to use SKIP for key management, authentication, and encryp-

tion. The approach seems to be easy and efficient to solve Mobile IP security problem, but all mobile nodes must support SKIP.

Mink et al. [11][12], propose the hierarchical mobility architecture FATIMA (Firewall-Aware Transparent Internet Mobility Architecture). In order to achieve the security, they define a special gateway in firewall and IPSec tunnels are established among those nodes. The approach is transparent to current Mobile IP and supports the micro-mobility. However, since all security-critical functionality is concentrated in FATIMA gateway, it causes a single point of failure. Moreover, the high cost in design and implementation of the architecture is additional disadvantages.

3 Proposed Secure Tunneling Mechanism

3.1 Secure Tunneling Mechanism

We describe the proposed secure tunneling mechanism. To simplify the security management we regard the firewall as screen-subnet firewall, where the private network and Internet are separated by de-militarized zone. Signaling between the mobile node and the firewall requires message authentication, integrity and replay protection, while mobile nodes roam foreign networks. We use an IPSec tunnel to protect the Mobile IP tunnel between firewall and MN, which traverse the insecure parts of the Internet, similar to the proposal by Montenegro and Gupta. Since all packets over the Internet are authenticated and encrypted by IPSec, the establishment of IPSec tunnel is able to implement a lightweight security in Mobile IP without additional security mechanism. Within the private network, we use the Mobile IP tunnel. Therefore, the traffic between HA and firewall is sent through IP-in-IP tunneling. The proposed tunneling mechanism provides authentication, integrity and replay protection of all IP packets sent during Mobile IP registration. Fig. 1 shows the proposed secure tunneling mechanism.

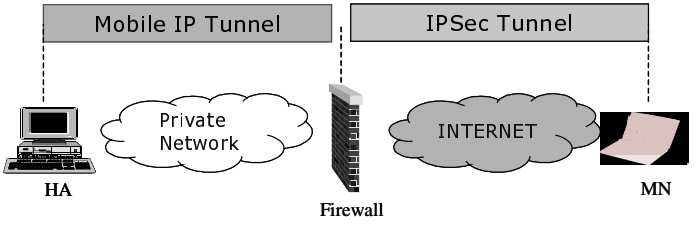


Fig. 1. Secure Tunneling Mechanism

We assume that MNs acquire their IP address from Dynamic Host Configuration Protocol(DHCP) [13] servers while MN is roaming abroad. DHCP allows to dynamically assign an IP address to a node and is a wide spread mechanism to reduce the administration overhead in dynamic changing network configuration. As regarding to the

practical implementation, the combination of Mobile IP with DHCP is used. Data transmission between the MN and the Correspondent Node(CN) takes place via the HA because of security reasons. It is also possible to communicate with the CN directly provided no security is needed. The encrypted and encapsulated Mobile IP packets are decrypted and decapsulated by the firewall and delivered to the HA. The HA finally decapsulates these Mobile IP packets and delivers them to the appropriate receivers.

3.2 Operation of Secure Tunneling Mechanism

When a MN moves new point of attachment, proposed tunneling mechanism operates as follows. When a MN enters new foreign network, the MN stops the old IPSec tunnels. The MN gets the co-located CoA from DHCP servers. Thereafter, bi-directional IPSec tunnel establishes between MN and firewall. Then, the MN registers at the HA and communicate with other CN. Fig. 2 shows the exchange of messages of proposed secure tunneling mechanism by time sequence.

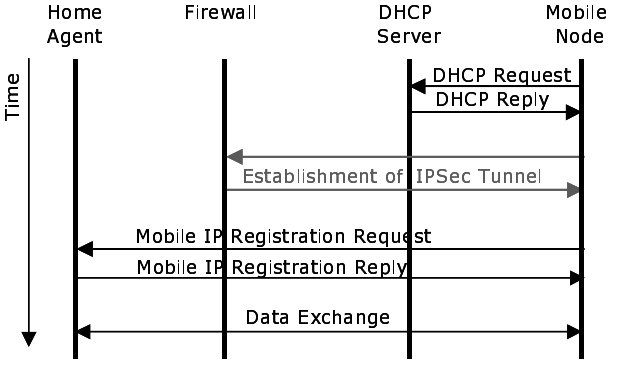


Fig. 2. Message Sequence Chart

4 Performance Evaluation

4.1 Simulation Environment

We analyze the performance of proposed mechanism through the simulation. In the following, we describe the four types of scenario for simulation and present the simulation results. Each scenario is simulated on Desktop PC with Intel Pentium IV 1G Hz CPU and run under Windows 2000 server operation system. In our simulation, we use the SDL(Specification and Description Language) which is standardized as ITU Recommendation Z.100. Because of SDL's suitability for real-time and stimuli-response

systems, it is well received in the telecommunication community and is used both in standards making and product development.

Fig. 3 shows our simulation network topology. In the simulation topology, the wired link bandwidth is 10 Mbps with 5 ms/km latency. The wireless side has 2 Mbps with 7 ms/km latency.

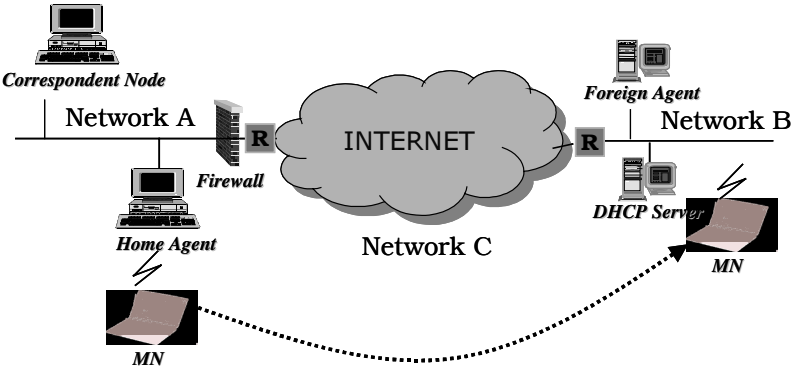


Fig. 3. Simulation Topology

Assumptions. We use the pre-established symmetric keys to eliminate the necessity of using asymmetric cryptography which represent more processing overhead for the system. We decide to use pre-negotiated security attributes between the HA and the MN and between the firewall and the MN for Mobile IP registration. Since all the security protocols simulated give various alternatives in terms of algorithms for the different security attributes, we have to decide the appropriate algorithms. For this purpose, we consider several aspects such as algorithm that have proven enough reliability, algorithm that comply with the basic requirement of the protocols, and computational time of cryptographic functions involved in the algorithm. We choose the HMAC-MD5 as the authentication algorithm to compute the keyed-hash and DES-CBC as the encryption algorithm.

4.2 Simulation Scenario

The simulation measures total latency involved in the necessary protocol operations. An important issue is the performance of authentication on the proposed tunneling mechanism. Our objectives focus on the performance impact of the registration procedure on the proposed secure tunneling scheme. To obtain the significant result, we describe the simulation scenarios, which take into account four different types of tunneling schemes and analyze this scenarios. The first one is basic routing mechanism, the second employs mobile IP tunneling, the third scenario employs the tunnel mode IPSec tunneling with AH, and the fourth use the tunnel mode IPSec tunneling with

ESP. Fig. 4 depicts the simulation environment and scenarios for the performance analysis.

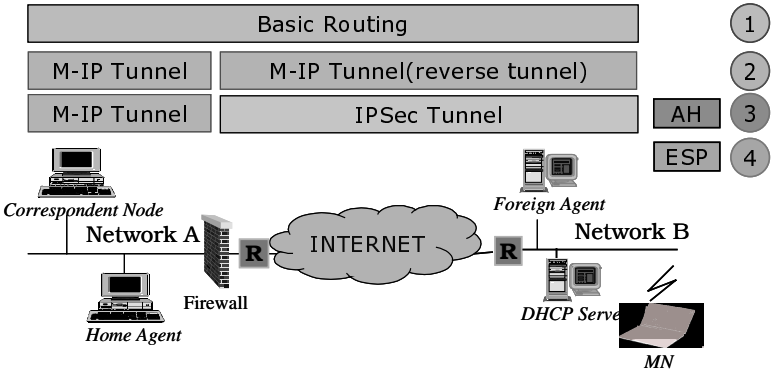


Fig. 4. Simulation Scenario

Scenario 1: Basic Routing. The first scenario is to measure the latency caused by raw data packet without tunneling and cryptographic operation.

Scenario 2: Mobile IP Tunneling. The second scenario is to measure the latency caused by Mobile IP tunneling. In this case, there are two kinds of tunnel. One is the Mobile IP tunnel between HA and firewall, the other is the reverse tunnel between firewall and MN. Since sending a packet from a MN located within a foreign network cannot be allowed by a firewall, we use reverse tunneling [14]. The basic authentication between the HA and the MN is represented in this scenario. The main reason of this stage of scenario is to get the least computational time for authentication compared with IPSec tunneling.

Scenario 3: IPSec Tunneling with AH. The third scenario is to measure the latency caused by the IPSec tunneling with AH protocol between firewall and MN for secure communication. The processing time of authentication is directly proportional to the type of security algorithm employed. Since there exist no uniform authentication standard in Mobile IP, we have measured the performance influence of authentication algorithm such as HMAC-MD5 and HMAC-SHA1.

Scenario 4: IPSec Tunneling with ESP. Compared with the previous scenario, this scenario provides the confidentiality as well as authentication. The scenario is to measure the latency when applying the IPSec tunnel with ESP protocol. This delay depends on the transported data in the ESP and security algorithm for authentication and encryption. The performance has been measured the total latency and the processing time of cryptographic function. In order to compare the influence of cryptographic function processing, we use the two kinds of hash algorithm such as HMAC-

MD5 and HMAC-SHA1 for authentication, and encryption algorithm such as DES-CBC and IDEA-CBC for encryption.

4.3 Simulation Result

We perform simulations of the scenarios described in section 4.2. Simulation measures the latency involved in the necessary protocol operations. Our objective focuses on the processing overhead of cryptographic operation produced by tunneling mechanism over Mobile IP network. Therefore, we measure the influence of cryptographic processing time in each node on the tunneling mechanism as well as total latency in each scenario. We evaluate the performance of Mobile IP registration using IPSec tunnel according to the following measures: total latency, time spent for cryptography computations, and time spent for message transmission.

Fig. 5 shows total latency of registration in each scenario. Total latency includes the message transmission delay, message generation time for registration request and reply, updating time in table, verification time for nonce or timestamps, and encryption/decryption time. In Fig. 5, the difference between Mobile IP tunneling(scenario 2) and IPSec AH tunneling(scenario 3) in total latency is not noticeable. But, there is a significant difference between IPSec AH tunneling(scenario 3) and IPSec ESP tunneling(scenario 4). Fig. 6 shows the time spent for registration in each node. Since a mobile node has limited resources, the processing time on MN is important. We can see that the processing time produced by a MN in case scenario takes shorter than any other. In the processing time on MN, IPSec AH tunneling is almost same as Mobile IP tunneling. IPSec ESP tunneling is three times longer than basic routing and is 1.7 times longer than IPSec AH tunneling. IPSec ESP tunneling has obvious effect on performance because of encapsualtion. In Fig. 5 and Fig. 6, we can conclude that IPSec AH tunneling is an efficient security mechanism to provide the security service such as authentication and integrity in firewall protected network, even though it doesn't provide the security service such as confidentiality.

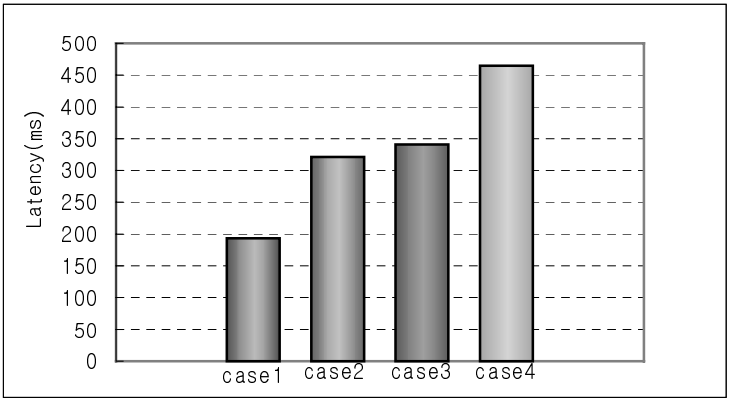


Fig. 5. Total Latency

Fig. 7 shows the cryptographic function processing time of each node according to each scenario. Fig. 8 shows the processing overhead of each node in scenario 3. Fig. 9 indicates the cryptographic processing overhead of each node according to cryptographic function in scenario 4. In Fig. 9, we can find that HMAC-MD5 for authentication algorithm and DES-CBC for encryption algorithm are appropriate.

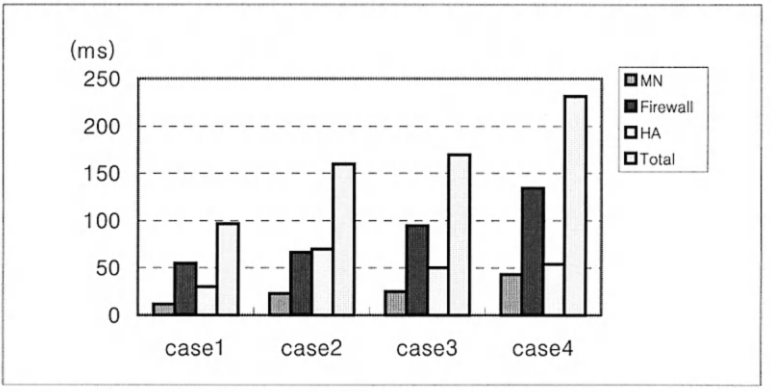


Fig. 6. Processing overhead for registration in each node

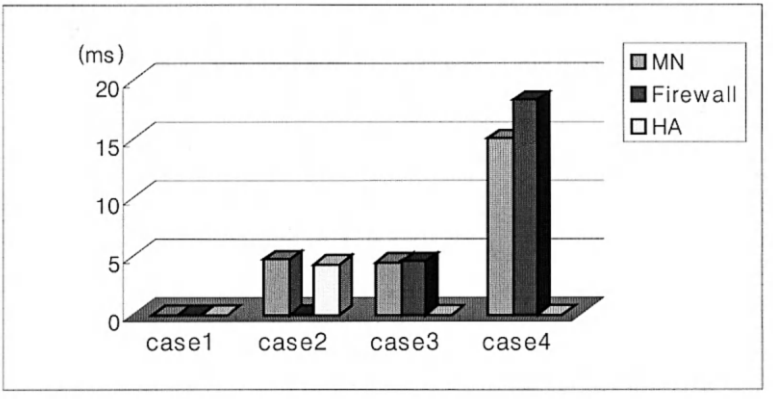


Fig. 7. Cryptographic processing overhead of each scenario

5 Conclusion

In this paper we have proposed the secure tunneling mechanism of combining Mobile IP and IPSec tunnels, which can provide secure traversal of firewall in home network while mobile nodes are roaming abroad. Additional functionality needs to be intro-

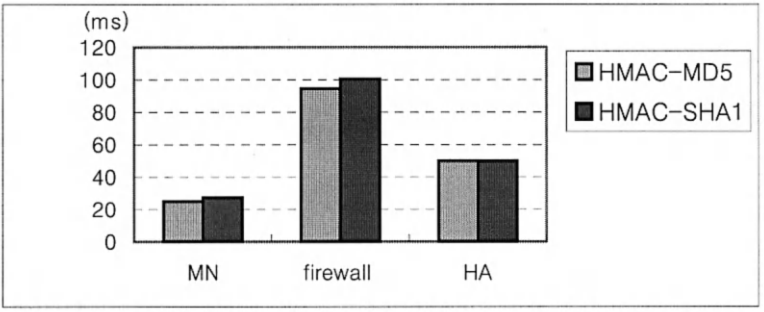


Fig. 8. Processing overhead in Scenario 3

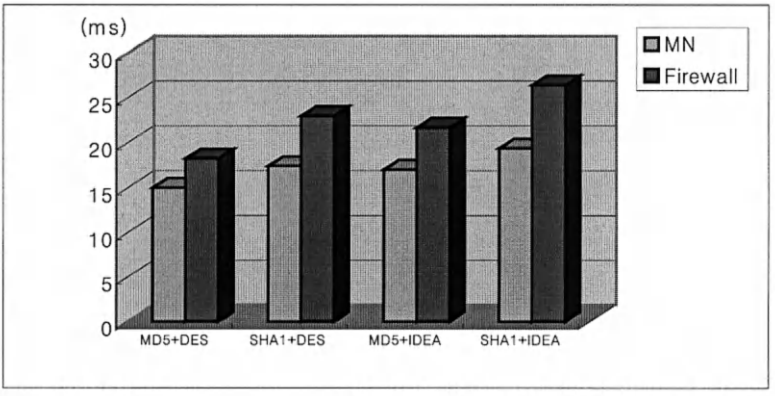


Fig. 9. Cryptographic processing overhead in scenario 4

duced only in the firewall and mobile node for IPSec tunnel whereas the rest of the Internet remains unchanged. The proposed mechanism can be efficiently managed since most function is concentrated in the home network's firewall. The Mobile IP for secure traversal of firewall in home network is transparent to the foreign network, which adds flexibility and efficiency for a network to host Mobile IP nodes and to provide the security services. In mobile networks, mobile nodes will require their security function such as authentication, confidentiality and integrity in a limited time. The proposed tunneling mechanism is not required significant processing overhead on mobile node for security. From simulation results, we may conclude that the proposed mechanism is reasonable performance toward the secure mobility. Our research will be extended to support firewall-protected foreign network and to minimize the delay in frequent handoff.

Acknowledgement. This work was supported in part by the University Research Program of Ministry of Information and Communication, 2002, Korea.

References

1. Internet Engineering Task Force, http://www.ietf.org
2. Perkins, C.: Mobile IP Design Principles and Practices. Addison-Wesley Wireless Communication Series (1997)
3. Perkins, C.: Mobile IP. IEEE Communications Magazine, May (1997)
4. Perkins, C.: IP Mobility Support for IPv4. RFC 3344, Aug. (2002)
5. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol. RFC 2401, Nov. (1998)
6. Kent, S., Atkinson, R.: IP Authentication Header. RFC 2402, Nov. (1998)
7. Kent, S., Atkinson, R.: IP Encapsulating Security Payload (ESP). RFC 2406, Nov. (1998)
8. Zao, J.K., Condell, M.: Use of IPSec in Mobile IP. Internet Draft Nov. (1997)
9. Binkley, J., Richardson, J.: Security Consideration for Mobility and Firewall. Internet Draft, Nov. (1998)
10. Montenegro, G., Gupta, V.: Sun's SKIP Firewall Traversal for Mobile IP. RFC 2356, (1998)
11. Mink, S., Pählke, F., Schäfer, G., Schiller, J.: FATIMA: A Firewall-Aware Transparent Internet Mobility Architecture. Proceedings of ISCC 2000, Antibes, France, (2000) 172–179
12. Mink, S., Pählke, F., Schäfer, G., Schiller, J.: Towards Secure Mobility Support for IP Networks. (2000) 555–562
13. Droms, R.: Dynamic Host Configuration Protocol. RFC 2131, Mar. (1997)
14. Montenegro, G.: Reverse Tunneling for Mobile IP. RFC 3024, Jan. (2001)

Buffering Management Scheme for Multimedia Synchronization in Mobile Information System

Gi-Sung Lee[1], Jeung-gyu Jee[2], and Sok-Pal Cho[3]

[1] Dept. of Computer Science, Howon Univ., Korea
ygslee@sunny.howon.ac.kr
[2] Korea Research Foundation, Korea
[3] Dept. of C&C . Eng. Sungkyul Univ., Korea

Abstract. In this paper we propose on buffering management scheme to address the problem how to maintain a smoothing stream presentation in multimedia mobile environments. For this case, a feedback scheme is used for sustaining a continuous presentation of multimedia stream and a presentation control scheme for maintaining a normal level of buffer. The proposed scheme shows not only a seamless play-out but also the higher play-out rate and the lower loss rate.

1 Introduction

The explosive growth of Internet access in parallel with the technological advances in mobile communications has motivated multimedia applications in mobile networks. For this reason, user's need for diverse multimedia services can't be admitted with the previous client/server based service structure. This limitation can be overcome by extending the concept of service providing systems to mobile systems connected with the mobile networks. Particularly, the interest of a distributed store technique for multimedia services is increasing in mobile information environments. It is different to play out multimedia data in a Mobile Host(MH) in mobile environments because high data loss rate and delay, and low bandwidth. For these reason, buffer is used in the distributed multimedia systems. The Base Station(BS) receives many sub-frames from multimedia servers. However the unexpected delay and the increase in traffics result in the decrease in the possibility of playing out sub-frames, in the case where the play-out time in the BS is faster then the arrival time from the multimedia server. The buffer at the BS provide temporary storage for multimedia data, in order to smooth potential delay and jitter introduced by the wired network and facilitate synchronization between the multimedia servers and the BS. Once the play-out starts, data is delivered with little delay variation. Among the previous buffering schemes for multimedia synchronization, a forward feedback scheme has a problem that it may not control the number of sub-frames in case of the underflow and the overflow occurrence. Due to this reason, the buffer level is always controlled with the same level, and this results in the media loss. In this paper we propose a smoothing buffering management scheme for sustaining a continuous

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 545–554, 2003.
© Springer-Verlag Berlin Heidelberg 2003

presentation for multimedia stream, and a presentation control scheme for maintaining a stately level of the buffer. As a result of the proposed methods, the safe state of buffers is continuously maintained because the filtering function and the network state variables are reflected in the value of feedback, and the play-out time is adaptively modified. MH is also absolved from his responsibility for synchronization because the synchronization for multimedia data is performed only in BS.

2 Related Work

In [4][7], a start-up time before transmitting streams is defined by sending a roundtrip packet in evaluation phase, and the skew problem between the buffer size and media size is solved by readjusting the number of servers with the maximum jitter and the first transmission time in other servers. This results in the increase in play-out rate by maintaining the safe state of buffer. However, it takes much time to move into the safe state from the underflow or the overflow because the network state is reflected in the value of filtering function needed to maintain the safe state. It is also hard to sustain the safe state because this method doesn't consider the play-out time. In[5], a BS serves as an interface between the wired and the wireless networks, and buffering at the BS for interfacing with the wired networks is considered to smooth inter-packet jitter delay. This method attempts only to apply synchronization to channel allocation. The buffer management and the feedback scheme in which the received multimedia units are registered to the buffer at a MH, and their play-out time is controlled based on the middle of the buffer size, is studied in [2][3]. However this method can not support the required QoS level by changing a play-out time even though the state of wireless networks is normal. In[1], a synchronization scheme for wireless clients and distributed multimedia systems using a quasi-sink to control synchronization was proposed. This method defines efficiently the role of system elements such as multimedia server, BS, and MH. However, the synchronization for multimedia data is performed only in the start point of transmission. That is, there is no policy for play-out in this method.

3 Buffering Management Scheme for Multimedia Synchronization

3.1 System Model in Mobile Networks

Fig.1 shows the system model in mobile networks. The overall system configuration consists of three parts: multimedia servers k, BSs m and MHs n. A BS should communicate with MHs n, and a MH should access multimedia servers over wired networks through a BS. This system allows the BSs to manage the buffers and the transmission time using the arrival time and delay jitter from the multimedia servers. This provides a solution to the problems of the MH with small memory and the BS with the limited resources than the sites in the wired

networks. In this system, a movie that is divided into objects i is separately registered to servers k. This system has the minimization of start up time and the decrease of the network traffic and the buffer size. A multimedia data can be divided into smaller Sprit Streams with logical time using a sub-frame stripping technique, and is separately registered to each server. The smaller stream is referred as a sub-frame. A message manager in a multimedia server skips and transmits sub-frames using the offset of a control function from a feedback manager of a BS. The BS registers the arrival time of sub-frames from each server, and gets the transmission start times from multimedia servers through dummy packets. A communication manager in a multimedia server classifies feedback packets and messages from a BS, and reports this information. A communication manager in a BS sorts the ordering of the sub-frames transmitted from a multimedia server.

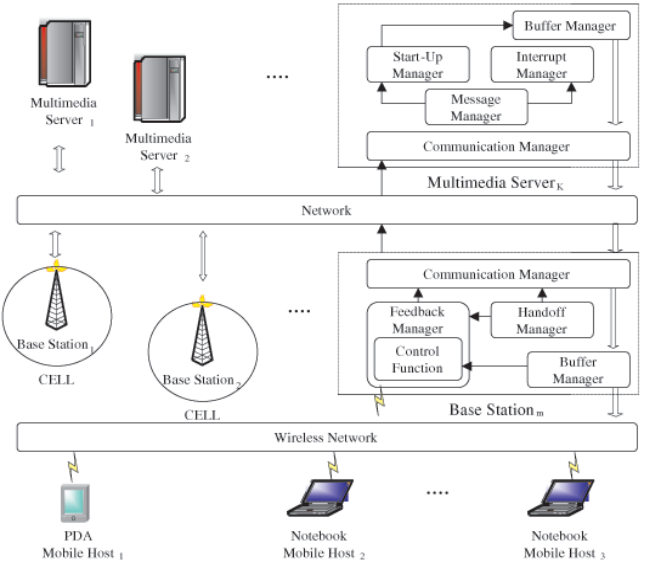


Fig. 1. System model in mobile networks

3.2 Buffering Management Structure

As shown in Fig. 2, a buffer manager in a BS controls the state of buffers. The buffer state can be classified into five levels: normal level, upper threshold level, upper control level, lower threshold level and lower control level.

1. normal level
   The state of the buffer is normal level. In this case, the buffer manager receives continuously media data. It also checks sub-frame numbers and the jitter scope, and registers the checked information.

2. upper threshold level and upper control level
   The possibility of suffering from the buffer overflow gets higher if the buffer reaches the upper threshold level, and the buffer overflow is originated if the buffer reaches the upper control level. If the state of the buffer is the upper threshold level, the buffer manager informs a MH of the sub-frames and the play-out time obtained from feedback policy of a feedback manager.
3. lower threshold level and lower control level
   The possibility of suffering from the buffer underflow gets higher if the buffer reaches the lower threshold level, and the buffer underflow is originated if the buffer reaches the lower control level. If the state of the buffer moves to the upper threshold level, the manager of the buffer level informs to a MH of a new play-out time obtained from a feedback policy of a feedback manager.

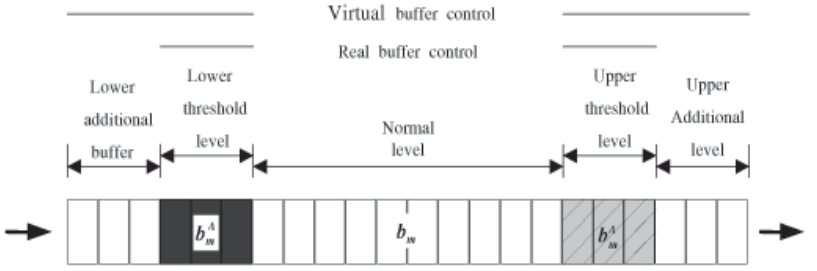


Fig. 2. Buffering State in a BS

3.3 Buffering Management Algorithm

We describe an algorithm in which the arrival rate and the play-out rate are adaptively adjusted to maintain the safe state in the buffer in a BS. The proposed algorithm is the following thing.

Procedure Buffering Management Algorithm

Procedure Start-up Time

Begin

BS_{time}=Current BS Time;

For $i = 0, 1 < k, i++$ do

Send(S_i, DummyPacket$_i$); /*S_i is each multimedia server. This function sends dummypackets to each server*/

Receive(S_i, D_k^i, Dummypacket$_i$);/*D_k^i is ith delay time of a server K.
This function computes the round trip time of each server */

Start-Up$_i$ = Max $D_k^i - D_K^i$; /*Max D_k^i is the maximum delay time,
Start-Up$_i$ denotes the transmission start time*/

Set Start-Up Time for Server S_k /* Set the thransmission srart time for each multimedia server */

End For

End /*Procedure Start-Up Time*/

Procedure Feedback /*Buffer Point at Normal Lever in BS*/

Begin

$q_{t,m}$=current buffer_point;

$C(q_{t,m}) = \alpha \cdot \overline{b}_{t-1,m} + (1-\alpha) \cdot q_{t,m}$; / * $\overline{b}_{t-1,m}$ is smoothed buffer, α is smoothing factor, $q_{t,m}$ is the buffer level, a filtering function*/

If DT_k^i = = First Variance Then /*The extent of the first variance in the average arrival time */

$O_{t,m} = q_{t,m} - \overline{b}_{t,m} - N_p$; / * N_p is Mobile Network State and the value is 1.*/

Else If DT_k^i == Second Variance Then /*The extent of the second variance in the average arrival time */

$O_{t,m} = q_{t,m} - \overline{b}_{t,m} - N_p$;

/*N_p is Mobile Network State, and the value is 2 */

End If

End

Procedure Nomal_Level /*Buffer Point at Normal Level in BS */

Begin

$\rho_i = 125$; /* A expected play-out time */

If buffer_point = = Normal level Then

$\tau_t = \rho_i$ /* τ_t is real play-out time */

End If

End

Procedure Upper_level /* Buffer Point at Upper Level in a BS */

Begin

$BL_{allsize} = BL_{allsize} + 1$; /* $BL_{allsize}$ is the number of the buffers corresponding to thethreshold level*/

BL_{psize}; /* BL_{psize} is the current number at upper threshold level */

$\omega = 0$; /* A weight value */ $\rho_i = 125$;

If buffer_point = = Upper level Then

Begin

$\omega = BL_{allsize}/BL_{psize}; \tau_t = \rho_i - (\rho_i \times \omega)$;

End If

End

Procedure Lower_level /* Buffer Point at Lower Level in a BS */

Begin

$BL_{allsize}; BLpsize; \omega = 0; \rho_i = 125; \lambda = 10;$

/* λ is the maximum delay jitter */

If buffer_point = = Lower level Then

Begin

$\omega = BL_{allsize} - BL_{psize}; \omega = \omega/BL_{allsize}; \tau_t = \rho_i + \lambda \times \omega$;

End If

End

Begin /* Start Main Procedure */

Call Start-Up Time; $\Delta = DT_k^{max} - DT_k^{min}$; /* Δ is jitter */

If buffer_point = = Normal level Then

call normal level;

Else If buffer-point = = Upper level Then

Begin

If $DT_k^i < \Delta$ Then /* DT_k^i is the sub-frame's delay time */

Begin

Feedback_value = Call Feedback;

Send (S_k, Feedback_value, DT_k^i)

End

Else

Continue;

Call Upper_Level;

End

Else If buffer_point = = Lower level Then

Begin

Feedback_value = Call feedback;

Send (S_k, Feedback_value, DT_k^i);

End

Else

```
        Continue;
    Call Lower_Level;
    End
    End If
    End
    End                                  /* Buffering Management Algorithm */
```

4 Performance Evaluation

We describe a simulation model that developed to evaluate the performance of our buffering management scheme. In our simulation environments, the interface and algorithm has been implemented by Java development kit JDK 1.3 and the result has been stored as a simulation.mdb in MDB. 1Kbyte of audio data is encoded by PCM, and resolution of video frame is 120X120. Twenty-four frames are used for encoding. It is assumed that wired networks are used for communication between BSs and multimedia servers, wireless networks for the communication between MHs and BSs. To carry out a proper work for each packet, the information used in actual simulation was calculated with Poisson distribution and the network delay time was applied equally to two types of the media. The number of frames is 1,000, and the maximum delay jitter time is 600 ms. The play-out rate and loss rate using the presentation time with the feedback policy and play-out control policy to keep the buffer level at a normal state is compared to the previous methods. Fig. 3 shows the buffer water-marker wherein The x-axis indicates the number of frames and the y-axis the buffer level. It indicates that the frame number in a buffer is delayed from 200 to 350 and thus the buffer is getting out of a normal state because the arrival time of frames is delayed. Fig. 3 also shows the state in which the buffer water-marker changes to the normal state by the feedback policy with the feedback management. It also shows that underflow is reached later than that by the previous policy. It is known that it is faster to progress to a normal state in variance 1 than that by a previous method. Also, underflow reached later variance 2 better than variance 1. The overflow occurs between 800 and 960 frames as shown in Fig. 3 We can see buffer water-marking of variance 1 and 2 is stable then that of the previous scheme. It also shows that the previous schemes don't control the overflow between 200 and 360 frames, while the proposed scheme prevents the underflow between 200 and 300 frames. The overflow state can be controlled between 800 and 960. Fig. 4 shows results of the play-out policy with buffer water-marker. It indicates that the frames between 200 and 350, and between 720 and 820 are waited and skipped, respectively. The play-out time is changed

between 200 and 350, and also the overflow gets to be reduced by the decrease in play-out time from 750 to 960.

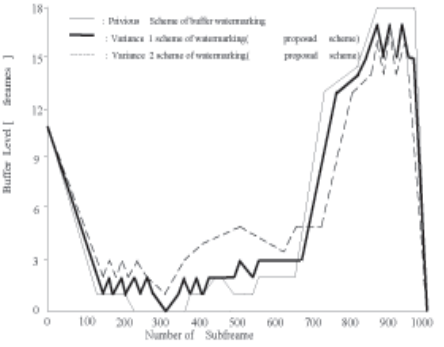


Fig. 3. Buffer Wate Marker at BS

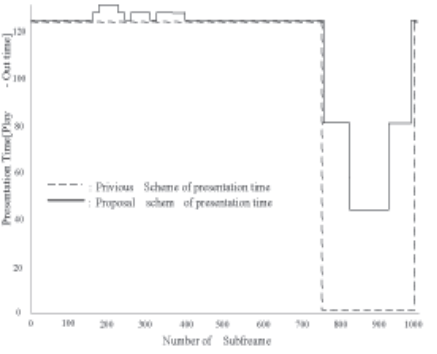


Fig. 4. Playout Time at MH

5 Conclusions

In this paper we have proposed a buffering scheme of synchronization which has been brought into a core technology for supporting multimedia services. We have shown that the proposed scheme decreases the buffer size and traffic volume by separately storing multimedia data with many servers. We also prevent the buffer from experiencing the underflow or the overflow by controlling the buffer level in a presentation control scheme, and maintain the safe state of the buffer by using the evaluated arrival time as one of feedback variables in a feedback scheme. The propose scheme allows the BSs to manage the buffers and the play-out policy,and provides a solution to the problems of the small memory and the limited resources.

Acknowledgements. This work was supported by the Foundation Fund of University in the Ministry of Information & Communication 2002.

References

1. Azzedine Boukerche, Sungbum Hong and Tom Jacob, "MoSync : A Synchronization Scheme for Cellular Wireless and Mobile Multimedia System," Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems IEEE 2001
2. D. H. Nam and S. K. Park, "Adaptive Multimedia Stream Presentation in Mobile Computing Environment," Proceedings of IEEE TENCON, 1999.

3. D. H. Nam, S. K. Park, "A Smooth Playback Mechanism of Media Streams in Mobile Computing Environment," ITC-CSCC'98, 1998.
4. Ernst Biersack, Werner Geyer, "Synchronization Delivery and Play-out of Distributed Stored Multimedia Streams", Multimedia Systems , V.7 N.1 , 70–90, 1999
5. M. Woo, N. Woo,N. U. Qazi, and A. Ghafoor, "A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information," IEEE Network, Jan./Feb. 1994.
6. T. D. C. Little, and Arif Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services," IEEE Journal on selected Areas in Comm., Vol. 9, No.9, Dec. 1991.
7. W. Geyer, "Stream Synchronization in a Scalable Video Server Array," Master's thesis, Institute Eurecom, Sophia Antipolis, France, Sept., 1995.

Resource Reservation and Allocation Based on Direction Prediction for Handoff in Mobile Multimedia Networks

Jongchan Lee[1], Hongjin Kim[2], and Kuinam J. Kim[3]

[1] Radio Access Research Team ETRI, Korea
chan2000@etri.re.kr
[2] Dept. of Computer Information KyungWon College, Korea
[3] Dept. of Information Security Konggi Univ., Korea

Abstract. Future mobile communication systems can support not only voice but also multimedia applications such as data, image and video. It requires greater resources than voice-oriented mobile system. If handoff events are occurred during the transmission of multimedia, the efficient resource allocation and handoff procedures are necessary to maintain the same QoS of transmitted multimedia traffic because the QoS may be defected by some delay and information loss. This paper proposes a resource reservation and allocation scheme to accommodate multimedia traffic based on the direction estimation in mobile multimedia networks. This scheme estimates the position of mobiles based on a two step estimation comprised of sector estimation, zone estimation. With the position information, the moving direction is determined.

1 Introduction

The explosive growth of Internet access in parallel with the technological advances in mobile communications has motivated mobile computing and multimedia applications in wireless mobile networks. A Key characteristic of multimedia services is that they require different Quality of Service (QoS) guarantees. Due to the limitations of the radio spectrum, the wireless systems use microcellular architectures in order to provide a higher capacity. Because of small coverage area of micro-cells, network resources availability varies frequently as users move from one access point to another [1]. In order to deterministically guarantee QoS support for a mobile unit, the network must have prior exact knowledge of the mobile's mobility. Majority of the existing schemes to support mobility make a reservation for resources in adjacent cells. However these techniques cause a waste of resources since it is regardless of the direction of mobiles. Also, existing methods for predicting and reserving resources for future handoff calls do not seem to be suitable for mobile multimedia networks. The amount of resources required to successfully perform handoff may vary arbitrarily over a wide range in a mobile multimedia network. For example, data and video applications may adapt to different service quality levels and consequently

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 555–565, 2003.
© Springer-Verlag Berlin Heidelberg 2003

may accept different levels of resources in order to ensure a successful handoff [2][3][4]. In this paper, we consider a mobile network supporting diverse traffic characteristics of voice, data, and video applications. Since the connections can now differ in the amount of resources required to meet their QoS needs, the question is how should a base station dynamically adapt the amount of resources reserved for dealing with handoff requests. In this paper, the handoff requests for real-time connections are handled based on the direction prediction and the resource reservation scheme. The resources in the estimated adjacent cells should be reserved to guarantee the continuity of the real-time connections. If handoff requests are occurred during the transmission of multimedia traffic, the efficient resource allocation and handoff procedures are necessary to maintain the same QoS of transmitted multimedia traffic because the QoS may be defected by some delay and information loss. This paper proposes a handoff scheme to transmit multimedia traffic based on the resource reservation using direction estimation.

2 Direction Prediction Method

Figure 1 shows how our scheme divides a cell into many zones based on the signal strength, and then estimates the optimal zone stepwise where the mobile is located. This process is based on a two step location estimations which determines the mobile position by gradually reducing the area of the mobile position [5][6]. This scheme is implemented as an estimator into the base station. The estimator is started with a timer, and then the estimation is performed sequentially in two steps. The estimator first estimates the location sector in the sector estimation step, then estimates the location zone in the zone estimation step, and then finally estimates mobile's direction.

2.1 Sector Estimation

The sector estimation, the first step of the location estimation, is done in the following procedure.

A. All the neighboring base stations transmit pilot signals periodically.
B. The demodulator of the mobile measures PSSs of neighboring base stations.
C. The mobile sends PSMM (Pilot Strength Measurement Message) to the base station.
D. The estimator in the base station compares the received strengths of the pilot channels with each other and chooses the sector neighboring to the base station of the greatest signal strength as the sector at which the mobile locates.
E. The sector number is registered to the object information.

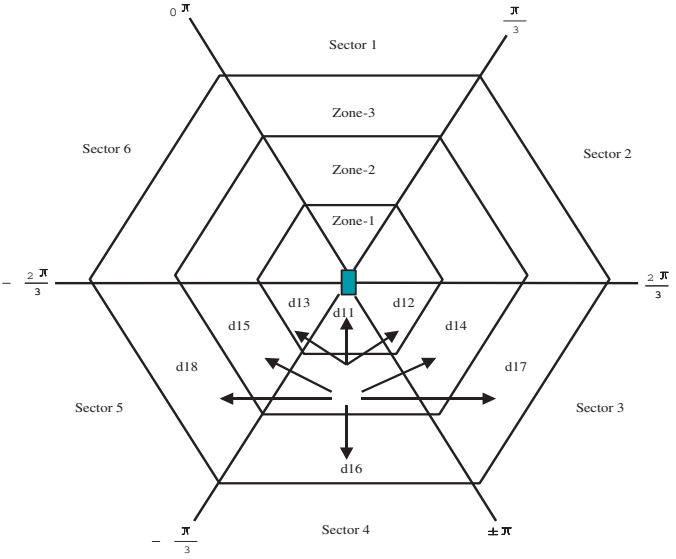


Fig. 1. Sector, zone and a mobile's moving direction

2.2 Zone Estimation

Each cell is divided into n zones with each zone classified by PSS. The following LOS algorithm summarizes the zone estimation procedure for LOS model.

1. Select each threshold considering PSS.
2. In order to map the signal strength onto the direction information, determine the distance function for each threshold with Equation(1).

$$\begin{aligned} P_A(d) &= k_1 - k_2(d) + u(t) \\ P_B(d) &= k_1 - k_2(D-d) + v(t) \end{aligned} \qquad (1)$$

   In Equation (1) D indicates the distance between two base stations, and d the distance between the base station A and the mobile. k_1 is proportional to the transmission power of the station and k has the offset value depending the radio propagation environment. Two random signals $u(t)$, $v(t)$ which indicate the power distributions of signals received at the distance d respectively from the station A and from the station B have i.i.d (identical independent distribution) with Gaussian distribution of $N(\mu(d), \sigma)$. The average value of the received signal at the specific location, $\mu(d)$, is determined by the path-loss component proportional to the distance and σ is assumed to be same. The changes in the LOS are depicted by k_2.

3. Classify zones using the distance function.
4. Assign the zone number and the PSS threshold to all divided zones.

Using the sector of blocks selected in the sector estimation, the zone estimation, the second step of the location estimation, estimates the zone of blocks at one of which the mobile locates. It is done in the following procedure.

A. The base station transmits the pilot signal periodically.
B. The demodulator of the mobile measures the signal strength of the pilot channel of the base station in which it is.
C. The mobile sends PSMM to the base station.
D. The estimator estimates the zone using the LOS algorithm.
E. The estimated zone number is registered to a zone object.

2.3 Direction Estimation

The estimator estimates a sector in the sector estimation step and a zone in the zone estimation step respectively, and then finally computes mobile's direction using the vector information between the estimated location and the previous one. In order to indicate the location of each zone within a cell, we use the vector data which is obtained by converting the rectangular coordinate of the zone to the polar coordinate with the origin of the base station [5]. Each vector has the information on a distance and an angle. The polar coordinate indicates the location by the distance from the origin to the mobile and the angle from the positive horizontal axis toward counter-clockwise. In our study we need the direction information to identify the sector relative to the base station and the relative position of a zone so we use the polar coordinates converted from the rectangular coordinates.

A mobile's direction is classified based on the movement into the upper zone (d11, d12 and d13), lower one (d16, d17 and d18) and the same one (d14 and d15), as shown in Figure 1. The moving radius of a mobile toward the upper from the lower zone becomes wider, while the moving radius of a mobile toward the lower zone becomes narrower. That is, the direction prediction for a mobile toward his BS is difficult more than mobiles toward his cell boundary. Therefore, for a mobile toward the upper zone it is efficient to increase the number of cells within his moving radius that is expected to be handed off, and to decrease the number of cells in case of a mobile toward the lower zone. Figure 2 shows the direction estimation algorithm based on the above conditions.

2.3.1 Resource Reservation for Low-Speed Mobiles

The moving radius and the moving pattern of a mobile has different characteristics according to the speed of the mobile. That is, a low-speed mobile (a pedestrian) has a smaller moving radius and a more complex moving pattern, while a high-speed mobile (a motor vehicle) has a larger radius and a simpler pattern. Using those characteristics, reservation variables such as the current position and the moving direction of a mobile are defined, and the neighboring cells that need to reserve resources are decided. In the case of low-speed mobiles,

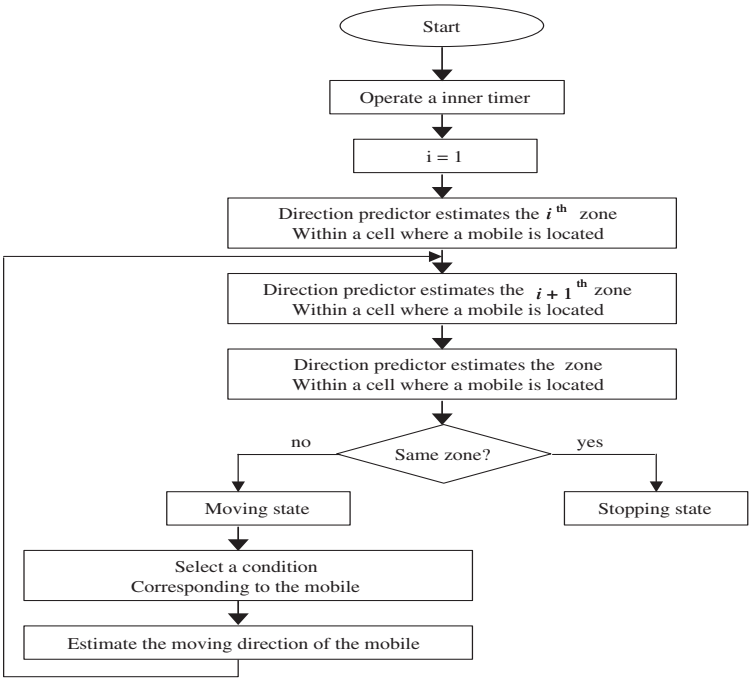


Fig. 2. Direction estimation algorithm

the result of the direction estimation can vary significantly due to two factors. First, the current position is considered. This is because the moving radius of low-speed mobiles is narrow, therefore, whether the handoff is done or not can be estimated according to his current position. Second, the moving direction is considered. The reason of using this factor is that the handoff attempt of the mobiles moving toward the inside of cell is decreased, while the handoff attempt of the mobiles moving toward the outside of cell is increased. The location zone is estimated in the sector and the zone estimation, and then based on the location of the estimated zone and the predicted direction, cells in the neighborhood are ranked according to the likelihood that the mobile will move into these cells. Cells needed to reserve the resource is decided by the sector estimation, and the resource reservation for the cells is done using the current position estimated by the zone estimation. Mobiles moving toward the upper zone need not reserve the resource. The resources for mobiles moving toward the lower zone are reserved only in case their estimated position is zone-3. Resource reservation conditions based on the reservation variable for a low-speed mobile is as follows.

– Condition 1: if the current position is zone-1, the reservation is not made regardless of its moving direction.

- Condition 2: if the current position is zone-2 and its movement is done from zone-1, the reservation is not made.

- Condition 3: if the current position is zone-2 and its movement is done from zone-3, the reservation is made to the maximum two cells.

- Condition 4: if the current position is zone-3 and its movement is done from zone-2, the reservation is made to the maximum one cell.

2.3.2 Resource Reservation for High-Speed Mobiles

The reservation variable for fast-speed mobiles is the moving direction. If the mobiles move fast, the feasibility of performing a handoff is expected to be higher; therefore the reservation is needed regardless of their current position within cell. Since the mobility of the fast-speed mobiles has a varying randomness, a cluster of cells that is reflective of the user mobility is needed to reserve the resources. Resource reservation conditions based on the reservation variable for a high-speed mobile is as follows.

- Condition 1: If the mobile moves from zone-1 to zone-2 within the same sector, the resources are reserved for three cells that is reflective of the moving direction.
- Condition 2: If the mobile moves from zone-2 to zone-3 within the same sector, the resources are reserved for one cell that is reflective of the moving direction.
- Condition 3: If the mobile moves from zone-3 to zone-2 within the same sector, the resources are reserved for five cells that is reflective of the moving direction.
- Condition 4: If the mobile moves from zone-1 to zone-2 within the same sector, the resources are reserved for three cells that is reflective of the moving direction.
- Condition 5: If the mobile moves from zone-1 to zone-2 within the other sector, the resources are reserved for three cells that is reflective of the moving direction.
- Condition 6: If the mobile moves from zone-2 to zone-3 within the other sector, the resources are reserved for one cell that is reflective of the moving direction.
- Condition 7: If the mobile moves from zone-3 to zone-2 within the other sector, the resources are reserved for five cells that is reflective of the moving direction.
- Condition 8: If the mobile moves from zone-2 to zone-1 within the other sector, the resources are reserved for five cells that is reflective of the moving direction

3 Direction Prediction Based Resource Reservation and Allocation

3.1 Resource Reservation and Allocation Structure

A real-time mobile performs resource reservation for hand-off, and a set of the reserved resources can be occupied temporally by non-real-time mobiles within the target cell. On the order hand, a non-real-time mobile doesn't perform resource reservation for handoff, and its resource allocation request is buffered in the waiting queue of the target BS during handoff duration time and is given the priority based on the service demand time. If the reserved set is returned because the corresponding real-time mobile is handed off, the priority of the request becomes the lowest lank. This strategy is explained in Figure 3.

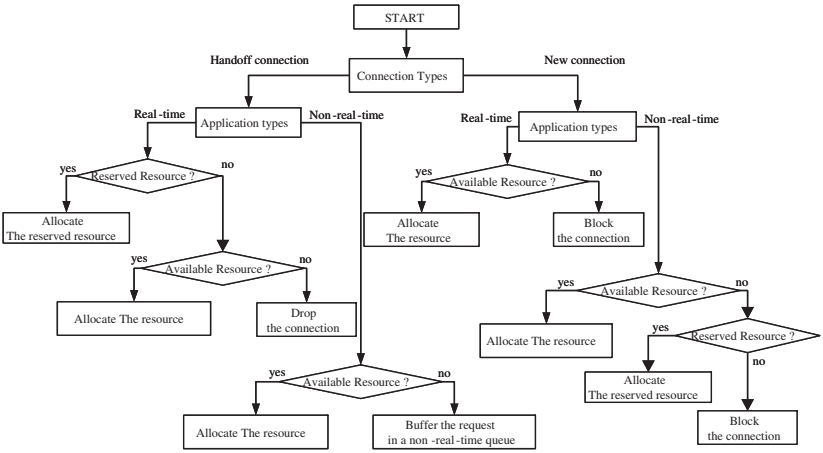


Fig. 3. Admission control for multimedia connections

3.2 Resource Reservation Procedure

The base station reserves only the resources corresponding to the minimum transmission rate to the mobile. Based on the location and the direction of the mobile within a cell, the resource reservation is performed with the following order: unnecessary state, not necessary state, necessary state, and positively necessary state. If the reservation variable for the mobile is changed, the reservation is canceled and the resources have to be released with the reverse order and returned to the fool of available resource. The set of the reserved resources have its priorities depending on whether it can be allocated to new connections or not: a real-time handoff connection (priority 1), a non-real-time handoff connection (priority 2) and a non-real-time new connection (priority 3).

– *Unnecessary State*
The resource reservation needs not be performed.
This state corresponds to the resource reservation condition 1 and 2 for low-speed mobiles.

– *Not Necessary State*
A set of the reserved resources corresponds to priority 3.
If any resources are available in each of the estimated cells, the resources are then reserved for each of the mobiles. A set of the reserved resources can be occupied by the new connections if enough resources are not available for a new connection in each of the estimated cells.
If there are any resources available to support the reservation in the estimated cell, and a moving connection competes with a new connection for the resources, the resources are occupied with the following order: a real-time handoff connection, a real-time new connection, a non-real-time handoff connection, and a non-real-time new connection.
If no resources are available, the reservation is not done.
This state corresponds to the resource reservation condition 3, 7 and 8 for fast-speed mobiles.

– *Necessary State*
A set of the reserved resources corresponds to priority 1.
If there is no enough resource available to accommodate a new connection, a set of the reserved resources for real-time handoff connections can be occupied by non-real-time new connections.
If there are resources available to support the reservation in the estimated cell, and a moving connection competes with a new connection for the resources, the order of occupying the resources is the same as Not Necessary State.
If no resources are available for the reservation in the estimate cell, the shared part resources are allocated and reserved for a real-time connection.
This state corresponds to the resource reservation condition 4 for low-speed mobiles.
This state corresponds to the resource reservation condition 1, 4, 5 and 8 for fast-speed mobiles.

– *Positively Necessary State*
The reserved resources correspond to priority 1.
New connections cannot occupy the reserved resources.
In case of a moving connection competes with a new connection for resources in the estimated cell, the resources are occupied with the following order: a real-time handoff connection, a non-real-handoff connection, a real-time new connection and a non-real-time new connection.
If no resources are available for the reservation in the estimate cell, the shared part resources can be allocated and reserved for both real-time connections and non- real-time connections.

This state corresponds to the resource reservation condition 4 for low-speed mobiles.
This state corresponds to the resource reservation condition 2 and 6 for fast-speed mobiles.

4 Simulation Model and Result

The proposed scheme is compared with two different methods to evaluate the performance.

Method 1: there is resource reservation. The resources are reserved exclusively for handoff connections in each cell. The remaining resources can be equally shared among handoff and new connections. This method is called Fixed_Res.

Method 2: the resources are reserved dynamically based on the current connections in the neighboring cells. This method is called Dynamic_Res.

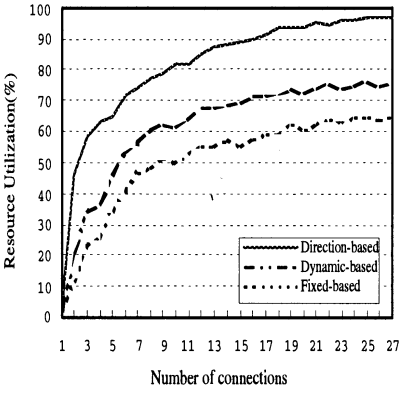


Fig. 4. Comparison of resource utilization

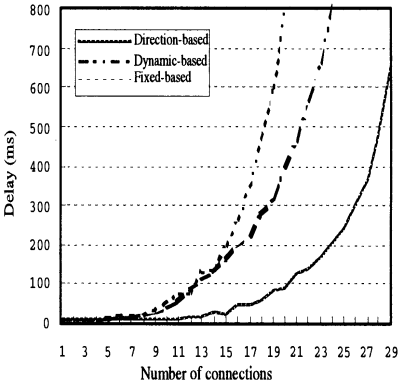


Fig. 5. Comparison of transmission delay

The simulation model composed of a single cell, which will keep contact with its six neighboring cells. Each cell contains a base station, which is responsible for the connection setup and tear-down of new applications and to serve handoff applications. We consider the following simulation parameters regarding the received signal strength. The mean signal attenuation by the path-loss is proportional to 3.5 times the propagation distance, and the shadowing has a log-normal distribution with a standard deviation of $\sigma = 6dB$. A value of the received signal strength less than -16 dB is regarded as an error, which is therefore excluded from the calculation. Figure 4 shows the results for an average percentage of resource utilization as a function of connections arrival with priority to handoff connections over new connections. The resource utilization for

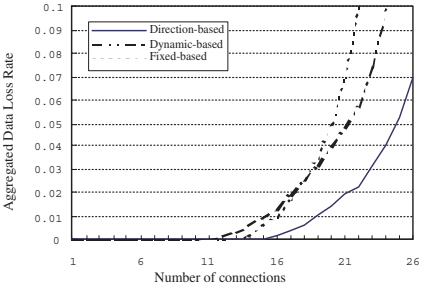


Fig. 6 Comparison of aggregated data loss rate

the Direction-based is increased up to 25 arrival, than that for Fixed and Dynamic. This improvement may be caused since the Direction-based allocates the reserved resources not only real-time handoff connections but also non-real-time new connections. In Figure 5, the comparison of transmission delay of the three schemes is plotted against the number of connections. From the figure, we can see the performance of the direction-based is up to 2.5 times better than that of the conventional schemes. This is because non-real-time connections can adaptively occupy the reserved resources for real-time connections, and we can prevent the performance degradation due to queueing of non-real-time connections.

Figure 6 shows the aggregated data loss rate. It is observed that as the number of connections increase, direction-based provides a noticeable improvement over the conventional schemes for real-time connections, while slightly degrading the performance for the non-real-time connections.

5 Conclusion

This main paper is to address the problem of guaranteeing an acceptable level of QoS requirements for mobile users as they move from one location to another. This is achieved through reservation variables such as the current location and the moving direction that is presented with a set of attributes that describes the user mobility. In this scheme, mobiles are classified according to their reservation variables. Based on reservation variables a scheme that provides predictive QoS guarantees in mobile multimedia networks is proposed. The proposed scheme shows a great improvement of the resource utilization, the delay and the data loss. It is because our resource reservation scheme is more adaptive than existing resource reservation schemes. In our scheme, resources are classified as ones having priority to the new calls and ones having priority to the handoff calls based on reservation variable. We improve the dropping rate for the handoff connections by dynamically adjusting the amount of the reserved resources according to the amount of occupied resources. The determination of the optimal direction should be studied consecutively. Also further researches are required on their implementation and applications to the handoff and resource allocation strategies.

References

1. P. Agrawal, D. K. Anvekar, and B. Narendran, "Channel management policies for handovers in cellular networks," Bell Labs Tech. J., vol. 1, no. 2, pp. 97–10, 1996.
2. M. Naghshineh and M. Schwartz, "Distributed call admission control in mobile/wireless networks," IEEE J. Select. Areas Commun., vol. 14, pp. 711–717, May 1996.
3. O. T. W. Yu and V. C. M. Leung, "Adaptive resource allocation for prioritized call admission over an ATM-based wireless PCN," IEEE J. Select. Areas Commun., vol. 15, pp. 1208–1225, Sept. 1997.
4. C. H. Yoon and C. K. Un, "Performance of personal portable radio telephone systems with and without guard channels," IEEE J. Select. Areas Commun., vol. 11, pp. 911–917, Aug. 1993.
5. J. C. Lee and Y. S. Mun, "Mobile Location Estimation Scheme," SK telecommunications Review, Vol. 9, No. 6, pp. 968–983, Dec. 1999.
6. J. C. Lee, B. H. Ryu and J. H. Ahn, "Estimating Position of Mobiles by Multi-Criteria Decision Making," ETRI J., vol. 24, no. 4, pp. 323–327, Aug. 2002.

Petri-Net Model of Multimedia Synchronization in Mobile Environment

Keun-Wang Lee[1], Jong-Hee Lee[2], and Hae-Seok Oh[2]

[1] Dept. of Multimedia Science, Chungwoon Univ.,Korea
kwlee@cwunet.ac.kr
[2] School of Computing Soongsil Univ., Korea
multistar@freechal.com
oh@computing.soongsil.ac.kr

Abstract. As demand for the application in multimedia networks is increasing rapidly, it is important to provide multimedia services in Mobile Environments (ME). Obtaining to multimedia services which satisfy synchronization constraints in ME and improving the delay time and Quality of Service(QoS) between media streams to be presented, new specification model has to be proposed. In this paper we propose Mobile Petri-net Model(MPM) as a new specification model including the QoS parameter that minimizes the transmission delay time. The proposed model has higher the guarantee of QoS such as the loss rate and the playout rate than it of the previous work.

1 Introduction

The interest in multimedia services coupled with a growing high-bandwidth structure has been increased in the area of wired communications. Recently, with the advancement in wireless communication systems and portable computing technologies, demand for mobile multimedia services is increasing in the area of wireless communications, such as data transmission, video transmission, www access, and so on[1][2][3]. Mobile multimedia service is complex concept of wireless extention of wired multimedia service and mobile specific multimedia service. That is, mobile multimedia service is to add user mobility to multimedia services in wired networks[4]. The synchronization in wired networks has been continuously studied, while scarcely in wireless networks. In wireless networks, a Mobile Host (MH) has small memory, and a Base Station (BS) supports the limited resources than the sites in wired networks. Moreover, the mobile communication system has low power, and must provide high-quality access with MHs. Consequently, the synchronization in wireless networks is complex than that in wired networks. In particular, because of the small memory and the limited resources, a MH may suffer from either the buffer underfloor the buffer overflow. Therefore, an adaptive synchronization method is needed for mobile multimedia networks. The previous works have been done in describing the synchronization model for multimedia applications[5],[6][7][8][9][10]. Among those, Petri-net

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 566–575, 2003.
© Springer-Verlag Berlin Heidelberg 2003

based specification models provide a good method to specify temporal relationships. The integration of various media and the definition of QoS requirements should be supported and easily described. However, previous extended Petri-net based models like as Object Composition Petri-Net (OCPN) and Real Time Synchronization Model (RTSM) have some constraints in modeling the QoS parameters of both intramedia and intermedia. So, we propose MPM model which guarantees the QoS in ME.

2 Related Work

At the time of creation of multimedia information, a user needs a model to specify temporal constraints among various data which must be observed at the time of playout[3]. Such specification models are OCPN and RTSM which are a timed Petri-net. OCPN has been proved to have the ability to specify any arbitrary temporal relationship among media and it may be applied to both stored data applications and live applications[3][5]. The OCPN model weakly describes synchronous temporal relationships between media at an object level. A multimedia application which has video, audio and text object can be expressed by OCPN modeling technique as in Figure 1. Generally, audio data is very jitter sensitive and cannot tolerate random delays between frame segments; hence, delay between audio frames may result in unrecognizable audio quality. In contrast, some extent of random delay between video frames may be acceptable in video data. Therefore a late transmission of video frames does not influence the QoS.

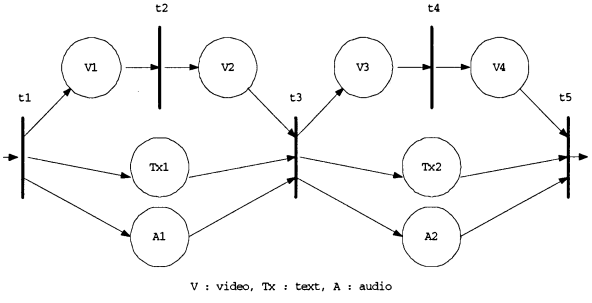


Fig. 1. An example OCPN model.

When taking the real-time issue of multimedia and the random delay of packet/cell networks into consideration, OCPN and other Petri-net based models are not sufficient to deal with the late transmission of packets. Here, we define late transmission of a packet to be that the packet fails to reach its destination in time and should be dropped. We explain the insufficiency of OCPN by the example shown in Figure 1. A delay of video object in OCPN will give an impact on playing out anaudio object. Such a delay of audio object in a real-time

application will cause a significant degradation of its QoS. RTSM is introduced to guarantee the QoS of real-time applications on the delay of intermedia. A jitter-sensitive audio object is defined as key medium and represented as enforced place [6]. However it is possible that a real-time constraint is exceeded due to the delay of key medium itself. If the text object is an one of the key medium in Figure 1, it is assigned as an enforced place just as an audio object. Once one of enforced place is unblocked, each transition of RTSM is fired. When an audio object is unblocked, the transition is fired, even if text object is still activated. Therefore the text object is not guaranteed to display its content. When the transition is fired in RTSM, the remainder part of other medium is dropped. That is, when taking a random delay into account, it is not sufficient to deal with late transmission of packets, and it is also insufficient in modelling the QoS parameters.

3 MPM

The explosive growth of Internet access in parallel with the technological advances has motivated mobile computing and multimedia applications in mobile networks. We assume that Multimedia Servers (MSs) over Internet act as repositories for multimedia applications orchestrated according to the synchronization model, such as OCPN. A MS searches the requested data from its databases and transmits it to the BS. Figure 2 shows the mobile multimedia system structure.

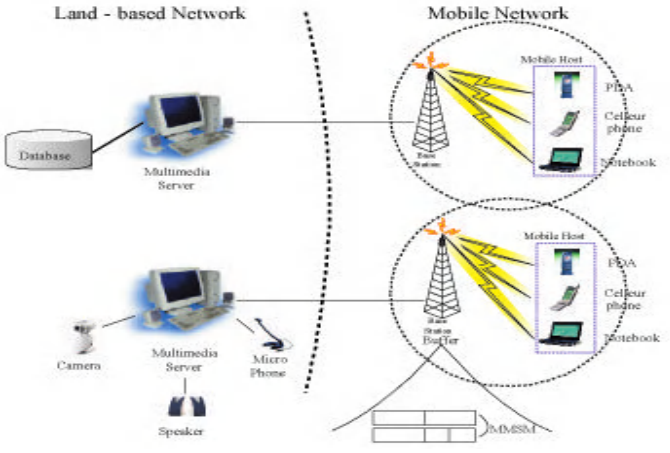


Fig. 2. Mobile multimedia system structure.

Multimedia data consists of many multimedia objects, a unit of synchronization, which are stamped with the current local time to allow the BS to calculate a round trip delay, jitter and inter-arrival time, during the MS transmits multimedia objects. The multimedia objects need to be buffered at the BS for carrying

out synchronization as interface between the Internet and mobile networks. Mobile multimedia network consists of three parts: MS, BS and MH. In addition to the conventional functions of the BS such as frequency resource management, error control and so on, the BSs in the mobile multimedia network have the additionafunction such as synchronization, rate matching, etc. The MH must access MSs over Internet through the BS. As shown in Figure 2, multimedia data is transmitted from the MS to the BS. After buffering the multimedia data in order to smooth an inter-packet jitter delay which occurs over Internet, the BS transmits the synchronized data to the MH.

3.1 Definition of MPM

The definition of MPM to specify Petri-net in any BS is the following.
The MPM specified by the tuple [*P, T, K, A, DIFF, R, D, J, Re, M*]
where
$P = p_1,\ p_2,\ \ldots,\ p_n$ *; Regular places(single circles).*
$T = t_1,\ T_2,\ \ldots,\ t_m$ *; Transitions.*
$K = k_1,\ k_2,\ \ldots, k_i$ *; Key places.*
$X = P \cup K$; *All places.*
$A = (X) \cup (T) \rightarrow I, I, = 1, 2, 3, \ldots$; *Directedarcs.*
$DIFF : X \rightarrow$ *Realnumber*; *Difference between maximum duration time and actual duration time.*
$R : X \rightarrow$ *Realnumber ; Relative duration time.*
$D : X \rightarrow$ *Realnumber ; Maximum duration time of places.*
$J : X \rightarrow$ *Realnumber ; Maximum delay jitter.*
$Re : X \rightarrow r_1, r_2, \ldots, r_k$ *; Type of media.*
$M : X \rightarrow I, I = 0,\ 1,\ 2$ *; State of places.*
Each place may be in one of the following states:

0 : no token.
1 : token is blocked Cross in the place.
2 : token is unblocked Dot in the place.

Places areused to represent a medium unit and their corresponding actions. The Place can have tokens. A Place without token means that it is currently inactive. A place with token is active and could be in one of the two states: blocked or unblocked. When a transition is fired and then token is added into a place, the action of the place is executed, and the token is blocked before finishing the action. The token is unblocked after finishing. Each place has some parameter that determines its relative importance compared with other mediums. Two factors should be considered in deciding the time medium with absolute time and the multiple key media parameter. First, the importance of the medium and second, the jitter-sensitivity of the medium.

3.2 Firing Rule of MPM

Firing rules of MPM are described as follows:
1) The firing occurs immediately if the end of time medium is done.

2) Transition t_i fires immediately when the arrived key medium places contain an unblocked token.
3) Upon firing, a set of backtracking rules is exercised to remove tokens from their input places.
4) Transition t_i removes a token from each of its input places and adds a token to each of its output places.
5) After receiving a token, a place p_j remains in the active state for the interval specified by the duration j. During this interval, the token is blocked. In MPM, there are two kinds of places as regular places and key places. To differentiate a key place from a regular place, a double circle is drawn for key place as shown in Figure 3 where transition t_4 is fed by two regular places, V_3 and Tx_1 and one key place, Au_1.

4 MPM Including the QoS Parameter

Each place p_i has time-related four parameters $\tau_{diff}, \tau_r, \tau_d$, and τ_j as well as key medium in order to handle QoS parameters for intermedia synchronization. Figure 3 presents the MPM with QoS parameter.

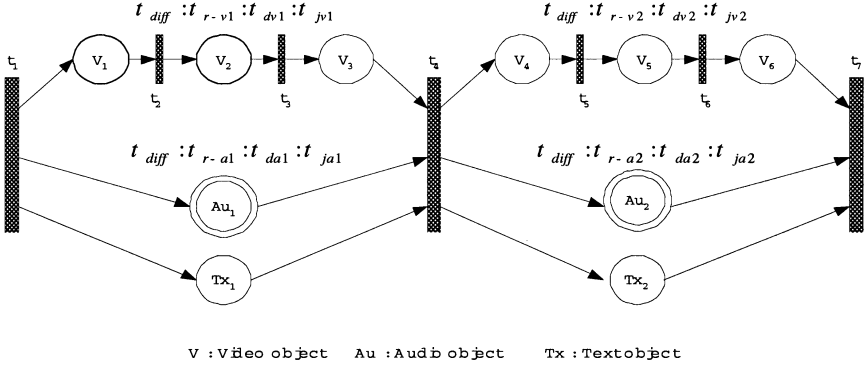


Fig. 3. MPM with the QoS Parameter.

Intermedia skew value 0 means that the two media streaming is completely synchronized. For instance, it can be considered as the synchronized state if it is within 80ms between audio and video streams. If skew value 80ms is applied to key medium of audio, the maximum delay jitter τ_j will be destroyed, which will result in degrading the audio quality. Because the maximum delay jitter of audio within media is 10ms, waiting longer than 10ms could be a factor that destroys the audio quality. Therefore, τ_j is compensated by means of applying the maximum delay jitter τ_j. Maximum delay jitter τ_j represents maximum delay jitter of the place p_i. The delay is a interval between medium i and medium $i + 1$, and it is used to obtain the better quality of service by additionally

playing out as much as the time of maximum delay jitter. Audio medium τ_{diff} represents a time difference between duration time and relative duration time. Video medium τ_{diff} is a result of jitter-compensatory time algorithm and it is obtained from comparing τ_{diff} with τ_{ja1} of audio medium. Applying maximum jitter delay not only improves QoS but also has an effect on compensating a time of playing out a place except key medium. A smoothing buffer 125ms is used at the BS to avoid a jitter delay in applications. A delay of a smoothing buffer is acceptable to video phone, video conference, tele-teaching and real-time application. The algorithm that decides the relative duration time is the followings.

Procedure Relative Duration Time

Begin { τ_{ma} is arrival time of media };

If $\tau_{ma} <= \tau_b$ Then { τ_b is maximum delay time };

$\tau_r := \tau_b$; {τ_r represents a relative duration time to playout an arrived media };

Else

Begin

$P_{ma} := P_{ms} \times \tau_b | \tau_{ma}$; /* P_{ma} is maximum size for media to playout during a duration time. P_{ms} is arrived size during a smoothing buffer delay */;

$P_r := P_{ma}/P_{ms}$; /*P_r represents a playout rate of P_{ma}/P_{ms}*/;

$\tau_r := P_r \times \tau_d$; /*$\tau_d$ is the duration time for which a playout or a display of a media */;

End

End

Jitter-compensatory time algorithm is the following.

Procedure Jitter Compensatory Time

Begin /*τ_{r-a} is relative duration time of audio */;

τ_{r-a}:=relative_duration_time (Au);

$\tau_{diff} := \tau_d - \tau_{r-a}$; /* τ_{diff} is the time difference between a duration time and a relative duration time */;

While media Then

Begin

τ_{r-m} :=relative_duration_time(m) ; /* τ_{r-m} is relative duration time of media excepts audio */;

IF $\tau_{diff} <= \tau_j$ Then /* τ_j is the acceptable maximum delay jitter time between each medium */;

Begin

wait (τ_{diff}) ;

$\tau_{r-mp} := \tau_{r-a} + \tau_{diff}$; /* τ_{r-mp} is playout time of media */;

End

Else

Begin

wait (τ_j) ;

$\tau_{r-mp} := \tau_{r-a} + \tau_j$;

End

End

End.

Transition t_4 is fired only by A_1 with key medium among input places in Figure 3. Let's assume that it takes 135ms for audio object and video object to reach the destination transit. A relative duration time of audio object will be 115ms according to a relative duration time algorithm and t_{diff} will be 10ms by a jitter-compensatory time algorithm, which means that audio object will wait as much as 10ms because it does not exceed the maximum delay jitter of 10ms. Video object t_{diff} will take a smaller one between t_{diff} and t_{ja1} of audio object. The relative duration time of video object will be 115ms by the relative duration time algorithm. 10ms from an audio object is added to 115ms of relative duration time of video object. Video object can be compensated in cases that

the total time added with t_{diff} of video object does not exceed 125ms. That is, video object can be compensated as much as 10ms because it does not exceed 125ms. If jitter is not considered, the delay of audio and video object gives a bad impact on playing a next frame, while a proposed jitter-considerable scheme results in playing a next frame efficiently.

5 Performance Evaluation

Media format used in the simulation is in the following. The 1Kbytes audio data is encoded by the PCM encoding scheme, and resolution of the video frame is 120 × 120. Note that the number of video places corresponding to one audio place is random due to the following reason. At the BS, the application gets one audio packet from the audio device every 125ms, and it gets one, two or third video frames from the video frame grabber, which is determined by the run-time processing overhead of the operating system. In the paper, the simulation environment is set to the MH in ME. The sample data to be used in the simulation are generated by the Poisson distribution and the network jitter delay is equally applied to all media. The number of media is limited to two types for comparing the performance. Notice that the delay value generated by the normal distribution is forced to be within the range [0.5*mean, 4*mean] in the program to make the delay values more realistic. The number of transitions is set to 200 and the maximum delay jitter is 10ms. On the purpose to verify the algorithm, the previons OCPN and RTSM are also evaluated in the same simulation environment. The playout rate of the proposed delay jitter scheme based on the maximum delay jitter time is compared to the existing methods. The efficiency of the proposed scheme is observed by being comparing with the palyout rate of the different models, in case of the normal and abnormal arrival of audio streams to BS, respectively. It is assumed that the average delay is 100ms and the variance is 20ms in case of audio streams with delay.Figure 4 shows the comparison of the palyout time of RTSM and MPM model as functions of the number of transactions in case of audio streams without delay. In Figure 5, the result for the palyout rate of RTSM and MPM model is plotted against the number of simulations, in which the playout rate of MPM model is increased to about 8%, as compared to the existing method.

Figure 6 shows the simulation results of the three models in terms of the video duration time. The duration time of OCPN model dramatically decreased however, MPM model shows the better performance, compared with RTSM model.

In Figure 7, the result for the palyout rate of the three models is plotted against the number of simulation, Result demonstrate that the playout rate for audio streams in MPM model is increased to about 8 % and 6.3 % in case of without and with delay, respectively. The result here seem to indicate that MPM model increases the playout rate and decreases the loss rate, in terms of audio and video, than OCPN and RTSM model.

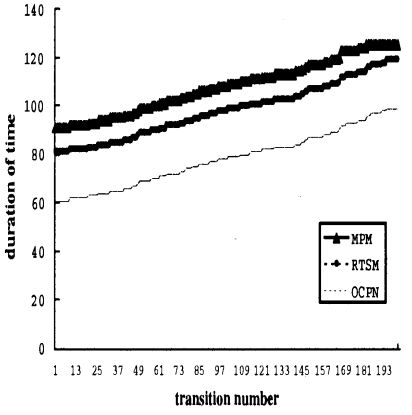


Fig. 4. Duration of time for audio without delay

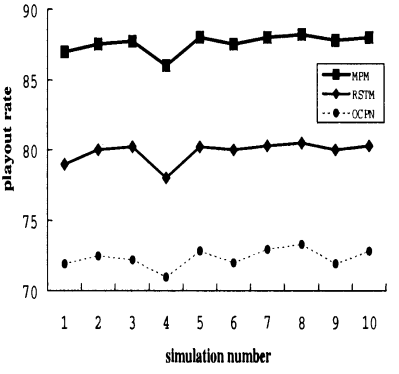


Fig. 5. Playout rate for audio without delay

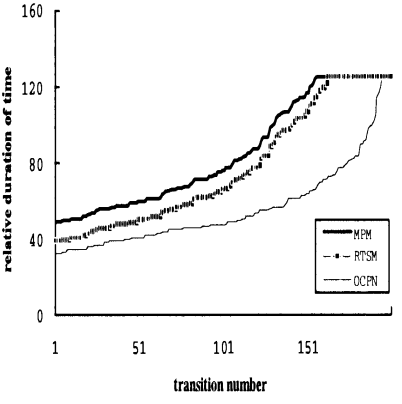


Fig. 6. Comparison of playout time with delay

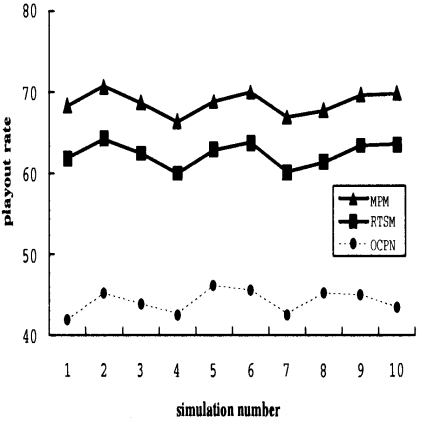


Fig. 7. Comparison of playout rate with delay

6 Conclusions

In this paper we have proposed a new specification model to describe the QoS parameter and the delay time between media data in ME. The proposed MPM has a distinct property which allows QoS parameter. This is very important since most applications represent without QoS in ME.The contributions of this paper are listed in the following. 1) An MPM to describe the QoS parameter for intermedia synchronization in ME is proposed. 2) The concept of multiple key media in MPM for synchronization control is suggested. 3) The concept of time medium associated with multiple key media to deal with late transmis-

sions of multiple key media is suggested. 4) The proposed method meets the requirements of synchronization between media data by handling dynamically the adaptive waiting time resulted from variations of delay time and the interval adjustment using maximum delay jitter time at BS in ME. The proposed MPM model supplies flexible description of the temporal relationships among various media. This model can provide modeling specifications from the live applications to the pre-orchestrated applications in ME. MPM model adaptively manages the waiting time of smoothing buffer, which leads to minimize the gap from the variation of delay time and it results in naturally playout to MH in ME. Also it is suitable to the system which requires the guarantee of high QoS and the model improves QoS such as decrease of loss rate and increase of playout rate in ME.

References

1. D. H. Nam and S. K. Park.: Adaptive Multimedia Stream Presentation in Mobile Computing Environment, Proceedings of IEEE TENCON, Cheju Korea.(1999) 966–969
2. A. Boukerche, S. Hong and T. Jacob.: MoSync: A Synchronization Scheme for Cellular Wireless and Mobile Multimedia Systems, Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems IEEE. (2001) 89–96
3. M. Woo, N. Prabhu and A. Grafoor.: Dynamic Resource Allocation for Multimedia Services in Mobile Communication Environments, IEEE J. selected Areas in Communications, Vol.13, No.5, June. (1995) 913–922
4. D. H. Nam, S. K. Park.: A Smooth Playback Mechanism of Media Streams in Mobile Computing Environment, ITC-CSCC'98. (1998)
5. P. W. Jardetzky, and C. J. Sreenan, and R. M. Needham.: Storage and synchronization for distributed continuous media, Multimedia Systems/Springer-Verlag. (1995)
6. C.-C. Yang and J.-H. Huang.: A Multimedia Synchronization Model and Its Implementation in Transport Protocols, IEEE J. selected Areas in Communications, Vol.14, No.1, Jan. (1996)
7. R. Steinmetz.: Human Perception of Jitter and Media Synchronization, IEEE J. selected Areas in Communications, Vol.14, No.1, Jan. (1996)
8. D. L. Stone, and K. Jeffay.: An empitical study of delay jitter management policies, Multimedia Systems/Springer-Verlag. (1995)
9. G. Blakowski and R. Steinmetz.: A Media Synchronization Survey: Reference Model, Specification, and Case Studies, IEEE J. selected Areas in Communications, Vol.14, No.1, Jan. (1996)
10. M. J. Perez-Luque and T. D. C. Little.: A Temporal Reference Framework for Multimedia Synchronization, IEEE J. selected Areas in Communications, Vol.14, No.1, Jan. (1996)

New Mobility Scheme for Reducing Location Traffic in Mobile Networks

Il-Sun Hwang[1], Myungsun Lee[1], Ki-sung Yoo[1], and Jin-wook Chung[2]

[1] R&D Network Management KISTI, Korea
his@kisti.re.kr
[2] Dept. of Computer Eng. SungKyunKwan Univ., Korea

Abstract. A major problem in such a mobile networks is how to locate mobile hosts. Two major operations,the location registration and the call tracking, are involved in managing a mobile host's location. The previous works can only minimize the traffic cost of one of these two operations. In this paper we propose new strategy that minimize the costs of both operations. Our performance analysis proves that the proposed strategy is superior to the previous methods.

1 Introduction

A great difference between wireless network environment and traditional wired communication environment us that it allows users to migrate anywhere and to retrieve all kinds of data they want anytime in mobile computing environment. To effectively monitor the movement of each mobile client, a large geographical region is partitioned into small Registration Areas (RA). Each RA has a Mobile Switching Center (MSC)which serves as the local processing center of the RA. The profiles of mobile clients inside a RA are kept, in the MSC's Visitor Location Register (VLR). On top of several MSC/VLRs is a Local signaling Transfer Point (LSTP) and on top of several LSTPs again is a Remote Signaling Transfer Point (RSTP). In this way, the whole system forms a hierarchy of station. The LSTP and the RSTP are routers for handling message transfer between stations. For one RSTP there is a Home Location Register (HLR). Each mobile client must register in a HLR.Communication cost depends on how high the message has to go up the hierarchy, as it indicates the distance and routers it has to go through.

In spite of many advantages available in wireless communication. It is not without difficulties to realize such systems. The first problem is how to locate a mobile user in a wireless environment. The IS-41 standard strategy is most often referred in resolving this problem. IS-41 used in the United States and GMS [6] used in Europe are examples of this strategy. Many papers in the literature have demonstrated that the IS-41 strategy does not perform well. This is mainly because whenever a mobile client moves. The VLR of a RA which detected the arrival of the client always reports to the HLR about the client's new location. Several other location management strategies have been proposed to improve the performance of the IS-41 strategy. Among them, the Forwarding strategy

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 576–585, 2003.
© Springer-Verlag Berlin Heidelberg 2003

[4][5] the Local Anchor (LA) strategy [8], and the Caching strategy [11] are representatives of the old VLR to the new VLR. Update of the client's location to the HLR's database is not always needed to minimize communications to the HLR. To locate a callee however, some extra time is required to follow the forwarding link(s) to locate the client. When the number of the forwarding links is high, the locating cost becomes significant. Our strategy, termed the Dynamic Cache & Carry (DCC) strategy, tries to further improve the performance of these previous methods by minimizing the locating cost. The key idea is for each mobile client (1) we define the VLRs that have been linked to by the same LA as the overseen VLRs of this LA, (2) a link from the overseen VLR to its LA is cached in the VLR, and (3) we allow multiple such LAs to be linked together by using forwarding links. In order to computer the performance improvement over the previous methods can be achieved.

2 The IS-41 Strategy

According to the IS-41 strategy, the HLR always knows exactly the ID of the serving VLR of a mobile terminal. We outline the major steps of the IS-41 location registration scheme as follows [6]:

1. The mobile terminal sends a registrationrequest(REGREQ) message to the new VLR.
2. The new VLR checks whether the terminalis already registered. If not, it sends a registration notification (REGNOT) message to the HLR
3. The HLR sends a registration cancellation (REGCANC) message to the old VLR.

The old VLR deletes the information of the terminal and the IS-41 call tracking scheme is outlined as follows:

1. The VLR of caller is queried for the information of callee. If the callee is registered to the VLR,the SEARCH process is over and the call is established. If not,the VLR sends a location Request (LOCREQ) message to the HLR.
2. The HLR finds out to which VLR the callee is registered, and sends a routing request (ROUTREQ) message to the VLR serving the callee. The VLR finds out the location information of the callee.
3. The serving MSC assigns a temporary local directory numbers(TLDN) and returns the digits to the VLR which sends it to the HLR.
4. The HLR sends the TLDN to the MSC of the caller.
5. The MSC of the caller establishes a call by using the TLDN to the MSC of the callee. Among the above 5 steps, the SEARCH process is composed of step1 and step2.

3 DCC Strategy

The previous methods for improving the IS-41 mechanism were all designed to reduce the locating cost.

3.1 Location Registration

The basic idea of the DCC strategy is that the new VLR determines whether reports to the LA or the HLR the mobile client's new location when a mobile client comes into its territory. Figure 1 tells how the new VLR reports to the HLR about the mobile client's new location. How the new VLR informs the HLR is new in this strategy. How the new VLR informs the HLR is new in this strategy. The algorithm is as follows.

1. When the new VLR learns that a mobile client is coming into its territory, it will determine whether requesting a location update to the LA or the HLR. If the new location of the mobile client is outside the area that the LSTP of the LA controls (before the mobile client moves to the new VLR), jump to the Step 7 (the new VLR will request a location update to the HLR). Otherwise, go on to the next step (the new VLR informs the LA of location Update).
2. The old VLR replies an acknowledgement to the new VLR.
3. The old VLR informs the LA that the mobile client has moved to the new VLR. Also the old VLR update its own MCL table by replacing the mobile client's Type value with "Visited VLR" and the Pointer value with the LA's ;location.
4. The LA replies a message to the old VLR, and updates its own MCL table. The Type value of the mobile client is not changed. The Pointer value is modified to the new VLR's location.
5. End of the Location Registration.
6. Goto Step 13.
7. The new VLR informs the HLR of the arrival of the mobile client.
8. The HLR returns an acknowledgement to the new VLR. The MCL table of the new VLR is inserted a record. If the mobile client has never visited this VLR, whose three attribute values are respectively $\mathrm{MC}_i d$, "Latest LA", and NULL. Otherwise. Only updates the Type and the Pointer values of its record.
9. The HLR updates its data about the mobile client's new location and informs the old LA about the movement of the mobile client.
10. The mobile client's record in the MCL table of the old LA is updated. Its second and third attribute values are modified to the "Visited LA" and the new LA's location. Respectively. The old LA also sends an acknowledgement message to the HLR.
11. The old LA informs the old VLR of location change of the mobile client.
12. The mobile client's record in the old VLR's MCL table is updated. Its second and third attribute values changed to "Visited VLR" and the location of the old LA, respectively. The old VLR also sends an acknowledgement to the old LA.
13. End of the Location Registration.

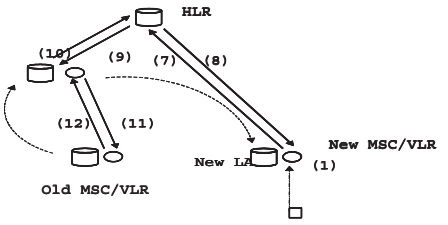


Fig. 1. The case which the new VLR informs the HLR

3.2 Call Tracking

Figure 2 are a example that could occur while locating the callee. The details of the operation is as follows.

1. When a VLR receives a request of locating a callee, it first checks whether its MCL table has the callee's record. If yes, then sends the locating requite to the LA stated in this record. Otherwise, jump to Step 7.
2. /* The caller is currently at a location where the callee visited before.*/ if the record of the MCL table of the LA stated in Step 1 says that this LA is a "Visited LA", then goto Step 3. If it says "Latest LA", goto Step 4.
3. The Locating request is forwarded to this visited LA (following the pointer given in the record of Step 2). While the request is forwarded to the next LA, the callee's record is again searched from this LA's MCL table. Goto Step 2.
4. The latest LA (who is overseeing the callee) finds the callee's record from the MCL table. If the value of the Pointer field is NULL, then the callee is right in one of this LA's governing Ras. Hence, a message is forwarded to the caller's VLR to make the connection. Goto step 13. If the value of the Pointer field is not NULL, then it must be a VLR who is currently overseeing the callee. Hence, the locating request is sent to the latest VLR to which the Pointer field refers.
5. The latest VLR sends a message to the caller's VLR to make a connection.
6. Goto Step 13
7. /* The caller is at a location where the callee has not visited before. Updates of the callee's new location in the LA, VLR, and HLR are associated with this locating operation */
8. The HLR forwards the request to the callee's LA.
9. The callee's latest LA forwards the request to the latest VLR. Also, the callee's record in this VLR's MCL table is updated by replacing its type with "Latest LA" and Pointer with NULL.
10. The callee's VLR acknowledges the receipt of the message to the LA and the LA will then update the callee's record in its MCL table by replacing type with "Visited LA" and Pointer with a pointer to the callee's current residing VLR.

11. the callee's VLR sends a message to the GLR. The HLR updates the callee's new location to the new latest LA (i.e., the callee's current VLR).
12. The HLR forwards the message about the current location (VLR) of the callee to the caller's VLR and the connection between the caller's VLR and the callee's VLR is built.
13. End of the Call Tracking Operation.

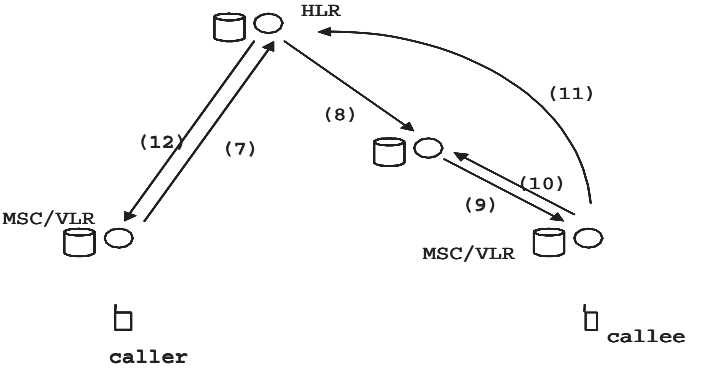


Fig. 2. The case of locating through the HLR

4 Traffic Cost Model

We present the traffic cost models that used to evaluate the performance of the proposed strategy.
The parameters used in our cost models are listed in Table 1. The costs of the IS-41 strategy, was discussed in the literature [4] [8]. In order to make a fair and reasonable comparison, we make some general assumptions and based on which we derive their cost functions in a uniform way. As the local database processing cost is insignificant comparing to the long communication time, we only consider communication cost in this derivation. Communication cost is dependent on the "distance" between two parties, and is classified into three levels: two parties are under different RSTPs, two parties are under the same RSTP but different LSTPs, and two parties under the same LSTP. Their costs are respectively h1, h2 and h3, we also need to use probability to model the location distribution of two communicating parties.

4.1 Cost Functions

The tasks of location management include managing registration operations and call tracking operations. Hence, the location management cost is computed

Table 1. The symbols of the parameters and their meanings

System Parameters	
Symbol	*Meaning*
h_1	The cost of sending a message from VLR to another VLR under a different RSTP;
h_2	The cost of sending a message from VLR to another VLR under a different LSTP but the same RSTP;
h_3	The cost of sending a message from VLR to another VLR under the same LSTP;
P_L	The probability of a mobile client's moving into a new RA which is under the same LSTP as the last RA that the client just left;
P_R	The probability of a mobile client's moving into a new RA which is under the same RSTP as the last RA that the client just left;
CMR	The call-to-mobility ratio;

according to these two operations. As the ratio of the number of calls (locating a callee) to mobility and defined as

$$total\ \ cost = \frac{1}{CMR} \cdot Registration\ \ cost + Calltracking\ \ cost$$

where CMR is the Call-to-Mobility Ratio.

We will use this definition in calculating the total cost of each strategy. For each operation, the time (cost) is counted from the instant that the VLR of the mobile client submits a request is finished. The cost of an operation that is not directly related to the locating operation is not counted in our cost functions. For instance, in the IS-41 scheme the location operation is counted from the time the client's new arriving VLR sending a request to the HLR till the time this VLE receives a message from the HLR to indicate that the location of the client has been update in the HLR. This is the cost of a location operation in the IS-41 scheme. Other operations such as sending a message from the HLR to the former VLR to remove the obsolete record from its table are not counted in our cost models. This is mainly because these operations are not as urgent and can be executed at a later time. Note that ignoring such a cost in our model is for reason of understanding the effect of other key steps in an operation. All such operations in different strategies are treated in the same way.

4.2 Cost Functions of the IS-41 Strategy

The IS-41 strategy is a standard scheme and is often used to be compared with other strategy. Hence, we first present the cost functions of the IS-41 strategy. Based on the definition of total cost, the total cost C_{Basic}^{total} of the IS-41 strategy can be represented as follows.

$$C_{IS-41}^{total} = (\frac{1}{CMR}) \cdot C_{IS-41}^{R} + C_{IS-41}^{T}$$

For clearness reason, we list the steps of the IS-41 strategy in the following for a reference. Its idea has been discussed in Section 2.The registration cost of the IS-41 strategy is therefore.
$C^R_{IS-41} = 2 \cdot h_1$
In this algorithm, the call tracking cost of the strategy becomes
$C^T_{IS-41} = 4 \cdot h_1$

4.3 Cost Functions of the DCC Strategy

We derive the total cost of the DCC strategy. Again, total cost can be stated as follows:
$C^{total}_{DCC} = (\frac{1}{CMR} \cdot C^M_{DCC} + C^L_{DCC}$
The algorithm of the DCC strategy is the same as that of the Dynamic LA strategy. That is,

$$C^M_{DCC} = \begin{cases} (P_R \cdot h_2 + P_L \cdot h_3) + \frac{1}{2} \cdot (h_2 + h_3) & if \frac{1-P_L-P_R}{CMR} < 1 \\ 2 \cdot h_1 & if \frac{1-P_L-P_R}{CMR} \geq 1 \end{cases}$$

As for the location cost, there is a subtle the DCC strategies in computing the cost. Notice that in the DCC strategy wherever a client moves out of the range of a RSTP, the new VLR (that the client just entered) becomes the new LA (an update). Updating a new latest LA makes the distance between two linked LAs much shorter than that in the previous strategy, and so does the communication cost, This factor is considered in the cost model.Two cases are considered.

1. Locate the callee through HLR.
2. Locate the callee througha VLR where the callee visited before.

As for case 2, its cost also involves four tasks (a), (b), (c), and (d). The costs of (a) and (c) of this strategy, however, are both $\frac{1}{2} \cdot (h_2 + h_3)$ The factor of h_1 is not in this cost, because the LA and its linked VLR will always be in the same RSTP. The cost of (d) is still $\frac{1}{3}(h_1 + h_2 + h_3)$. Computing the cost of (b) involves two cases:

• Two linked LAs belong to different RSTPs: Between two consecutive calls, a mobile client makes on average $\frac{1}{CMR}$moves. As the region of a RSTP is rather large, we assume that the callee will at most move out of the region of a RSTP once between two consecutive calls/ The other $(\frac{1}{CMR} - 1)$ moves are all within the same RSTP. That is, the cost of this case is

$(1 - (\frac{1}{CMR} \cdot (P_L + P_R)^{\frac{1}{CMR}-1} \cdot (1 - P_L - P_R)) \cdot h_1$

• Two linked LAs belong to the same RSTP: The probability of this case is 1-(the above case) which is $(1 - (\frac{1}{CMR} \cdot (P_L + P_R)^{\frac{1}{CMR}-1} \cdot (1 - P_L - P_R))$ For two LAs within the same RSTP, as they may be either within the same LSTP or different LSTPs, we simply take the average in computing their communication cost, which is $\frac{1}{2} \cdot (h_2 + h_3)$ Finally, we have the total cost

$$C^L_{DCC} = P + Q, where$$

$$P = (1 - \sum_{i=0}^{K_{C\&C}} P_i) \cdot C^L_{DLA}$$

$$Q = (\sum_{i=0}^{K_{C\&C}} P_i) \cdot \{\frac{1}{K_{C\&C}} \cdot \sum_{i=0}^{K_{C\&C}} [\frac{1}{2} \cdot (h_2 + h_3)$$

$$+i \cdot [(1 - \frac{1}{CMR} \cdot (P_L + P_R)^{\frac{1}{CMR}-1} \cdot (1 - P_L - P_R)) \cdot (\frac{1}{2} \cdot (h_2 + h_3))$$

$$+(\frac{1}{CMR} \cdot (P_L + P_R)^{\frac{1}{CMR}-1} \cdot (1 - P_L - P_R)) \cdot h_1] + \frac{1}{2} \cdot (h_2 + h_3)$$

$$+\frac{1}{3} \cdot (h_1 + h_2 + h_3)\}$$

5 Performance Analysis

We see two important factors that affect the performance of the DCC strategy:$K_{C\&C}$ and $P_i$where $K_{C\&C}$ is the length of a forwarding and P_i is the probability that a caller's request is issued from LA_i and its overseeing VLRs. Hence, we first study the effect of these factor to the DCC strategies. Also, we vary the ratio h1/ h3, which represents varying region size of a RSTP versus a LSTP. This is a general factor which affects all strategies. The default values of the parameters used in our evaluation are given Table 2.

Table 2. Default values of parameters

Parameter	h_1	h_2	h_3	P_L	P_R	CMR	$k_{dc\&c}$	P_i
Default Value	8	4	1	0.7	0.2	0.5	6	0.05

5.1 The Cost Result of P_i

P_i is the probability that a caller places a call from a VLR that happens to be under one of the linked LAs of the callee. When this occurs, the locating cost is cheap. We vary P_i from 0.01 to 0.16. As the default $K_{C\&C}$ is 6, the total probability of a call from the VLR under a linked LAs (*i.e.*, $\sum_{i=1}^{K_{C\&C}} P_i$) is actually 0.06∼0.96 The result given in Figure 3 shows. However, for the $C\&C$ strategies a dramatic decrease of the cost when P_i increases. Although in general P_i may not be large for every kind of mobile clients, it could definitely be so for a certain type of users. For instance, most of the commuters always drive along the same route between their offices and houses. Many office workers and students, etc., also appear at several fixed location regularly. Our performance result shows that the proposed strategy is especially good for managing mobile clients of this kind. The curves of the other strategies are flat because they are independent of the varying parameter P_i .

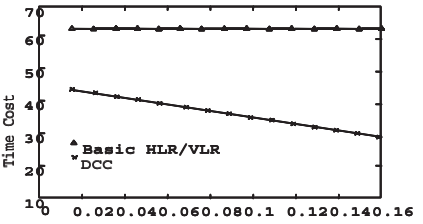


Fig. 3. P_i to the IS-41 strategies

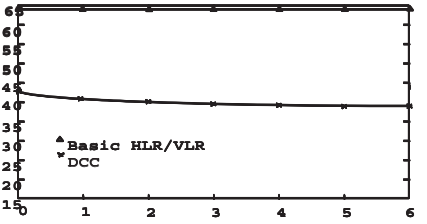


Fig. 4. 4 K_{DCC} to the DCC strategy

5.2 The Cost Result of K_{DCC}

In our proposed strategy, multiple LAs may be liked together due to the movement of a mobile client. The length of this link is represented by K_{DCC}. Figure 8 shows the result by varying K_{DCC}. A large K_{DCC} means that many VLRs that are under the linked LAs can locate a callee through the providing links of LAs, which helps to reduce the locating cost. However, the curves of the proposed strategy will become high after a certain K_{DCC} point. This is because the locating cost will increase with the lengthy linked LAs. The longer the linked LAs, the longer it takes to locate a callee. According to our derivation earlier, the length of the link should be less than $[4{*}h_1/c_1 - 3]$ in order to have benefit. The improvement of the proposed strategy over the IS-41 strategy is very significant.

The reason is the links in the proposed strategy are built on LAs. While a mobile client moves, he/she may move out of the boundary of a VLR easily and get into another VLR. But in the proposed strategy, the client doe not move out of the boundary of a LA so easily because the region of a LA could be much larger than that of a VLR.

6 Conclusions

In this paper we proposed a new mobility strategy for reducing location traffic cost in mobile networks. Different from the previous strategies, the proposed strategy can reduce the locating cost at the same time. The proposed strategy avoids updating the client's location to the HLR when the client moves to a new VLR. The client's new VLR always updates the client's location to the LA. We also derived the cost models of the proposed strategies and the related work. Our analysis results reveal that in most case the proposed strategy performs better than the IS-41 strategies. This is mainly because it can reduce the locating cost simultaneously, while the other strategies only reduces one of them. Our strategies are also easy to implement based on mobile computing environment.

References

1. Amotz Bar-Noy and Ilan Kessler, "Tracking Mobile Users in Wireless Communication Networks", IEEE Transactions on Information Theory, Vol. 39, No. 6, 1993, pp. 1877–1886.
2. Ing-Ray Chen, Tsong-Min Chen, and Chiang Lee, "Modeling and Analysis of Forwarding and Resetting Strategies for Location Management in Mobile Environments", in Proceedings of the 1996 International Computer Symposium (ICS'96), Kaohsiung, Taiwan, R.O.C., December 1996, pp. 121–128.
3. Ing-Ray Chen, Tsong-Min Chen, and Chiang Lee, "Performance Characterization of Forwarding Strategies in Personal Communication Networks", in 21th IEEE International conferences of Computer Software and Application (COMPSAC'97), Washington, D.C., August 1997, pp. 137–142.
4. Ing-Ray Chen, Tsong-Min Chen, and chiang Lee, "Performance Evaluation of Forwarding Strategies for Location Management in Mobile Networks", The Computer Journal, Vol. 41, No. 4, August 1998, pp. 243–253.
5. Ing-Ray Chen, Tsong-Min Chen, and Chiang Lee, "Analysis and Comparison of Location Strategies for Reducing Registration Cost in CS Networks", Wireless Personal Communications journal, Vol. 12, No. 2, 2000, pp. 117–136.
6. EIA/TIA IS-41.3, "Cellular Radio Telecommunications Intersystem Operations", Technical Report, Technical Report (RevisionB), July 1997.
7. Joseph S. M. Ho and Ian F. Akyildiz, "A Dynamic Mobility Tracking Policy for Wireless Personal Communications Networks", In IEEE Global Telecommunications Conference (GLOBECOM'95), Singapore, November 1995, pp. 1–5.
8. Joseph S. M. Ho and Ian F. Akyildiz, "Local Anchor Scheme for Reducing Signaling Costs in Personal Communications Networks", IEEE/ACM Transactions on Networking, Vol. 4, No. 5, October 1996, pp. 709–725.
9. T. Imielinski and B. R. Badrinath, "Querying in Highly Mobile Distributed Environments", in Proceedings of the 18th VLDB Conference, Vancouver. British Columbia. Canada, 1992, pp. 41–52.
10. T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management", Communications of the ACM, Vol. 37, No. 10, 1994, pp. 18–28.
11. Ravi Jain, Yi-Bing Lin, Charles Lo, and Seshadri Mohan, "A Caching Strategy to Reduce Network Impacts of PCS", EEEE Journal on Selected Ares in Communications. Vol. 12, No. 8, October 1994, pp. 1434–1444.

Mobile Animation Algorithm for Cyber Museum

Sung-Soo Hong[1] and Sang-Kil Kim[2]

[1] Dept. of Computer Eng. Hoseo University, Korea
sshong@office.hoseo.ac.kr
[2] Dept. of Computer Eng. Hoseo University, Korea
ksknet@empal.com

Abstract. Communication technologies have been brought tremendous change to various aspects of an ever-fast changing world at present. Particularly, the use of Internet and cyberspace is widespread in every corner of our life. In this paper we propose a gesture algorithm using an animation technology. It was developed for educational purposes, and accessible through the WWW of Internet. The visual processes can be adapted for a variety of fields. For instance, in Cyber Clam Museum, the visual processes were used to design a screen with a realistic image and create a system that makes possible show at 360. Therefore, in the system, mobile hosts can observe the detailed clam's structure and save the image. This proposed algorithm provides multiple techniques to user may manipulate, visualize and interact with image on the Web. And every such transformation as translation, rotation, and scaling can be applied in the picture interactively for the convenient and effective viewing. ...

1 Introduction

As information industry has been developed modern times, the demand for user-oriented information has increased more than text-oriented one. Human being prefer visual technique method to written text, for the visual recognition has been developed in recent years. Especially, he or she is trying to make every effort to communicate more easily with information users with visual configuration, using computer and through them. [3]

The previous computers provided only linguistic and two-dimensional pictures and to and to use mouse and keyboard only to control cyber world. But, it didn't provide enough for information users to believe the world as he or she saw it in cyber world. Displaying information on Internet gives users both time-saving and economic benefits. To provide museum and other useful information on Internet or through Internet will give much more educational effects. One of new methods which constructed museum on Internet, especially visualization, is new one which is generating interaction between computer and human. This visual scheme is creative art in expressing graphic and factual image, depicting on screen and giving the feeling of real thing vividly. Of course in mobilizing human imagination through three-dimensional space it is useful too. There are

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 586–595, 2003.
© Springer-Verlag Berlin Heidelberg 2003

a lot of problems to solve in order to construct factual image on Internet. First, the biggest problems of all, is the lacking of reality and control of the image for users. Furthermore, there is a problem in terms of Internet transmission velocity. In order to transform a real thing into object in cyber space through computer, the art which deals with factual method with three-dimensional space is required. That is, the emphasis on visualization make us artificial feeling without understanding traits of real objects. On the contrary, the constructing on factual movement causes this disharmony, confusion between fact and visual situation. The visualization using the JAVA can solve these problems at once. The animation generating from it can express real things freely and make the control of image on screen smoothly according to intention of producer. We can display animation on screen, stop, enlarge, contract, and control animation speed freely. Thus we can construct user-oriented system.

2 Related Work

One of the goals of image-based animations is rendering. There have been many great minds in the field. Chen [11] created images at a new point of view by preserving the images from one frame to another instead of using the traditional 3D rendering. Expanding [11] and Chen [12] introduced the Quick Time VR, which provides many different points of view using the environmental map from the real world to create images. In making panoramic images, Szeliski [4] used a new technique, which has no limits in inputting images. This rotating matrix technique only requires three parameters instead of typical eight and thus lets the user make panoramic images in a fast and easy way. Solving the problem of unexpected curves with the image technique Chen used in View Interpolation, Seitz [4] applied the principles of the projective geometry and called his way View Morphing. Levoy [8] interpreted that the 2D slices in a 4D function made a set using new images and created images in new points of view by sampling with no 3D information such as particular point matching. Horry [14] suggested Tour into the Picture, with which you can do animations by only using a piece of 2D image. Although it's amazing in that you can create animations even without a perfect 3D environment, it does have a disadvantage that you should make distinctions of backgrounds and select vanishing points manually. In his new portal texture algorithm, Aliage[1] divided the inside of a building into two groups, one of which is the space surrounded by the walls and the other of which is the portals that can been in the space such as doors, windows, and furniture. Then he replaced the geometric models seen through the portals with images. While the old approach presented all 3D models seen through the portals, this new method provides 2D images, which makes real-time processing possible. Debevec and Niamark succeeded in making the virtual environment of a Canadian national park by installing two 16mm cameras and catching the stereo images every meter. Kanade[13] introduced a stereo image matching technique with 50 cameras. Debevec[9] used aerial photographs to suggest a new approach to modeling and rendering a building. In this approach, the prepared 3D structures will

come up as the user moves the pre-defined primitives over the pictures and gets close to a boundary.

Recently, the technology for providing three-dimensional image on Internet has been regarded as an important field. Indeed, VRML that makes three-dimensional graphic image, requires VRML Plug-in set-up and is difficult to express a real image. Also, users should take a special training to operate the system. Moreover, the translating rate becomes much slower with a big file [3][5].

Currently, information through the Internet is provided by means of the first dimensional image or text. Although there are several ways to transfer a variety of information, it is rare to be shown as three-dimensional translation on the web. Such a three dimensional presentation is normally carried out in the space through Java Applets program. In fact, the JAVA language may be the best tool for expressing a three dimensional image on the Internet. However, there are still some problems such as ways to add real images to the Internet graphic or the object in the three dimensional space. The three-dimensional graphic tool or image-extracting procedure may be available to visualize the object. However, three-dimensional images can be created on the Internet by producing an optical illusion through a succeeding two-dimensional image. Indeed a system using a realistic image is a better way rather than simple graphic manufacturing for the creation of an effective visualization [10].

3 Mobile Animation Model

In a theoretical point, it is natural to make an imaginary actuality equivalent to real condition by using of imagination. On the other hand, an imaginary actuality may become a new art mode in a point of artistic view. As artists create a world through a painting, a musical score, a stage or a stone, the world of imaginary actuality will be artificially realized in the computer. If users plunge into the imaginary actuality and interact with a variety of items in the world, they must obtain a better pleasure. Now we first suggest a model for the interaction between users and system for the animation of imaginary actuality. How can we define a relationship between users and system for the animation of imaginary actuality? And what is the basic relationship? A model that we are to suggest in the present paper is based on the model provided by J. Latta et al. [2]. Our model mainly consists of inter-phases connecting human with imaginary environment. The final goal of our animation system of imaginary reality is that when users replace a real situation with imaginary one, users cannot realize any difference between two situations. Furthermore, our system shall enable users to feel the better color sense or contours.

As shown in Figure 1, users can look real things through a monitor on displaying module. Sensing module in our system is to sense human's action or gesture. This module includes a variety of sensors and digital camera equipped with input and output devices. Information received through above two modules is processed on Virtual Perception module and extracted according to user's idea. This module connects the physical module with the logical sensor and a

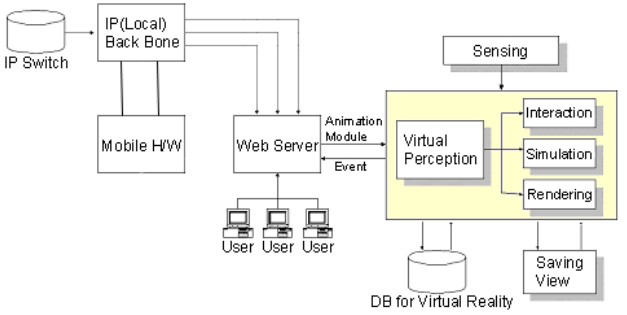


Fig. 1. Virtual Reality Animation Model

limitation that can be interacted with imaginary world is determined according to user's goal. All these works are determined on Interactive module.

Interactions with pre-determined environment are actually performed in the Simulation module. Of course, users can operate the basic processes without establishment of Interactive. Rendering module takes a part to draw the changed imaginary world. This module can reduce or magnify a real object and make a rotation to all sides in the monitor. The DB for the imaginary reality contains an image and a text of all objects in the imaginary world. This also includes geometrical, physical and behavioral characteristics on the intellectual action. Saving view can save and output the part that a user aims to.

The animation algorithm consists of an Init processor, a Display processor, a Zoom processor, and a Speed processor and all processors run by calling a run processor. The animation is done by creating a file from the image to be animated taken by a camera and by utilizing a Java program that is saved in the memory address of an image. In the run processor, a thread runs first to determine a rotation/stop and left/right rotation situation and then the animation is performed sequentially if the satisfaction is made. The reverse operation is performed in the case that the nix is reducing. Then, the Speed processor and Display processor are called.

```
   Init Processor()
{
imgName = getParameter ("image");
pauseTimer =
Integer.valueOf(getParameter("pausetime")).intValue();

   for (int i=0; i<count; i++) {
img [i]=getImage (getCodeBase (),imgName+indexName [i]+".jpg");
}
```

```
offscreenImage =
createImage (this.size (). width, this.size (). height);
offscreenGraphics = offscreenImage.getGraphics ();
setLayout (new BorderLayout ());
Panel panel = new Panel ();
add ("South",panel);
panel.setLayout (new FlowLayout ());

   panel.add(button_shift = new Button("reverse"));
panel.add(button_rotate = new Button("stop"));
panel.add(button_lowspeed = new Button("slow"));
panel.add(button_highspeed = new Button("fast"));
panel.add(button_zoomin = new Button("zoom in"));
panel.add(button_zoomout = new Button("zoom-out"));
}

   Run Processor ()
while (runner != null) // thread start
if(rotate == 0) // rotation, stop
if (shift == 0) // left, right
inx++;
if (inx >= aniCount-1) inx=0;
else
inx--;
if (inx < 0) inx=aniCount-1;
Speed ();
currentImg = img [inx]; // change image
Display ();
```

The display processor is one that displays an image sequentially on the screen after getting the information of the size and position of an image to be displayed by calling a zoom processor. The xpos represents an x-axis position, ypos is a y-axis position. Img Width is the width of an image, imgHeight is the height of an image, and drawImage is a Java function that draws an image.

```
   Display Processor ()

   Zoom (); // Calling zoom processor
{
int imgWidth = currentImg.getWidth (this);
int imgHeight = currentImg.getHeight(this);
if (zoom == 1) {
xpos=0;
ypos=0;
```

```
offscreenGraphics.drawImage(currentImg,xpos,ypos,this);
g.drawImage(offscreenImage,xpos,ypos,imgWidth,imgHeight,this);
}
else {
xpos=0;
ypos=0;
offscreenGraphics.drawImage (currentImg,xpos,ypos,this);
xpos=-1*imgWidth/2;
ypos=-1*imgHeight/2;
g.drawImage(offscreenImage,xpos,ypos,imgWidth*2,imgHeight*2,this);
}
}
```

The zoom processor is one that controls the size of an object. Here, the current image is the image to be displayed, the zoom parameter controls the image with the step of 1/10, and xpos and yos are an x-axis and y-axis positions of an image to be displayed, respectively.

```
   Zoom Processor ()
int imgWidth = currentImg.getWidth(this) + (zoom * 60);
int imgHeight = currentImg.getHeight(this) + (zoom * 40);
xpos = -1 * (zoom * 30);
ypos = -1 * (zoom * 20);
```

The speed processor is one that makes the speed of an image to be animated as optimum as possible.

```
   Speed Processor ()
try
Thread.sleep (pauseTime);
catch (InterruptedException e);
```

4 Implementation

The Cyber Shell Museum is implemented using HTML, Java Script and Java based on the Web. Technologies employed in this paper can provide better reality than still images of existing museums through visual effects that allows zoom, rotation as well as still views as shown in http://cyber.hoseo.ac.kr. The implementation has performed as follows: First, digital camera digitizes real shells with 360 rotational views or using animation technologies on developer's purpose. Second, real parts of shells are extracted and their sizes are adjusted to

employ using the Photoshop. Third, animation programs were developed to provide interactions with users considering zoom-in, zoom-out, 360-degree rotation. Finally, background of shells was determined and Java programs were developed to get system worked as animation.

For the shells of the world, about 1,000 species of shells are selected and they are classified into 5 kinds; Gastropods, Bivalves, Tusk shells, Chitons and Cephalopods, and more detailed classification has been done respectively. For example, in case of Gastropods, shells are classified into nine kinds of Ear, Cap, Pear, Corkscrew, Top, Spindle, Club, Barrel, and Egg using the appearance of shells in Fig. 2.

Fig. 2. World Shell

The Bivalves is classified into 7 categories, i.e. Discus, Fan, triangular, Boat, Paddle, Heart, Irregular. If the CLUB of Gastropods is clicked using mouse, selected kinds of shells are displayed in the left side of the screen. A user can select one of them to see related information and the front view of the selected shell through animated pictures. With various functions such as still picture, zoom-in, zoom-out, 360-degree rotation, the use can enjoy nice virtual reality in Fig. 3.

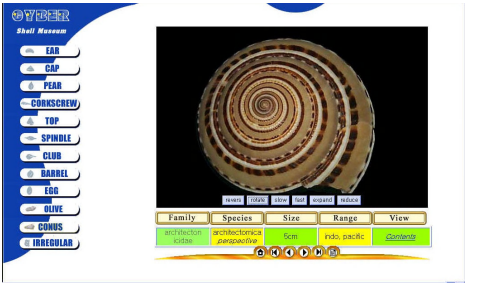


Fig. 3. Virtual Reality of Shell

Fig. 4. Korean Shell

For the shells of Korea in Fig.4, shells are classified into 10 categories and selected shells are displayed continuously with time interval. The world of Marvelous-shells contains 20 kinds of marvelous shells in the world and gives uses curiosity due to the fancy looking shells.

The Story-shells was implemented in order for students to get knowledge about shells. For example, it provides pictorial and textual information for easy understanding about what Gastropods is, terminologies for spire, aperture, apex, growth line, and etc. The Rare-shells contains five categories about rare shells in the world. A user can search shells by using images when the user doesn't know shells' name or appearance in Fig. 5.

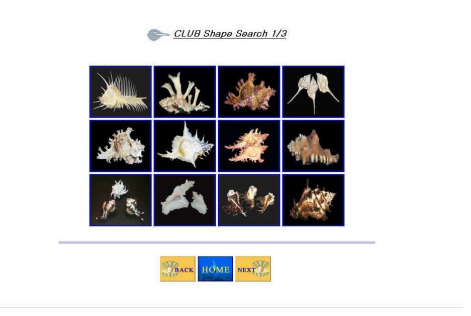


Fig. 5. Visual Search

If the user knows shell's name, the textual search can be used with shell's names and characteristics in Fig. 6.

After experiment was done by joining the shell museum system on Internet network (http://cyber.hoseo.ac.kr), advantages were found. The differences and advantages of using the animation technology with VRML system [Table. 1].

Fig. 6. Textual Search

Table 1. Comparison of Animation technology and VRML

	Animation	VRML
Advantage	*factual mage*	*cyber image*
Need System	*No*	*Plug − in program*
Image express	*very easy*	*very hard*
Dimension	$3D, 2D$	$3D$
Train	*No*	*special train*
Memory	*small file*	*big file*
Internet	*easy*	*very difficult*
Speed	*fast*	*slow*

In deed, VRML system that makes three dimensional graphic image, but there are big problem. First VRML system need a plug-in setup program. Second it is very difficult to express a real image. Also, users should take a special training to operate the system. Moreover, the translating rate becomes much slower with a very big file. So, we can not accept mobile system. The animation system can express real images freely and make the control of image on screen smoothly according to intention of Producer. We can display animation on screen, stop, enlarge, contract and continue animation speed freely. Thus we can construct animation system based on the mobile system.

5 Conclusion

In this paper we showed the similar system three dimension, using shell museum on cyber space technique and animation. So far we have utilized simple image, and the graphic visual objectives on Internet. Besides these, we could give more

detailed information that of going to museum in person using the factual image. The visualization of utilizing Java makes the Internet users who are familiar with complicated and difficult information understand easier. Furthermore, the users control the necessary information which is available directly. What is called a kind of users-oriented system? The weak-points of this system realized rather slow when we use telephone line. But we can solve the problem naturally by improving Internet transmission velocity promptly and the processing speed of computer. This paper, different from the visual image of graphic, gives not only users using the Internet much interests but also, the educational value. This cyber shell museum provides new conceptual Internet information and share materials which are display on web beyond space limitation. So far, we construct based on the materials for shell over 1000 species worldwide with animation.

References

1. D.G. Aliage, Anselmo, A. Lastra, "Architectural Walkthroughs Using Portal Textures", IEEE Visualization97, pp. 355–362,1997.
2. J. Latta, D. Orberg, A conceptual Virtual Reality Model, IEEE Computer Graphic and Application; vol. 4, No.1, pp. 23–20, Jan, 1994
3. John J. McArdle, "Benefits and Limitation of Mega-Analysis Illustrated using the WAIS, Vol.18, No.1, pp. 12–16 Sep, 2002
4. J.W. Shade, S.J.Gortler, R.Szeliski, "Layered Derpth Images" , SIGGRAPH98, 1998.
5. Kah-Seng Chung, Yia fourg Chen, "A versatile Digital Mobile Channel Simulation." Apcc 2002, pp. 10–14, Sep. 17. 2002.
6. L. McMillan, G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System ", SIGGRAPH95, pp. 39-46, 1995.
7. M. Brady et al, VRML Testing: Making VRML Worlds Look the same Everywhere, IEEE Computer Graphics and Application, Mar 1999, pp. 59–67
8. M. Levoy, P. Hanrahan, "Light Field Rendering", SIGGRAPH96, pp. 31–42, 1996.
9. P. Debvec, "Randering Synthetic Objects into Real Scenes : Bridging Traditional and Image Based Graphics with Global Illumination and High Dynamic Range Photography", SIGGRAPH98, pp. 189–198, 1998.
10. Saied Hoezzi et. al, Virtual view generation for 3D digital video, IEEE Multimedia, pp. 18–26, Jan 1997
11. S.E. Chen, L. Williams, "View Interpolation for Image Synthesis", SIGGRAPH93, pp. 279–288, 1993.
12. S.E. Chen , "QuickTime VR-An Image Based Approch to Virtual Environment Navigation", SIGGRAPH95, pp. 29–38, 1995
13. T. Kanade et.al, "Constructing Virtual Words Using Dense Stereo", ICCV98, Bombay, India, pp. 3–10, Jan. 1998.
14. Y. Horry, K.I. Anjyo, K. Arai, "Tour Into the Picture: using a Spidery Mesh Interface to Make Animation from a Single Image", SIGGRAPH 97, pp. 225–232, 1997.

Certificate Management Client System for E-transactions on Internet

Jeom goo Kim[1], Yoochan Ra[2], and Jaehwan Lim[3]

[1] Dept. of Computer Science Namseoul Univ., Korea
jgoo@nsu.ac.kr
[2] Dept. of E&IC Eng. Namseoul Univ., Korea
[3] Dept. of Multimedia Namseoul Univ., Korea

Abstract. In the paper we propose to design and implement an efficient certificate management client system. A basic function of the system authorizes client oneself, and manages to a certificate safely using the Rijndael algorithm. Also, this proposed system solves to the efficiency of private key and the vulnerability problem of certificate management in all kinds of E-transactions on Internet environment.

1 Introduction

While a diffusion and utilization on the Internet network were generalized, information distribution of various forms that used the Internet and a financial transaction, E-commerce became generalization. A certificate is the strongest means to prove an individual and an organization to use these E-transactions service. However, various problems have that an individual uses a certificate with direct management. It is hard to keep a certificate safely. The reason is because some certificate management system is not holding carelessness of a developer or standard observance to embrace, and it is containing a problem on security [1][2][9]. A certificate is a very important means to confirm personal identity in electronic transactions, Internet banking, electronic documents exchange and the same business on the Internet. Therefore, use is convinced of gradually, but it is embracing a problem on security that realistic use subject manages a certificate directly. We propose to design and implement that the client system could save a certificate safely and management that makes use of the Rijndael algorithm for a public key authentication method. The system have been implemented which applied authentication about client oneself and a dual authentication mechanism of safe private key save in order to improve vulnerability of the existing certificate management system.

2 Related Work

Various products were developed for Internet various user authentication in a lot of enterprises. We are considerate a characteristic about Magic PKI v2.0 of DreamSecurity company, UniCERT of Baltimore company, Managed PKI of

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 596–604, 2003.
© Springer-Verlag Berlin Heidelberg 2003

VeriSign company and MIT CA [11][12][13],and does in order to be comparable with a system to have been implemented in this paper. Magic PKI v2.0 ensures reliability of a mutual inter-working and service. It is based on wire and wireless integrated certificate issue function,and solves a memory load it uses a multi-thread method, and to occur in the existing fork method and a problem about delay of processing speed and implements the most suitable processing. Magic PKI v2.0 disconnected a key for issue and a key for an operator that raised safety of authentication system which applied certificate management business to various organization systems and maximized flexibility of systems operation. However, the disadvantage where Magic PKI v2.0 uses The RSA and the ECDSA cryptographic algorithm which is implementing a digital signature and an encoding function, and a speed was slowly [13]. The UniCERT reduced bottleneck phenomenon which applied a fabricated design,and it was constructed by hardware for established each department and a business burden about a part of a company in order to disperse duly. The UniCERT use is easy and flexible,and it is usable properly according to scale and there is the merit that compatibility is outstanding. Also,The UniCERT supported strong security, multiple security policy and carried various applications. However,The UniCERT had disadvantage capacity of a system grew larger for large compatibility, high expensive, and a lot of encoding and decoding time waste [11][13]. Managed PKI is suitable for demand of a place to use that increased and is composing. Therefore, it is hard to discuss because each characteristic is each different. However, Managed PKI is using ID and a password in a CA authentication method and has a weakness point problem of transmission layer. MIT CA does not provide application of a certificate and an up-to-date menu in a client, and application and an update of a certificate, all functions about distribution are performed through a MIT software distribution page of a MIT CA system. MIT CA is supporting verification of a certificate, safekeeping, and a function about a reading. However, development of all software cannot support the newest algorithm, and there is a security problem. The reason is because it was performed with construction of a Korbers system in the later half of 1990's [12].

3 Certificate Management Client System

This certificate management client simplifies a complicated process related to public key authentication and carries out efficient certificate management which lets encoding of a message coming and going in authentication of a client and an interval with certificate server and encoding save of a key be efficient and establishes a system in order to be safe. In order to be so, safe communication channel must be established between certificate server with the client which stands up simplification of appropriate business and an instant answer about a request of a client must be possible in the range that does not injure reliability of public key-based authentication system. Also, the system can verify a public key pair generation for a certificate generation and effectiveness true or no of a certificate with a client self-enemy that accesses an open treasure house which a

certificate and an inquiry of a certificate disuse list must be possible for certificate utilization and management to be efficient.

It can try to divide a module related to an authentication function of a client by a basic module for a certificate generation and management which an expansion module for certificate utilization greatly. A basic module can try to divide by an initial registration/authentication module for an initial generation of a certificate, EE initialization module, and a key pair generation module. An expansion module is a module to verify effectiveness of a certificate, the certificate update module that used key pair update, a certificate disuse request module and a certificate reading modules.

3.1 Client System Model

A certificate management client inter-works with certificate server, and it is asked authentication relation of a user and makes a request message. Then it takes charge of role which it receives a message of certificate server responded to in transmission and a request to authentication server, and to transmit to a user.

Client model structure: A certificate management client is composed of a request process module, a message switching module, a Kerberos module and a private key save module. A client as above compares with a small scale system administered on current construction and supports safe safe-keeping of private key used by electronic signature, and block off illegal message, a change message coming and going of a client between authentication server.

The key save module: As keep the private key used in encoding and decoding of a received document very important in authentication system. If it has been lost private key then basic reliability collapses in authentication system. Also a relevant certificate cannot use and future problems to occur remain. This paper used the Raijdael algorithm with encoding and decoding for private key safe-keeping. The Raijndael algorithm is the safest algorithm to be able to raise security intensity recently.

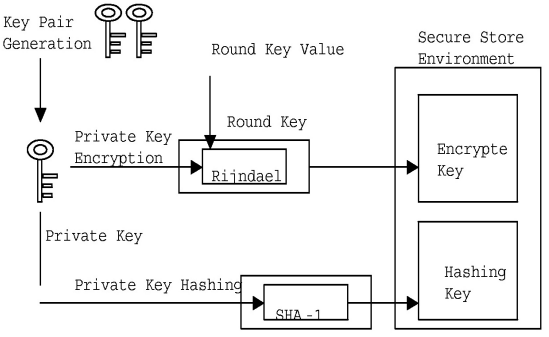


Fig. 1. The private key save module using the Raijndael algorithm

The Rijndael algorithm is block-cipher appointed the DES by ANSI in place of in 2000. A characteristic of the Rijndael algorithm is the 128 bit, 196 bit and 256 bit that block length and key length are variable, and the later cancer that carried out round algorithm of 14 times of 10 and 12, gets encoding and decoding data, and it is byte sub, shift row, key addition, mix column with round function for cryptography basically. A round for encoding carries out four above conversion processes in order, and round function for decoding passes through the process when reciprocal transformation holds a round for encoding. As for the Rijndael algorithm, it is fast as stream-cipher algorithm 128-bit AES, 256-bit SHA-1 and cryptology enemy intensity of the same degree. Fig. 1 shows the model that implemented the private key save modules.

Message exchange module: If it is exchanged a message for a client in the Internet where certificate server was opened to, it is exposed for message effluence and a security menace element as a change in a provincial office of hackers. A message related to a certificate is safe with encoding message format defined by RFC 2511, RFC 2797 and a PKCS #7 standard. However, it is important to establish safe communication channel between certificate servers with the client which stands up in order to raise security intensity more.

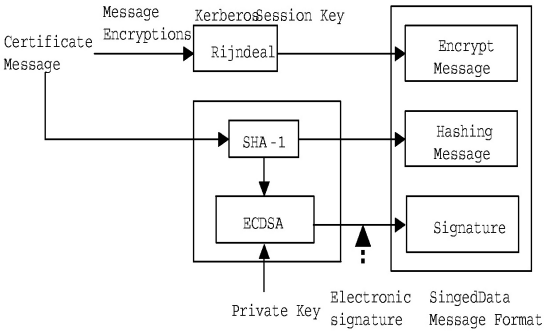


Fig. 2. The message encoding

1. Message encoding is to do an attack of a lot of form with developer and administrator who doing authentication to authorize about specific service to a user under distribution environment in order to be able to stop attack. There is a process a Kerberos authentication mechanism encrypts a message exchanged through established communication channel to transmit. A message to generate in a client or authentication server is composed to an encrypted message part and signature section. Fig 2 illustrates that describes a process generating the message which passes through the message encoding process that was encrypted. Key value of the Rijndael algorithm can exchange a message in a safe channel with what use the later exchanged confidential key which passed through a Kerberos authentication process in

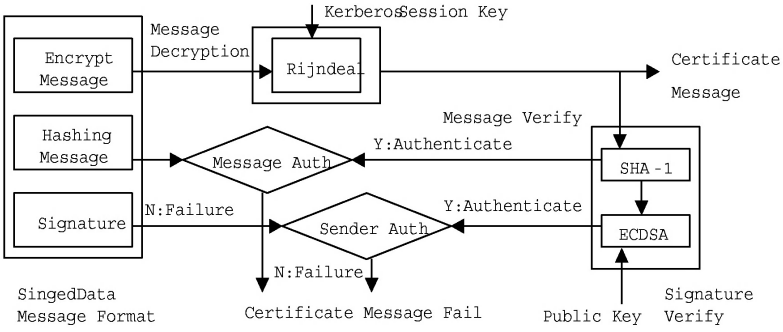


Fig. 3. The message decoding

a message-encoding model. The following is technology about a process to encrypt a message.

(a) It uses the confidential key which it got in a Kerberos authentication process when it uses Rijndael algorithm and encrypts a message.
(b) It uses the SHA-1 algorithm, and a message does hashing.
(c) It signs Hash value with private key of a message transfer.

2. Message decoding is the certificate management message which was received to a client or authentication server verifies decoding and signature to each part. Fig. 3 illustrates that describes these process.

(a) Decoding does a message with the confidential key which it uses the Rijndael algorithm which got a received message in a Kerberos authentication process.
(b) Hashing does the message which it uses the SHA-1 algorithm, and was restored.
(c) It uses the ECDSA algorithm which verifies signature with a public key of a message transfer.

Clint authentication module: A client system uses the Kerberos for safe message exchanging, but an adaptation of a Kerberos total system makes a certificate management system complicatedly that it is increased an unnecessary management element. Because it was so, it applied a Kerberos model of a simplified form in this paper and implemented.

As for the Kerberos which applied to this paper, an AS (Authentication Server) module was deleted. Even if authentication about ID and password of a client user operating in AS and an access ticket about the TGS deletes, it does not have a large influence on a total system. Clint and process that CA gets approval for message exchange can summarize as following.

1 step: ClientCA -> TGS: IDv || Tackett's || AuthenticatorclientCA
2 step: TGS -> ClientCA: EKc, tgs [Kc, v || IDv || TS2 || Ticket
3 step: ClientCA -> CA: Ticket || AuthenticatorClientCA
4 step: CA -> ClientCA: Ec, v [TS4+1]

(a) A client is ID of a user in order to order a service-approval ticket, ID of service to request, and transmits the message which included a ticket-approval ticket to TGS.
// User ID and a time-stamp creation to ask a service-approval ticket byte [ ] A = pbeCipher2.doFinal(ID); byte [ ] t = pbeCipher2. do Final (ts);
// Transmits a created ticket-approval ticket to TGS and requests a user and service authentication //
Con1.sendInt (cipher _Ticket.length); Con1.sendBytes (cipher _Ticket);

(b) TGS decode the ticket which was received, decides on success of decoding whether relevant ID exists and checks effectiveness time. The TGS compares a user ID and a network address with information to have come for confirmation of a user. If access of authentication server is admitted, the TGS issues a ticket-approving interface in the service that it requested.
// A service approval ticket issue about demand of a client byte [ ] cipher_Ticket = pbeCipher2.doFinal (Ticket);
// Guarantee value transmission to allow access of a client and certificate server//
Out.writeInt (cipher_Kab.length); Out. write (cipher_Kab, 0, cipher_Kab.length);
Out.writeInt (cipher_Nbr.length); Out. write (cipher_Nbr, 0, cipher_Nbr.length);
// Transmits a ticket approving interface from a client to the service that requested//
Out.writeInt (cipher_Ticket.length); Out. write (cipher_Ticket, 0, cipher_Ticket.length);

c) A client requests access to certificate server. A client sends the message that user ID and a service-approval ticket were included in authentication server for this object. Authentication server uses contents of a message to gain in authentication and approves use of service.
// It is decoding with a transmission message of a client byte [ ] decrypt_ID = pbeCipher2.doFinal(ID); byte [ ] decrypt_t = pbeCipher2.doFinal(t);
// Compares a message and approves service use
if (str.regionMatches (0, str1, 0, decrypt_ID.length))
if ((t1<ts*100) && (ts*100<t2*10000))

(d) It transmits a message to share an established confidential key, and it is used the same confidential key to gain in authentication with a client as a session key.
// Transmits an established session key to a client and shares a session key
Out.writeInt (ra2.length); Out. write (ra2, 0, ra2.length);

It is authorized a client in order to exchange a message for ClientCA proceeded by the fourth step between authentication server, and it is exchanged a confidential key necessary later by encoding.

3.2 Implementation of Certificate Management Client

The core of this client is a management class of a certificate adhering to certificate management protocol, the kerberos class which secures safe communication channel between a KeyStore class and a client and authentication server saves private key safely, and it is a CipherMessage class for safe message switching. Fig. 4 illustrates that is a formation table of each class composing a client.

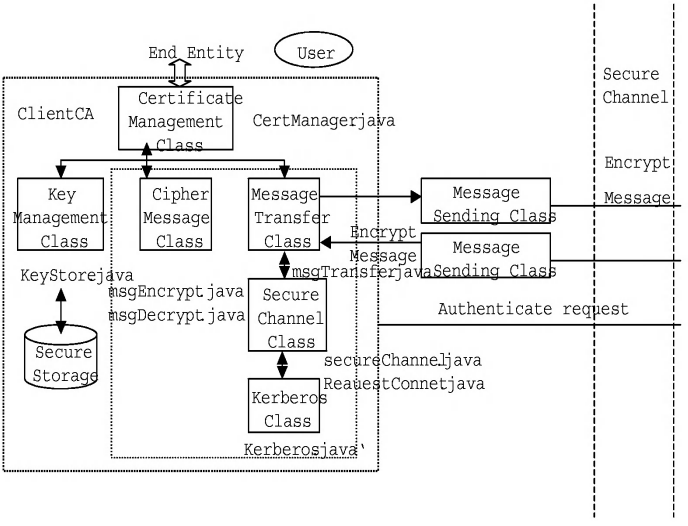


Fig. 4. Composition Class

Private key save: The private key which is used so that decipherment does a signature about a private key save document or a received document must be kept safely. Safekeeping of private key selects a private key save button in a menu of client application. Generally, it appoints safe mobile save media or a directory and database, and save of private key is saved. Safe communication channel establishment: A message exchanged for a safe communication channel establishment between client and authentication server is including personal information of a user. It is a security menace, and, as for these messages which is carried a cow by others when transmitted through an opened network. It has to form a secure data exchange channel between authentication server with a client in order to prevent these problems. This paper proposed the system is certified that used the Kerberos in order to be able to establish safe communication channel, and the system was finished a client and a user.

4 Comparison Performances

The client that implemented in this paper (like the client systems that mentioned) provides application of a certificate to manage a certificate, an update,

verification, and a private key save function. And a level can achieve high security requirements with security intensity applying high ellipse curve cryptography algorithm to a module it encrypts a message to transmit each request to, and to verify. Finally, it is coming illegal access and can block off a globe from the outside with using a Kerberos authentication mechanism if an individual client approaches authentication server effectively.

Table 1. Comparison with the proposed client and commercial software

Division	Proposal System	B Product	V Product	E Product	M Product
– Management function of certificate–					
Apply	Yes	Yes	Yes	Yes	No
Update	Yes	Yes	Yes	Yes	No
Verify	Yes	Yes	Yes	Yes	Yes
Save	Yes	Yes	Yes	Yes	Yes
Reading	Yes	Yes	Yes	Yes	Yes
— Algorithms–					
Key save	Rijndael	3-DES	3-DES	3-DES	DES
Message encoding	Rijndael	RSA	RSA	RSA	RSA
Message hashing	SHA-1	SHA-1	SHA-1	SHA-1	SHA-1
Message signature	ECDSA	ECDSA	ECDSA	DSA	DSA
— Protocols —					
Authentication	Kerberos	ID/PW	ID/PW	ID/PW	ID/PW
Encoding	Authentication Key	SSL	SSL	SSL	SSL

Table 1 compared and analyzed a function and an application algorithm with a client of a different product and the client system that it implemented in this paper. Also, it is same as BBB if it tries to currently compared to the Rijndael algorithm and difference with existing private key algorithm with the 3-DES used as private key save algorithm.

Table 2. Table 2 Block-cipher algorithm comparison

Algorithms	Block size	Key length	Number round	Attack
3-DES	64	168	48	K:2/112/56
RC2	64	8 ∼ 1024	18	C:64/64/.(16)
RC5	128	8s, s<256	16	C:83/./., C:123/./.(24)
IDEA	64	128	8,5	C:/56/67/32(3,5)
Rijndael	128	128,192,256	10, 12, 14	?

– An attack way explanation
∗ K:a/b/c: Plain text/cipher teat of 2^a is necessary in Known Plaintext Attack, must enconding of 2^b and needs memory of 2^c.
∗ C:a/b/c: Plain text/cipher teat of 2^a is necessary in Chosen Plaintext Attack, must enconding of 2^b and needs memory of 2^c.

∗ (r) : An attack round is numerical, and a round of algorithm is numerical
if it is a blank
∗ ? : If attack law is unknown

It has a smaller round number than the 3-DES, and the Rijndael can know that the clear attack way that it was made to know does not exist so far as it sees in the above table. That is, it is displaying that it is distinguished than the existing algorithm in the accomplishment speed side and a safety plane.

5 Conclusion

A certificate is carrying a form of electronic documents authorizing a public key of an owner in an organization having trust. And A certificate is including a position organization of an owner, a contact point and the personal information to the certificate. But it is not based on most carelessness of the authentication service product program developers who listened to or standard recommendations excluded a product used in an authorization, it holds weakness point on security because of the cause and it has a problem without certificate effluence changes, hacking and robbery about private key. It applied authentication and a security technology element of encoding to public key authentication system which it designed a safe certificate management client and implemented the plan that can solve these problems in this paper. A system proposed in this paper can be operated in the limit that it does not injure range of a standard document with safely. And specially, it will be usable in a small-scale organization and authentication system construction to use in the organization which had electronic transactions specific object effectively.

References

1. Joan. Daemen, Vincent. Rijmen, “AES Proposal: Rijndael”, 1999.
2. RSA Data Security, Inc., Public Key Cryptography Standards #1-9, June 3, 1991.
3. IETF, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, RFC 2459, 1999
4. IETF, “Internet X.509 Public Key Infrastructure Certificate Management Protocols”, RFC2510, 1999.
5. IETF, “Internet X.509 Certificate Request Message Format”, RFC 2511, 1999
6. IETF, “Certificate Management Message over CMS”, RFC 2797, April 2000
7. Baltimore, Inc., “PKI based E-security”, 2000.
8. William Stallings,“Cryptography And Network Security”, Prentice-Hall Inc., 1998.
9. The Advanced Encryption Standard, http://home.ecn.ab.ca/ jsavard/crypto/
10. Carl Ellison, Bruce Schneier, “Ten Risks of PKI: What You’re not Being Told about Public Key Infrastructrue”, Computer Security Journal, Vol. 16 No. 1, 2000.
11. IETF, “The Kerberos Network Authentication Service(V5)”, RFC 1510, 1993.
12. http://www.verisign.com/products/onsite/index.html
13. http://www.pki.or.kr/eng/detail.asp?co=14

Threat Description for the PP by Using the Concept of the Assets Protected by TOE

Tai-hoon Kim[1], Byung-gyu No[1], and Dong Chun Lee[2]

[1] KISA, 78, Garak-Dong, Songpa-Gu, Seoul, Korea
{taihoon, nono}@kisa.or.kr
[2] Dept. of Computer Science Howon Univ., Korea
ldch@sunny.howon.ac.kr

Abstract. Evaluation has been the traditional means of providing assurance and is the basis for prior evaluation criteria documents such as ITSEC. The Common Criteria (CC) defines a Protection Profile (PP) that defines the security environments and specifies the security requirements and protections of the product to be evaluated. The security environments consist of assumptions, threats, and organizational security policies, so the editor of the PP must describe the threats for the PP. In this paper, we propose a method for the description of the threats for the PP by introducing the concept of the assets protected by Target of Evaluations (TOE).

1 Introduction

The CC philosophy is to provide assurance based upon an evaluation of the IT product or system that is to be trusted. Evaluation has been the traditional means of providing assurance.

There are many evaluation criteria. The Trusted Computer System Evaluation Criteria (TCSEC), the European Information Technology Security Evaluation Criteria (ITSEC), and the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) existed, and they have evolved into a single evaluation entity, the CC. The CC for Information Technology Security Evaluation is the result of an international project involving NIST and the National Security Agency (NSA), on behalf of the United States and security organizations in Canada, France, Germany, the Netherlands, and the United Kingdom. These countries have worked in close cooperation with the International Organization for Standard (IOS) 15408 [1][2][3][4].

The CC is a standard for specifying and evaluating the security features of IT products and systems, and is intended to replace previous security criteria such as the TCSEC. This evaluation process establishes a level of confidence that the security functions of such products and systems, and the assurance measures applied to them, must meet.

The CC defines a PP that specifies the security requirements and protections of the product to be evaluated. If someone wants to select IT systems which

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 605–613, 2003.
© Springer-Verlag Berlin Heidelberg 2003

meet his requirements for security, he can use the PP to express his security requirements.

Those who want to express their security requirements using the PP must first define the security environments which consist of assumptions, threats, and organizational security policies, and connect the security environments to the security objectives and security requirements in the CC. When the writing the threat phrases for the PP, we use the production rule of multi-stage graph. But the assets and the attack methods increase rapidly, the count of the total threat phrases increases too and some new method for reducing these is needed.

In this paper we introduce the concept of the 'assets protected by TOE' and propose a new method for reducing the count of the threat phrases by using that concept.

2 Threat Description for the PP

2.1 Protection Profile

A PP defines an implementation-independent set of IT security requirements for a category of TOEs. Such TOEs are intended to meet common consumer needs for IT security. Consumers can therefore construct or cite a PP to express their IT security needs without reference to any specific TOE.

The purpose of a PP is to state a security problem rigorously for a given collection of systems or products (known as the TOE) and to specify security requirements to address that problem without dictating how these requirements will be implemented. For this reason, a PP is said to provide an implementation-independent security description. A PP thus includes several related kinds of security information (See the Fig. 1).

A description of the TOE security environment which refines the statement of need with respect to the intended environment of use, producing the threats to be countered and the organizational security policies to be met in light of specific assumptions.

2.2 Threating Be Identified

A 'threat' is simply an undesirable event, which is characterized in terms of a threat agent, a presumed attack method, an identification of the asset under attack, and so on. In order to identify what the threats are, we therefore need to answer the following questions:

- What are the assets that require protection? : The assets subject to the attack (e.g., sensitive data),
- Who or what are the threat agents? : The threat agent (e.g., an authorized user of the TOE),
- What attack methods or undesirable events do the assets need to be protected from? : The attack methods employed (e.g., impersonation of an authorized user of the TOE).

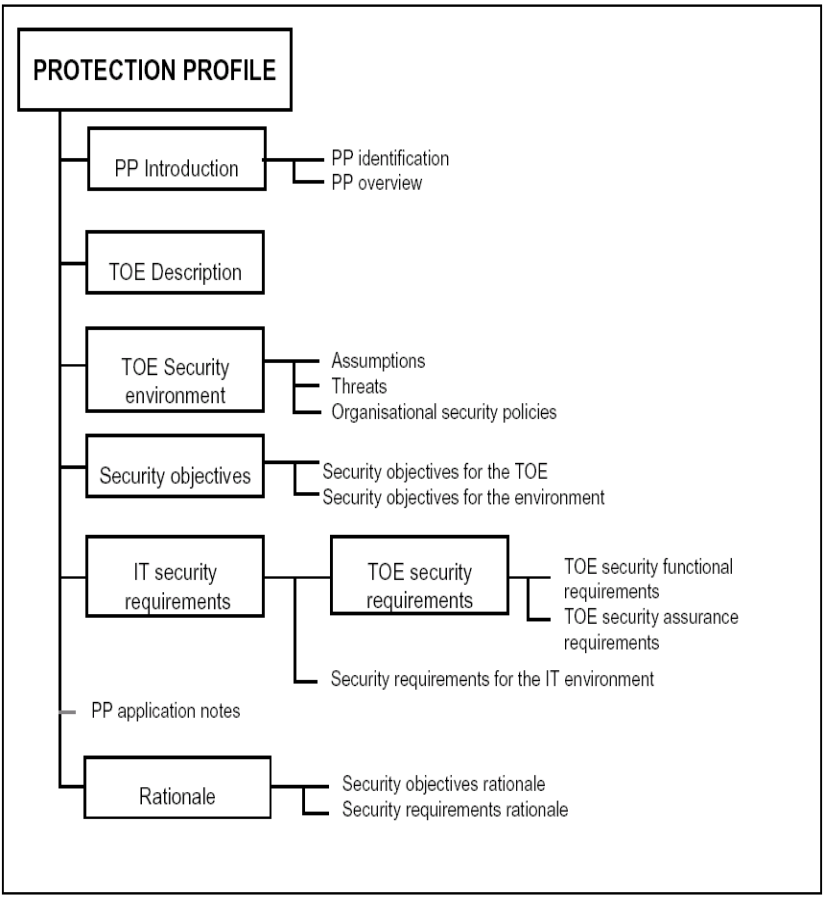


Fig. 1. Protection Profile content

2.3 Threats of the Wireless Networks

There are many kinds of 'threat' in the wireless networks and these are expressed in many books [7][8][9] or web-sites. But these threats are not the unusual things which are different from the threats other systems have. In fact, the security functional requirements in the Part 2 of CC are not restricted to the network systems as though they are the major target for development of the PP.

For example, there is the FPT_TDC family in the Part 2 of the CC which means 'Inter-TSF TSF data consistency'. In some cases, especially in a distributed or composite system environment, a TOE may need to exchange TSF data with another trusted IT product. The FPT_TDC family defines the requirements for sharing and consistent interpretation of these attributes between the TSF of the TOE and a different trusted IT product. But this family does not comment whether the environment is wire or wireless networks. In fact, the

difference between wire and wireless networks is not important for meeting of the requirements of the FTP_TDC.

The security functional requirements of the CC are described as high level form independent on a special technical factor. Therefore, they can be applied to many kinds of systems and security environments.

3 Describing the Threats in the PP

In the PP, in order to ensure we have a 'concise' statement of threats, the threat descriptions should be 'disjoint' as far as possible. In other words, there should be minimal overlap between different threats. This will help avoid potential confusion on the part of the reader of the PP as well as helping to simplify the PP rationale by avoiding needless repetition.

If we specify all threats at the same level of detail, the overlaps between threats can be more easily avoided. For example, it's not good that a threat description contains a detailed attack method against a specific asset if this is a specific attack scenario that is already associated with a more general threat stated elsewhere in the PP. Therefore, if we want to describe the threats for the PP, more considerations are needed.

Actually, all IT systems and communications channels face security threats that can compromise the systems themselves, the services they provide, the data stored on, and the data transmitted between them. For describing the threats for the PP, next items must be identified.

3.1 Identifying the Assets

In the case of writing threat phrases, it is very important to categorize the asset. When considering the assets of systems, we first think the factors threat agents want to compromise. Maybe we can use the next items as assets of the wireless networks.

- Wireless devices or systems
- Service that wireless networks provide
- Data stored on the wireless networks
- Data transmitted between the wireless networks.
- Other network resources.

In fact, these are very simple categorization of asset. For example, let's consider a wireless phone. The device user has is an important asset and must be protected from unauthorized user. If the device provides a commerce service, the service is protected from misuse, abuse, and so on. And the phone numbers or keys or passwords stored in the device are critical things and if these data transmitted to other system, they must be checked for confidentiality and integrity.

As mentioned earlier, the categorization of asset is very important thing to write threat phrases for the PP, because the threat may be described by using the assets.

3.2 Identifying the Threat Agents

Threat agents may either be human or non-human, and we can use the next items as threat agents of the wireless networks.

- Authorized users or owners of the wireless networks.
- Unauthorized users of the wireless networks.
- Attackers or intruders or hackers.
- IT entities.

When we consider the threats described in the PP, authorized users or owners of the wireless device is looked on as threat agent because they can commit abuse or unauthorized use.

In the field of requirement engineering, the concept of user is inclined to be restricted to human. But in the CC, the concept of user is extended to the IT entities and this may be correct thing because the malicious codes or viruses can be the user of the function.

3.3 Identifying the Attack Methods

Having identified the assets to be protected and the threat agents which may be the subject of attack, the next step is to identify the possible attack methods which could lead to a compromise of the assets.

This will be based on what is known regarding the TOE security environment. There is a very important thing must be considered. If we want to consider all attack methods already known, maybe it is impossible because the attack methods are too various and created or found everyday. Therefore, if we want to describe the threats about the TCP/IP, should use the phrase 'by using TCP/IP vulnerabilities already known' not dividing as like flooding, spoofing, DoS, and so on.

4 Description of Threat in the PP

After identifying the assets, threat agents, and attack method, we can describe the threats for the PP. For example, we can describe the threats concerned with the confidentiality of assets in the PP as like this:

- Threat agents may discover wireless networks by unauthorized methods,
- Threat agents may disclose the services provided by unauthorized methods,
- Threat agents may disclose the data stored by unauthorized methods,
- Threat agents may disclose the data transmitted by unauthorized methods,
- Threat agents may disclose the systems or network resources by unauthorized methods,
- Etc.

But if we describe the threats as like these, there can be too many threat phrases in the PP, so a new method defining the threat is needed. We did not consider various threat agents and the threat methods above. If we divide the threat agents as authorized user and unauthorized user, threats may be two times as above.

4.1 Identifying of the Threats Considered the Composition of the Assets, the Threat Agents, and the Attack Methods

Many cases, when we describe the threats in the PP, we can use the 'production rule of multi-stage graph' (See the Fig. 2). In other words, the threats can be described by the combination of the assets, the threat agents, the attack methods, and so on. (But the Fig. 2 is not complete form.) Therefore, there can be very many phrases for the threats, and these phrases are the headache of the editors of PP.

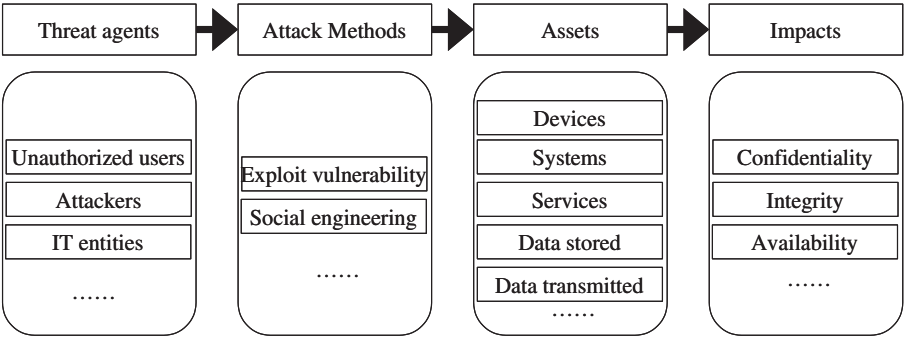


Fig. 2. Threat production rule of multi-stage graph (Example)

As you saw in the Fig. 2, the count of the threat phrase is decided by the next numerical formula:

Count of threat = Threat agent * Attack method * asset * Impact * Etc.

Therefore, we must re-consider and minimize the terms of numerical formula mentioned above for the efficient writing of threat phrases. In many cases, we can solve this problem by adjusting the 'Assumption' and 'Organizational Security Policy (OSP)', but this method is 'minimizing the threat agent' or 'reducing the attack methods possible'. When developing the systems which consider the security, most functional requirements are implemented by the threats not assumption or OSP. If we describe the security environments with many assumptions and OSPs, many of the functional requirements can't be implemented correctly and these systems may not cover the real threats.

4.2 Assets Protected by TOE

We propose a method for minimizing the assets of IT systems including wireless systems to reduce the count of threat phrases. As mentioned earlier, maybe we can use the next items as assets of the wireless networks.

- Wireless devices or systems,
- Service that wireless systems provide,
- Data stored on the wireless systems,
- Data transmitted between the wireless systems,
- Other network resources

In the aspects of the security requirements and the evaluation of the IT systems, we can re-identify the assets as like;

- TOE (target of evaluation),
- Assets protected by TOE
  - Systems not included in the TOE
  - Services provided
  - Data stored
  - Data transmitted
  - etc.

This method for identifying of assets is available for the PP, because the identifying of the security environments is connected to the security functional requirements included in the CC.

For example, let's describe the threat concerned with the confidentiality of assets (In this example, we didn't classify the threat agents and the attack methods, and applied 'the assets protected by TOE' to the description).

- Threat agents may compromise the confidentiality of the assets protected by TOE by unauthorized methods.

But this phrase is not sufficient because the threat agents and the attack methods are having the possibility to be divided.

It's very important to keep in mind that many kinds of threat agents and attack methods may exist in the 'assets protected by TOE. Therefore, the threat agents and the attack methods must be considered to contain all cases. For example, the phrase 'by unauthorized methods' is not suitable in some cases, because the 'threat agents' can contain the 'authorized users' and the authorized users may comprise the confidentiality of the assets protected by TOE by accident. So we should remove the phrases 'by unauthorized methods' and 'Threat agents', because all types of users and attack methods must be considered. We can use the threat description as like:

- The confidentiality of the assets protected by TOE may be compromised

As seen at the above phrase, in the case we use the concept of the 'assets protected by TOE', the descriptions for the threat are expressed as a kind of 'possibility' of attack. Some examples are like these:

- Malicious code may exist in the assets protected by TOE,
- Vulnerabilities may exist in the assets protected by TOE,
- Etc.

Systems, services, and others are included in the assets protected by TOE, and systems and services may consist of resources and source codes. Therefore, they may contain the malicious codes or vulnerabilities in the resources and codes.

4.3 Benefits

The benefits of this method we proposed are the following things:

Simplicity: When describing the threat in the PP for security environments, many threat phrases which finally come to same security functional requirement are needed because the assets, attack methods, and the threat agents are different. Sometimes, it's possible to use the assumptions or OSPs to reduce the threats, it is not sufficient. If we use the concept 'the assets protected by TOE', we can reduce the count of the threat phrase because the reason of dividing threat may be removed.

Expandability: Because of the fast growth of technology, everyday we are faced to many new malicious codes, viruses, and security holes. If does the new asset is added in the operating environments? If does the new vulnerability is found in the source codes which consist of a service? In these cases, we can solve problems by adding a few phrases to the 'PP INTRODUCTION' part of the PP, or maybe, if we are lucky, nothing will be changed.

Comprehensiveness: In fact, the form we used does not help the reader to understand what the security need is, especially since it applies to any TOE. But this may be another benefit of the method we proposed. The method using the concept of the assets protected by TOE is applied to many security systems like as Intrusion Detection Systems (IDS), Firewall, Virtual Private Network (VPN), and so on. The assets of these systems can be divided by using same way we proposed in this paper.

It's very important to keep in mind that the method we proposed is applicable only to the assets must be protected by the TOE. The threat descriptions for the TOE should only refer to potential events which could 'directly' compromise the assets requiring protection.

5 Conclusions

Many kinds of PPs are developed already and they have strong theoretical rationale about the development process. But there is no consistent theory for describing the security environments especially threats. Therefore, when readers review the PPs developed by other person or company and developers write the PPs they are to use make many kinds of mistakes and feel complexity. In fact, when we develop the PPs, we can find that there are many similar things between secure systems because the IT systems which aim to protect something have the similar structures and characteristics.

In this paper we proposed a method for dividing the assets and this method will be helpful to describe the threat phrase for the PP.

References

1. ISO. ISO/IEC 15408-1:1999 Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model
2. ISO. ISO/IEC 15408-2:1999 Information technology – Security techniques – Evaluation criteria for IT security – Part 2: Security functional requirements
3. ISO. ISO/IEC 15408-3:1999 Information technology – Security techniques – Evaluation criteria for IT security – Part 3: Security assurance requirements
4. KISA. Information Security Systems & Certification Guide, 2002
5. ISO. ISO/IEC 15292:2001 Information technology – Security techniques – Protection Profile registration procedures
6. ISO. ISO/IEC PDTR 15446 Guide for the Production of PPs and STs, Version 0.92
7. Russell Dean Vines. Wireless Security Essentials, 2002
8. Brian Carter and Russell Shumway. Wireless Security End-To-End, 2002
9. Randall K. Nichols and Panos C. Lekkas. Wireless Security Models, Threats, and Solutions, 2002
10. ISO. ISO/IEC WD 18045 Methodology for IT Security Evaluation
11. ISO. ISO/IEC WD 18028 Information Technology – Security techniques – IT Network Security
12. Science Applications International Corporation. Intrusion Detection System System Protection Profile, Version 1.4, February 4, 2002
13. Science Applications International Corporation. Intrusion Detection System Scanner Protection Profile, Version 1.1, December 10, 2001
14. Science Applications International Corporation. Intrusion Detection System Sensor Protection Profile, Version 1.1, December 10, 2001
15. Science Applications International Corporation. Intrusion Detection System Analyzer Protection Profile, Version 1.1, December 10, 2001
16. DGA. Protection Profile Firewall a exigences reduites, Version 2.2, 1999
17. NSA and SPARTA. U.S. Department of Defense Traffic-Filter Firewall Protection Profile For Medium Robustness Environments, Version 1.4, 2000
18. NSA. Mobile Code Desktop Draft, Version 1.0, 2000
19. NSA. Protection Profile For Multilevel Operating Systems In Environments Requiring Medium Robustness, Version 1.22, 2001
20. Schlumberger. Smartcard Embedded Software Protection Profile, Version 1.2, 1999
21. Atmel Smart Card Ics. Smartcard IC Platform Protection Profile, Version1.0, 2001

Real-Time Data Management for Network Information System

Dong Chun Lee

Dept. of Computer Science Howon Univ., South Korea
727, Wolha-Ri, Impi, KunSan, ChonBuk, Korea
ldch@sunny.howon.ac.kr

Abstract. Network system, such as switching system, are operated in the restricted conditions that the suggested events must finish in the time-constraints. In this paper, we propose the data model which is suitable to the unstructured multi-dataset environment and the algorithm to clarify the structure of our model. The proposed model supports the fast data access for unstructured dataset and enables users to easily retrieve data needed at the execution.

1 Introduction

The complexity of data in the network system is not so serious that the current network environment can be represented as the structured dataset and can be used enough to retrieve information just like Data Lookup. This characteristic in the communication application doesn't have any trouble ensuring the time-constraint process of data [1]. The recent network system has been through with various changes and the network environment should be connected with the plenty of services. The previous structured dataset saving and managing can come with limits of processing time and expansion. And with the increasing requirement of the communication common carrier for the software of network system that is compatible with open system, the definition and management method of communication data that have more complex and comprehensive semantic than the previous models are needed. In addition to it, the data throughput of the service systems has been sharply increasing in proportion to the data transmitting speed that is also getting faster result from the vitalized network services on the high-speed network. Consequently, faster data processing for the plenty of the data of complicated structure are imperative [2][3]. Generally database system is used as a way of data management. However, data processing system based on the disk is excessively big size and has disappointing data processing performance. It is not suitable in the aspect of data processing speed and system adaptation required in the communication environment. Thus, every dataset to be operated is made resident in the main memory in order to offer very fast and uniform accessing time in the time-constraint system environment

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 614–625, 2003.
© Springer-Verlag Berlin Heidelberg 2003

such as switching system and it is made to access with consistency and high speed [1].

This paper proposes the data model that can easily configuration the unstructured scheme. This model makes each data system have the unique Record Identifier (RID) in it, and has the complex relations and constraints that are required in the communication environment, represented consistently and shows the way of ensuring rapid processing performance through the navigation among the Record Identifiers (RID) without any physical reconstruction of the datasets at the request of multiple data.

2 Related Work

DREAM-S [1] has been used to manage call process and subscribers' data in the network system for the last years and it manages the information of the real world as a type of structured table. In the related data of routing process in switching center, dataset D-RT-RESOURCE-INF saves and manages the information about the root resource and D-SRT-RESOURCE-INF maintains the physical location information in the system that the sub-root belongs to. Meantime, each dataset is managed independently without any data model to clarify the correlations and constraints depending on designed semantic among datasets. The dataset, which has this type of construction, makes its structure and process simplicity. But it has a lot of restraints to solve the problems because it cannot read the semantic among the datasets due to the absence of the model. Unfortunately, the guarantee of the constraints makes the application systems more complex and it cause degrade of whole performance owing to the frequent access to the data processing system.

And also, when it is required to retrieve the information about the multiple datasets rather than to search a dataset as a simple lookup type, the Cursor is definitely needed as a tool that can locate the reconstructed dataset in the system, and support the consecutive access to the data [6]. The Cursor is a kind of structure used to access orderly to the data that are correspondent to current query context. The reason the Cursor is needed in the data processing system is that database system deals with the group of data that is not restrained in the size of them, and the general programming language doesn't have any specific means to deal with data. Since the previous Cursor saves the result data from the data service separately in the main memory as much as the size of it in order to allow the access orderly to the application program, it produces the problem of wasting the saving space in the main memory and it needs additional time and algorithm to construct extra system for the access to the pertinent saving space. Currently the structured scheme based on database set presents various solutions, but the structured scheme, that can store other different datasets in it, is very inefficient and complex in processing.

3 Data Model of the Unstructured Dataset

We describe the functional things provided by the system. That is, it describes more details about the composition of basic Dataset to operate a Cursor and the processes to recompose the related data through the Cursor. Dataset is the group of data with same quality or attribution. The data has inherently and means the data collection in the same type. Each data is uniqueness in the system, it was distinguish by the identifier with the physical location (that is, RID). The relations of the data are defined by regarding the related Identifier in the data as the attributions. For example, in the Figure 1, if there are any relations, **Rc1-c2**, between Dataset **C1** and Dataset **C2**, then related two data, **oc1**∈**C1**, **oc2**∈**C2**, have each attribution as the identifier **oc2, oc1**. And **REF, SET** and **Inverse** are defined to clarify the relations and the multiplicity of the relations.

Create Dataset C1 (Attrel Rc1_c2 **REF** C2)	**Create Dataset** C2(Rc2_c1 **SET inverse** C1, Rc1_c2, Attrc2)

Fig. 1. Definition of the relationship in Dataset

In the above example, **oc1** is unique relation with data in **C2**, on the other hand, **oc2** can show the relation with **oc1** as well as several data in **C1** since the **Rc2_c1** is declared by **SET**. Furthermore, Inverse means that it has always the both directions in the relations used **Rc1_2** and **Rc2_1** between **C1** and **C2**, which **C2** can be referred in **C1** by using **Rc1_2** and **C1** can be also referred by using **Rc2_c1** in reverse, and it means the guarantee about **Referential Integrity** between the two data. The clarification of the relation is made when the data has its identifier. For example, getting **oc2** in the related **C2** data specifies **Rc1-c2** in **C1** data. The specification of **oc1** and **oc2** into **Rc1_2** means that it's possible to access to **oc2** data through **oc2**.**Rc1_2**, and, in reverse, it's also possible to access oc1 data through **oc2**.**Rc2_c1**. The previously defined relationship between Datasets are shown as the starting root with basic dataset when it is used in the query to retrieve the data from the system, the retrieval of the data was complete by using the identifier which the attribution has in the specified path.

Figure 2 shows the query about the **Join** of **C1** and **C2** with the application of the above. As it is seen in the query, the reference to the other data through Identifier Rc1_c2 or Rc2_c1's value means the Join, therefore the path operator ("->") is used to refer to other data through Identifier. Path operator is used when it designates the other related data, it can be iterative joined with another data through the identifier in the designated data.

Retrieve Attrel, Rc1_c2-¿Attrc2 **from** C1
Retrieve Rc2_c1− > Attrc1, Attrc2 **from** C2

Fig. 2. Definition of the relationship in Dataset

In case that the **Join** is needed, it can be accessed to other Dataset, **C2** or **C1** directly by using **Rc1_c2 or Rc2_c1of C2** from **C1**, and it's possible to access without creating temporary Dataset so that it makes the system search faster than the previous methods [4][5].
Furthermore, in the communication environment, operating to retrieve the minor data which is identical to the condition is more often required than the operation to retrieve the entire Dataset after the **Join**, which often reconfiguration of the Dataset have many effects on the data processing performance in **Joining** [1].
And it is mainly described the navigation algorithm necessary for Join between related data with the identifier to express the relationship in the data model and the structure of the algorithm.

4 Navigation-Based Dataset Cursor in Implementation

Cursor is the structure to be used for the sequential access operation in the data of successful Query. While the previous cursor provides consecutive access to the recomposed data, the proposed method using the RID is described more flexible and effective cursor by direct access to the data through the navigation with no physical reconfiguration of the data.

4.1 Structure of the Cursor

The Cursor can be described with following four elements. At first, D is the dataset that has the information to be retrieved, and A is the group of data RID that are objects of search in D. The group of RID is distinguished by the path, and refers to the group of data that becomes the object of operation at the same time as the operation. C is the conditions to be met in order to return to the result, and consists of attribute, operator, and value. S indicates the attribution information retrieved when it meets the condition; **Cursor Csr = (D,A,C,S)** where D is Set of dataset, A is Set of data RID (=Alias) of the searching object in D, C is Set of condition expression and S is Set of result attribute.

For instance, the Cursor c' ∈ Csr that consists of "**retrieve** Rcl c2->Attrrc 2 **from** C1 **where** Attre1=10" in the definition of figure 2 can be expressed ({C1, C2}, {{oc1}, {oc1.Pc1_c2}}, {Attrc1=10}, {Rc1_c2->Attrc2}). The oc1 is the RID referring to the data in the C1, and oc1.Rc1_c2 is the RID referring to C2 related to oc1. Therefore, oc1.Rc1_c2->Attrc2 refers to specific character of

C2 data. To process the query, the Cursor is composed through the analyzing process of the query, and the result is retrieved through the navigator. The tree that uses the basic dataset as a root is built up when referring to other dataset through the relations like REF and SET from the basic dataset in the query, namely when the path is specified.

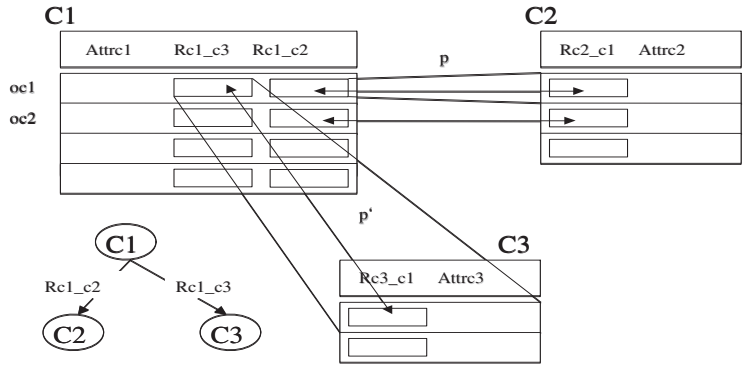


Fig. 3. Hierarchical structure of the path

Figure 3 shows that Rc1_c2 refers to C2 in C1 and Rc1_c3 can be the path to C3. Thus, the attribution in C2 using C1 as a path can be referred through the path expression that uses the Rc1_c2, the C1 character, and the character in C3, through the Rc1_c3. Moreover, the condition expression can specify by using the path expression. Here reference means Rc1_c2 and Cc1_C3 in C2 has the RID for the specific data in each C2and C3. Representing relations and retrieving attributions between datasets C1, C2 and C3 by accessing directly to the data in C2, and C3 can be possible by solely systemized representation, path expression.

4.2 Retrieval of Alias in Query

To connect and operate different and independent dataset through RID, the path to distinguish its dataset must be identified and be correspondent to the effective RID within its RID group. This is because the object data of C2 and C3 have to be changed together according to the object of operating in C1 changed from oc1 to oc2. This is the course of retrieving the RID of actual C2 corresponding to the attribution value of the specific data in the path p of C1 (Pc1 (Pc2_c2)-> C2). The change of Rc1_c2 into Rc1_c2' means the operating object of C2 has to be changed to Pc1 (Rc1_c2'). For this case, each path existing in the datasets must be analyzed and RID group correspondent to them should be produced.

This is called the **Alias** and it is defined as follows; **Alias A=(D,P,F)** where D is Dataset which refers to the current dataset, P is The path that distinguish the current D with previous Alias and attribution and F is REF/SET attribution.

Alias be distinguished by the paths referring to datasets. That is, the fact that C2 has the one path passing through only Rc1_c2 from the all the data in C1 shows that there is only one Alias.
Suppose the situation that several aliases can exist in one dataset. For example of C3, both RID by Rc1_c3 and RID referred by Rc3_c3 refer to the data in C1, but it must be classified by the context referring to C3. As a result of it, it is defined to only be distinguished by the previous dataset referring to itself in the path in order to distinguish the several RID in the dataset, its attribute, and all the three elements (dataset, character and path) composed by the compounded path. But, the basic dataset is defined to have the basic default path. For example, C1 has only the default path, C3 has the path of C1-C3, (C1, Rc1_c3, 0) (Here, this is the default path ID of C1 which is the basic dataset of 0). If C1 has self-reference structure, it will have additional (C1, Rc1_c1 and 0) except for the default path. And if it has the relations with C3 through (C1, Rc1_c1 and 0), C3 will have (C1,Rc1_3) and (C1, Rc1_c1, 0) additionally. This path is used to retrieve the Alias (= RID of dataset) which is the object correspondent to the current context in the course of navigating the classified dataset through the RID.
The Figure 4 is more specified figure 4 based on Alias. The basic dataset C1 (=BASE_dset) is initialized to have the basic Alias **al**(=BASE_alias). Alias **a2** of dataset C2 is produced by BASE_alias, al and Rc1_c2.When Rc1_c2', the relation from C1 to C2, has already been defined, this path produced new alias **a3**(Base_alias, al and Rc1-c2').

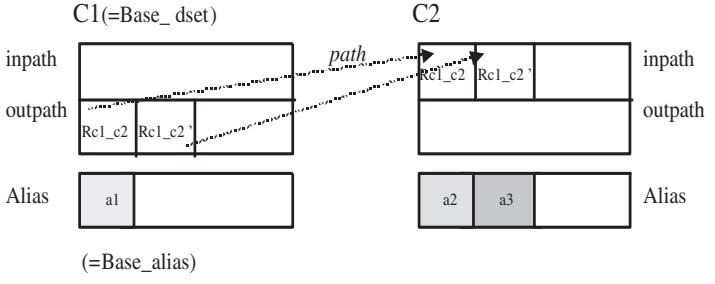


Fig. 4. Retrieval of the Alias

The following algorithm is the one to produce Alias by analyzing the path expression when basic dataset and the path expression are given. This path expression has the form of " s->n-> ..." and s and n are the names of attributions that are the types of REF or SET referring to each dataset.

```
Algorithm Makes Cursor (name: dataset name, p: path expression)
Dset: dataset information
Alias: working Alias begin
 dset = create_dataset_info (name) ;
 alias = dset.active_alias () ;
 while (  all attr in p )
   path = dset.search_out_path (alias, attr ) ;
 if (searching is successful)
    if (attr is leaf)
       cursor.add_attr_info(dset, alias, attr)
     end if
   else
     next_dset = dset.get_next_dataset (attr) ;
     if (no next_dset exist)
       create_dset_info (next_dset.get_name ()) ;
     end if
     next_dset.add_inpath (path)  ;
   end if
   dset = next_dset ;
   path' = dset.convert (path) ;
   if ( not dset.exist_path (path'))
     dset.create_alias (path') ;
   end if
   alias = dset.getAlias (path') ;
 end while
end.
```

This algorithm is noticed that new alias is produced when the path to the present attribution of present dataset doesn't exist.

4.3 Navigation Using the Alias

The composed Alias is needed to navigate the dataset. The way of navigation is made up be the evaluation of the condition expression. For this case, the Alias including the attribution stated in the condition expression evaluates the referred data and expression. If the evaluation of all the expression is successful, all the data referred by alias become the object of query result. If not, next data RID is replaced by alias and repeat the evaluation of the expression. The figure 6

shows such a process course, which it is 2 aliases in C2. If the alias **a3** is defined that it has a relations with C1 and SET, and if the evaluation of the condition expression about the data referred by current a3 (that is, RID is oc3) is not true, the a3 evaluates the expression replaced by possible next data RID, oc4. After the evaluation of the expression, the result is not true again, next RID is retrieved to replace a3. By the time it is not possible to retrieve the next RID, it is required (touch_coursor in algorithm) to replace alias a1 in C1 which is more upper path than a3. If a1 which is either REF or SET doesn't have any RID to replace, it is necessary to replace a1 itself. Namely, the next data except for oc1 is newly referred by a1 in C1. According to it, the lower alias (a2 and a3) than a1 are initialized by a1 and then begins to estimate the former expression.

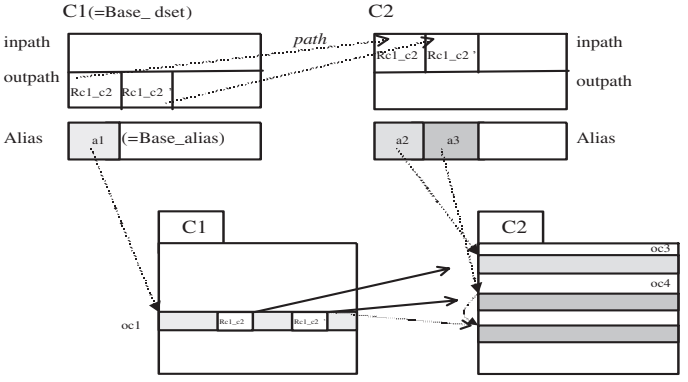


Fig. 5. Navigation of Dataset

By repeating this process, the results having evaluated all the data in C1 returned to the user. The following algorithm shows the course of evaluating the condition expression given on the basis of dataset and retrieving alias.

```
Algorithm Touch_Cursor (dset, alias, attrs)
begin
re-evaluate:
  while ( each attrs in all attrs )
   dset = attr.get_dset () ;
   alias = attr.get_alias () ;
   data = dset.get_data (alias) ;
   result = data.evaluate_expr (attr, attr.get_operator(),
            attr.get_value ()) ;
   if (result = true)
```

```
      continue
    end if
    status = touch (dset, alias) ;
    if (status = RequiredReEvaluate)
      goto re-evaluate
    return status
  end if
   end while
end.
```

The basic way to replace Alias is to require replacement of upper alias on the path when the alias itself cannot meet the condition evaluation. And also, the following algorithm shows the procedure to require the replacement of upper alias.

```
Algorithm Touch (dset, alias)
begin
  if (dset = BASE_dset and alias = BASE_alias)
     return RequiredNext ;
     pre_dset = dset.find_pre_dset (alias) ;
     path = dset.find_path (alias) ;
     if (path.get_type () = SET and not path, all_traverse ())
       oid = path.getNextSetElement () ;
       dset.set_identifier (alias, oid) ;
       return RequiredReEvaluate ;
     end if
 return touch ( pre_dset, dset.get_prealias ())
end.
```

In above algorithm, the **RequeiredReEvaluate** means that the condition evaluation given in query has to be reevaluated after being replaced by lower alias (a2 or a3) without any change of basic alias (For example, a1 in C1).

4.4 Retrieval Result

When all the condition evaluation are evaluated successfully, the data referred by current alias in each dataset have to be returned. If the query clarifies the specific character of data, only that specified attributions have to be retrieved and returned. For this case, the Cursor has the list of the result attributions and the result attributions are defined as following three elements; **Result attributions RS = (A, P, S)** where A is Dataset and alias where each attribution belongs, P is The location value of result value at the beginning of the data referred as alias and S is Actual size of the result value.

For instance, actual value states that attr3 is as big as size 2 and it has the same offset location as ofs1 of data (oc3) referred by alias a2 in dataset C2. Therefore, attr3 can be presented as (a2, ofs1 and size1)∈RS, and attr4 can be also described as (a3, ofs2 and size2)∈ RS because it refers to the character in the data 0c4, referred by alias a3 in the same dataset. The composition of the result value to be returned without considering multiple dataset is possible by cursor's retrieving continually the value as much in the location of data offset which specified alias refers to in the list of result characters.

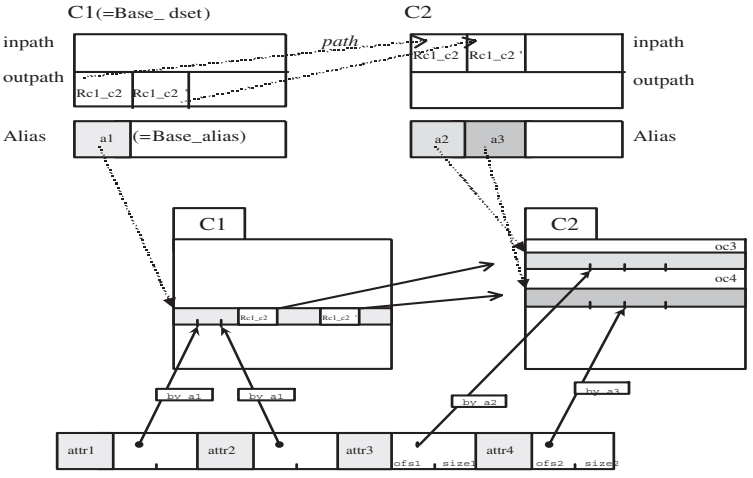


Fig. 6. Data retrieval

In the algorithm, **attrs** means the list of result attribution and results are the buffers to save the actual value of the attributions. When they are added to the result buffer, each attribute has the type and size information additionally.

```
Algorithm Get$\_$Info (attrs)
results: result values
begin
  while ( each attrs in all attrs )
   results.add_value (attr.get_data ()) ;
 end while
 return results;
end.
```

As shown above, the data retrieving method through Alias navigation makes it possible to retrieve the results without any physical recomposition compared to

previous methods [1][4][5] that recompose all the dataset in the query physically. This function offers rapider performance and more uniform access performance in retrieving many results.

5 Conclusion

When joining multiple dataset, previous methods are using the way of recomposing new dataset or index by copying a lot of memory though it is the database in the memory. As a result, it cannot avoid the bottleneck problem [7] of the memory which has to be minimized in terms of current hardware structure and this may consume much preparing time depending on the size and the state of the dataset [8].

We propose the Join method which minimize the physical repeating data service between memories. This method makes it possible to get the results, tracing the addresses of memories without any physical memory copy according to the relations at the request of retrieving. More specifically, we are proposed structured access system and interface through the data model that is including path expression in term of the dataset that has the unstructured and complex structure.

As a result, the proposed method makes the unstructured scheme represented easily in the communication environment and enables us to access to data easily, and meets the time-constraints of event which can be made up at the time of dataset Join and has the effect in reducing the complexity of the application program.

Acknowledgments. This work is supported by Howon University Fund 2003.

References

1. Y.I. Yoon, et. al., "Scalable Distributed Real-time Database Management for Switching System," Int'l Switching Symposium (ISS'97), Toronto, Canada, pp. 539–545, September 1997.
2. M. Stonebreaker and P. Brown, "Object-Relational DBMSs: Tracking the Next Great Wave," Morgan Kaufmann Publishers, pp. 320, 1998
3. W. Wolfe, J. Prichard, L. Dipippo, and J. Black, "Real-Time Database Systems: The RTSORAC Real-Time Object-Oriented Database Prototype," Kluwer Academic Publishers, pp. 279–303, 1997.
4. Polyhedra Plc., "Bloor Research Overview of Polyhedra," White Paper, http://www.polyhedra.com
5. Raima Co., "Raima Database Manager++ In Real-time and Embedded Systems," White Paper, http://www.mbrane.com/rdm.html
6. C. T. Yu and W. Meng, "Principles of Database Query Processing for Advanced Applications," Morgan Kaufmann Publishers, pp. 512, 1997.

7. Peter Boncz, Stefan Manegold, Martin Kersten, "Data Architecture Optimized for the New Bottleneck: Memory Access," Proc. of VLDB'99, 1999.
8. H. Garcia-Molina and K. Salem, "Main Memory Database Systems: An Overview," IEEE Transaction on Knowledge and Data Engineering, Vol. 4, No. 6, pp. 509–516, Dec. 1992.

Efficient Migration Algorithm for Mobile Agents in Wired/Wireless Internets

Sukwoo Han[1], Kuinam J. Kim[2], and Young-Sung Kim[1]

[1] Dept. of Computer Science Chonbuk National Univ., South Korea
swhan@wkhc.ac.kr
[2] Dept. of Information Security Konggi Univ., South Korea

Abstract. Mobile agents are active objects that automatically migrate from one node to other node in wired/wireless Internets. Due on failures of clients, mobile agents may be blocked or crashed even if there are available service. To solve this situation, we propose efficient migration algorithm with reordering and backward recovery of the paths to guarantee the migration of mobile agents. The proposed algorithm not only affords to avoid any faults of nodes or clients of mobile agents on Internet but also affects to agents' life span.

1 Introduction

Mobile agent technology has been applied for various application areas such as wired/wireless Internets management, and E-commerce [1][3][4][7][8][11]. Mobile agents are autonomous objects that migrate from node to node of Internet and provide to user which have executed themselves using database or computation resources of clients connected by the wired/wireless Internets. To migrate the mobile agent, it will be needed a virtual place so-called the mobile agent system to support mobility [8]. Several prototypes of mobile agent system have been proposed in several different agent systems such as Odyssey [3], Aglet [4], Agent TCL [19], Mole [2], and so forth. Most systems are little ensured its migration for a fault of Internet nodes or a crash of clients (or hosts), which may happen during migrating after a mobile agent launch. On the faults such as a destruction of the nodes or the mobile agent systems, we may consider that mobile agents may be destroyed, or blocked against the seamless wired/wireless Internets processing. It is no natural attribute to monitor the seamless progress of agent's execution, in a viewpoint to guarantee the autonomy of mobile agents. Therefore, we proposed a strategy with the simple techniques of path reordering and backward recovery to migrate mobile agents. It is implemented in the JAMAS, which we developed a model of mobile agent system based on Java language for system independent platform.

2 Related Work

Most of mobile agent systems provide a virtual place for migrating mobile agents under our basic ideal condition that there are no faults on the systems or

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 626–634, 2003.
© Springer-Verlag Berlin Heidelberg 2003

nodes, or include relevant protocols. While a mobile agent is launched to specific nodes/clients according to relevant routing schedules [4], it is possible to happen some problems about migration of mobile agent if the host happens an accident within where the agent visits and executes. Typically, ORB [6][7] implements distributed garbage collection in order to delete objects having no more references. Voyager [6] provides five policies for mobile agent's life cycle. Mole [2] supports the shadow protocol for orphan detection and successful termination for agents in mobile agent systems. The protocol is for detecting and processing what occurs any fault on migrating mobile agents. However, it does not provide to guarantee migration reliability of mobile agent. There is a simple protocol using transaction message queue [10], which is a procedure that the sender puts messages in the queue and receiver gets messages. There is also the same problem as the process of autonomous mobile agent in that it does not include facilities for monitoring the progress of an agent's execution. For example, assume that there is an agent in input queue of a host and the node's error occurs before the agent moves to queue of next node. Then the agent is blocked until that the node is recovered. This situation differs from problem in client/server. Mole [2][10] provides a fault tolerant protocol to support effective way for 'exactly once' migration using voting and selecting protocol as copying mobile agent to all nodes

3 Efficient Migration Algorithm for Mobile Agents

Figure 1 shows how the node repository can use in implementation instead of transaction message queue for agents. Assume that an agent moves from a node to the consecutive node along the path N1→ N2→ ...→ N(k-1)→ Nk (where Ni is a Internet node, Hi is a client (or a host), and Ri is an agent repository). As an agent may visit the same node several times Ni and Nj (1<=i, j<=k) may denote the same or different nodes. Assume further that an agent is stored in a repository when it is accepted by the agent system for execution. Except Nk, each other node performs the following sequence of operations on Transaction Ti such as Get (agent); Execute (agent); Put (agent); Commit. Get removes an agent from the node's repository. Execute performs the received agent locally. Put places it on the repository of the host that will be visited the right next time. Three operations are performed within a transaction and hence consisted of the atomic unit of work.

In Figure 1, we assume to happen a failure in a particular node Ni within the migration path of the mobile agent. For example, although the node Ni of the client Hi lives, the agent can't be migrated. Inversely, although the node Ni can be communicated with the previous node N (i-1), the agent can't occasionally migrate if the client Hi does not operate the agent system. In the above cases, the agent is never arrived by the last node Nk. It may be very weak point that the agent at previous client Hi-1 needs to receive user's assertion. In the worse case, if a shared client on the multiple agents launched occurs to crash on launching,

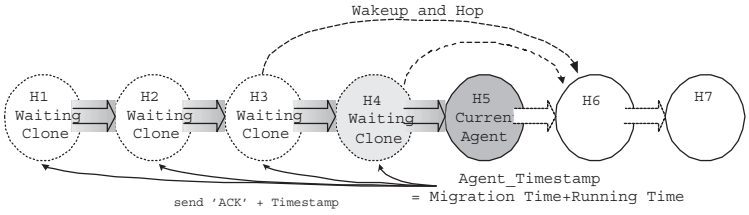


Fig. 1. Migration path of a mobile agent.

the agents will block or destroy even if other nodes are available to process continuously.

3.1 Reordering Algorithm of the Path

Figure 2(a) supposes that there is a migration path corresponding with an agent's routing schedule and some faulty nodes, such as N3, N4, and N7. An agent migrates and executes from node N1 to N2 sequentially, but it is blocked at the client of node N2 until the node N3 is recovered. If the node N3 dose not recovered, the agent may be orphaned or destroyed by the particular client. To solve this situation is for the agent to skip the faulty node N3 that includes on the migration path, and to move the address of node N3 back to the last one of the migration path. And then, the node N2 successfully connects the next other node N4 without any fault. Node N4 also has a particular fault. Therefore, node N2 Hop the right consecutive next one of node N4. As the same method is also applied to other nodes, the agent's migration path has reordered. That is, despite of any particular faulty nodes, the agent tries to connect subsequent nodes for the migration touring. This solution changes the previous arranged migration path by connecting with normal nodes except that some nodes have the particular fault. Afterward, the agent retries to connect each certain fault node after it waits for the timestamp assigned by the mobile agent system. If the certain faulty node is recovering by the timestamp, the agent will succeed in migrating to the destination node. Otherwise, the address of the faulty node will be discarded. Since the agent may be apt to loophole, we will give a restriction against the number of reconnection times. Figure 2(b) shows that a whole re-arranged migration path for the mobile agent be changed by this strategy.

Algorithm 1: Reordering of the path

For each agent's routing-table {
Extract a target address and fail _ checked information;
//multicasting the signal to eliminate that clones if the agent has no more than the destinations
if (no more a target address)
Backward multicasts 'Agent _Fire' signal to successful _ target nodes;
    //noticing of the flag to re-entering for the failure of some nodes

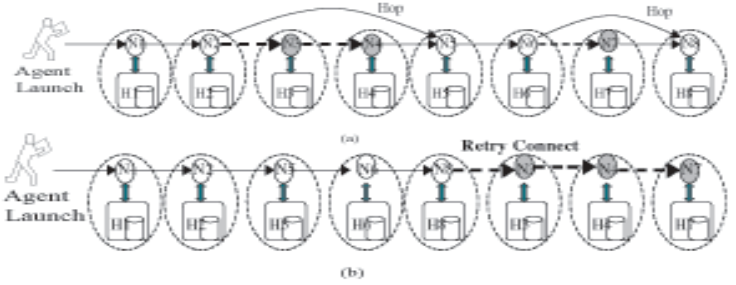


Fig. 2. The reordering before and after meet with faulty nodes

```
if ( exist a fail _checked _address) { // check whether connect more than two
times or not
Wait the agent during some system _ timestamp;
    Try to connect Socket to the address;
    if (success) {
        Call go Agent;
            Exit;
    } else {
            Notify to user the address is unavailable;
            Ignore the address; }}
//trying to connect the destination node the agent starts to migrate.
else if (not a fail _checked _address) {
    Try to connect Socket to the destination node;
    if (success) {
        Call go Agent;
            Exit; }
//if the agent does not connecting or migrating, set the flag and notify to user,
// Re-ordering the path in the routing-table at the same time.
    else {
            Notify to user;
            Move the current failed _address to last in the routing-table;
            Set the fail _checked information;
} } }
```

The mobile agent system executes the path reordering to connect every nodes of the migration path. If the agent doesn't connect a particular node in the middle of reaching the destination node via the migration path, it will succeed with connecting the right next one of a particular node of the path. After the failed address is pushed to the top one of routing table, it will retry to connect to the node. When it does reconnect each failed node, it does wait as much the timestamp assigned by the mobile agent system to reconnect. If it passes over the timestamp, it does ignore this address, and repeatedly connect the next faulty node. We have a limitation for reducing Internet overhead that mobile agent can

just try two times to connect the failed node. That means a mobile agent to occur infinitive looping for just connection. Although it is connected, it applies equally the same as that way if each host of the node errors the mobile agent system. Algorithm 1 offers automatically to reorder the migration path when the mobile agent can not migrated the next host due to the faulty nodes.

Backward Recovery Algorithm. In Figure 3, we suppose that the migrated agents execute autonomously at the client H5. If the client H5 of node N5 crashes, all agents at that host are blocked or destroyed. To prevent it, the agent copies itself (that is, the clone) in the current client when an agent migrates the next node after it ends its job at the current client. The clone is unconditionally waiting until receiving an acknowledge signal 'ACK' from the next client. If the signal 'ACK' doesn't reach to the current client H4 within the timestamp from the next client H5, the cloned agent at the client H4 has automatically activated to resolve this hindrance. Consequently, it hops to the next node N6. If the agent faults at the client H6 on execution, it will work repeatedly the same method to connect the other next node. In Figure 3, it happens that the current running agent at a client H5 is destroyed by a particular clash. At the same time, if the current client H4 also happens the succeeding fault, a cloned agent (which has already copied the previous client H3 of the client H4) wakes up and re-runs. This is so-called Backward Recovery.

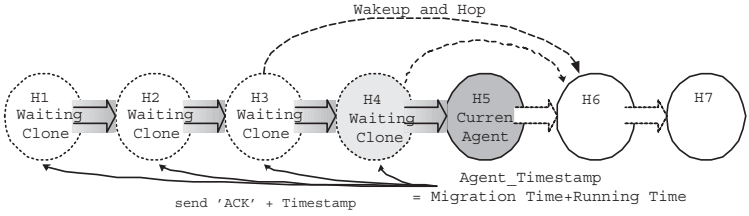


Fig. 3. An example of Backward Recovery

The backward recovery method is as follows: the agent system copies the clones of its agent on the client before migrating to the right next client. Each clone is waiting by it's own timestamp. Its timestamp of the original clone is maximum at the original host of the migration path, the next clone will be less than the migration and execution time of the previous one, and so forth. From launching an agent, the timestamp accumulates every clone of the previous clients with it's own moving and running time before it depart for the current client. Therefore, clones are waiting by the timestamp. Each clone spontaneously revives and attempts to work the path reordering as soon as regarding as a clashed host when none received any signal from the next host. At the final destination's node of the host, the agent system should broadcast a signal 'Agent _Fire' to all copied clones of the agent excepting the faulty nodes and failed hosts until reaching the destination. In such method, the algorithm 2 provides a

backward recovery method for which mobile agents support adaptive migration from a host to the next one.

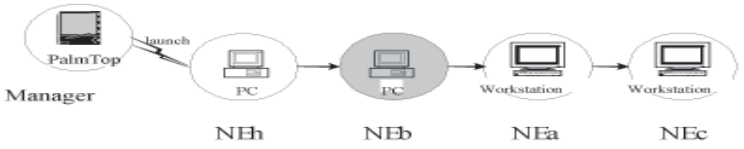
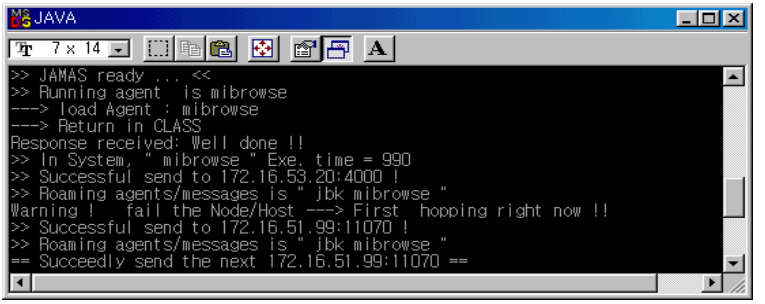


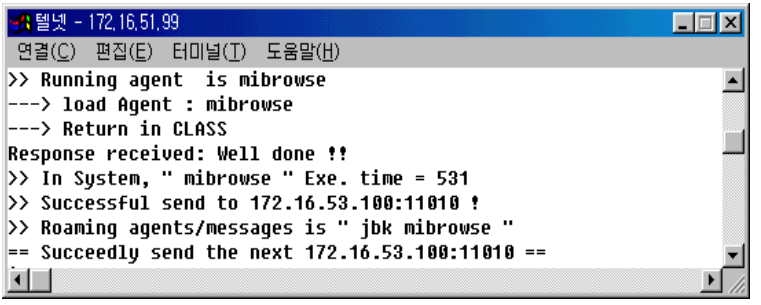
Fig. 4. A routing path with a fault of clients NEb

Algorithm 2: Backward Recovery
Waiting Clones Check

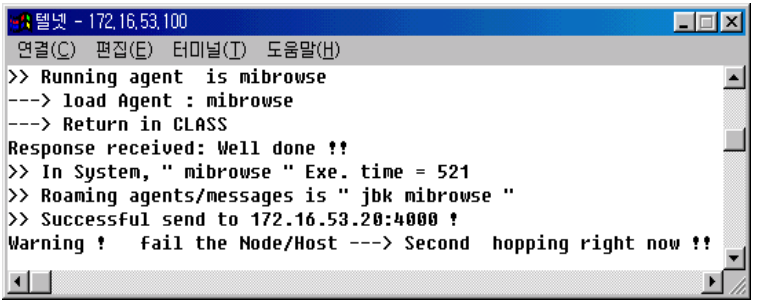
```
// Periodically checking the timestamps of the clones.
for each sleeped _clone
    if (empty a clone _timestamp) {
        Notify to user;
        Call wakeup Clone; }
Go Agent // migrating the agent or cloning it.
Send the agent;
Wait the agent's 'ACK' signal during send _timestamp;
if ('ACK') {
        Clone the agent;
        Call sleepAgent; }
        else call wakeupClone;
Arrive Agent // if the agent arrives in the JAMAS, noticing the previous node
with the signal and
//and executing.
    Send 'ACK' to the previous _node;
    Execute the agent;
Sleep Agent // Each cloned agent is waiting for the assigned timestamps
for each cloned _agent {
    Add agent _timstamp to system _timestamp;
    Add the agent to the sleeped _list;
    Sleep the agent;}
Wakeup Clone // re-activating the clone of the agent.
for the sleeped _list
    Find a cloned _agent;
Remove it from the sleep _list;
if ('Agent _Fire') remove the cloned _agent;
else {
    Move the current failed _address to last in the routing-table;
    Set the fail _checked information;
    Call the algorithm 2; // arranging the path. }
```



(a) A screen shot of executing at the NEh



(b) A screen shot of executing at the Nea



(c) A screen shot of executing at the NEc and attempting migration of the second at the NEb

Fig. 5. Fault-tolerable executions of a mobile agent at each NE

4 Implementation

The proposed scheme is implemented in the JAMAS, a model of mobile agent system. The JAMAS consists of Graphic User Interface, Agents Mobile Service Component, Agents Execution Environment Component, and Agents Repository to provide the naming transparency of agents. In addition, it may be executing one more systems within a host.

We show to launch through the process of an agent which manages some NEs (Internet elements). The following figures show that the simple agent as a role of Management Information Base (MIB) browser should be migrated and

executed according to the routing schedule. Figure 4 depicts the routing path of the agent such as NEh → NEb → NEa → NEc, and we assume to be a fault at the client NEb.

Finally, Figure 5 shows executions of the agent at each NE as follows: Figure 5(a), as a screen capture of the host NEh, shows hopping by a failure of connection at the next NEb after the launched agent normally progresses. Due to a failure of the host, the agent passes to next one. Thereafter, Figure 5(b), (c) capture executing of the agent at the hosts NEa, NEc. Then it is adapted to the proposed policy. Therefore, the agent has toured for all nodes having no faults before that it does re-connect with the faulty nodes.

Therefore, the adaptive migration scheme for mobile agents ensures the persistency of computation to preserve autonomous mobility and information of state for agents though there are some faults of nodes or clients on the routing schedules of the Internet.

5 Conclusions

In this paper we introduce the path reordering algorithm and backward recovery algorithm to ensure the migration of mobile agents in wired/wireless networks. The proposed scheme not only affords to avoid any faults of nodes or clients of mobile agents on Internet but also affects to agents' life span. All presented techniques have been implemented in our system, JAMAS. Therefore, JAMAS can improve effectively the problem of performance and wired/wireless networks overhead due to the imposed characteristics of distributed architecture since a mobile agent offers not only the migration reliability and transparency for mobile agent as autonomously as possible but also computing environment which is capable of distributed processing with mobile objects.

References

1. K.A. Baharat, L. Cardelli, "Migratory Applications", Proc. of the 8th Annual ACM Symp. on UISTech., November 1995.
2. J.Baumann, "A Protocol for Orphan Detection and Termination in Mobile Agent Systems", TR-1997-09, Stuttgart Univ. Jul., 1997.
3. General Magic, "Odyssey", URL: http://www.genmagic.com/agents/
4. IBM, "The Aglets Workbench", URL: http: //www.trl.ibm.co.jp/aglets
5. Ing-Ray Chen, Tsong-Min Chen, and Chiang Lee, "Analysis and Comparison of Location Strategies for Reducing Registration Cost in CS Networks", Wireless Personal Communications journal, Vol. 12, No. 2, 2000, pp. 117–136.
6. D. B. Lange, M. Oshima, "Seven good reasons for mobile agents", Proc. of CACM, Vol. 42(3), Mar. 1999, pp. 88–89.
7. Objectspace Voyager, GeneralMagic Odyssey, IBM Aglets: A Comparison, June, 1997.
8. OMG, "Mobile Agent Facility Interoperability Facilities Specification (MAF)", OMG.

9. A. Puliafito et al., "A Java-based Distributed Network Management Architecture", 3rd Int'l Conf. on Computer Science and Informatics (CS&I'97), Mar. 1997.
10. Robert S.G., "AgentTCL: A flexible and secure mobile-agent system", TR98-327, Dartmouth Col. June 1997.
11. K. Rothermel, M. Straer, "A Fault-Tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents", Proc. of 17th IEEE SRDS'98, Oct. 1998.
12. J. Vitek and C. Tschudin, "Mobile Object Systems: Towards the Programmable Internet", Springer-Verlag, Apr. 1997.

OpenMP in the Field: Anecdotes from Practice

Russell K. Standish[1,2], Clinton Chee[2], and Nils Smeds[3]

[1] School of Mathematics
[2] High Performance Computing Support Unit
University of New South Wales,Sydney, 2052,Australia
R.Standish@unsw.edu.au, chee@hpc.unsw.edu.au
http://www.hpc.unsw.edu.au
[3] Center for Parallel Computers, Kungl Tekniska Högskolan, 10044 Stockholm, Sweden
smeds@pdc.kth.se, http://www.pdc.kth.se/

Abstract. The High Performance Computing Support Unit at UNSW has a mission to support and encourage the scaling of computationally intensive applications from existing desktop implementations. OpenMP is a good match for this task. This paper reports on several projects in which OpenMP was used to parallelise an application, sometimes successfully, sometimes not so. The interest in these cases is that they are not the usual run of the mill applications that can be parallelised by simply adding a few OpenMP compiler directives, but required some lateral thinking.

1 Introduction

The High Performance Computing Support Unit at UNSW has a mission to support and encourage the scaling of computationally intensive applications from existing desktop implementations. Whilst some of our applications require the extreme performance that can only be achieved through distributed memory programming with MPI[7], for the most part our applications just need to run up to an order of magnitude faster than the desktop implementation. For this reason, we will often employ OpenMP[2] implementations with limits to scalability, but is an order of magnitude or so easier to program than an MPI implementation.

The HPCSU has a long history with OpenMP, in that an SMP parallel programming course developed during 1998 used the just released Fortran specification as a *lingua franca* to teach SMP programming. A table of equivalent compiler directives was provided for students to be able to program on their platform of choice, which at that stage did not support OpenMP. Nowadays, OpenMP support is ubiquitous, and this is no longer necessary. The course materials and tutorial examples are available from http://nswcpc.pvl.edu.au/SMP-course.html. It should be noted that this is not a self-study course, and needs to be presented by an experienced practitioner of HPC techniques.

The HPCSU supports customers of the *Australian Centre for Advanced Computing and Communications* (ac3), which has a 64 processor SGI Origin, a 68

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 637–647, 2003.
© Springer-Verlag Berlin Heidelberg 2003

processor IBM SP2 and a 2 processor NEC. As well as this, ac3 users may also use the *Australian Partnership for Advanced Computing*'s Compaq SC system. Both the SP2 and the SC are cluster systems based around quad processor nodes, and are generally not targets for OpenMP optimisation. An exception to this might occur if the application does not scale past 3-4 threads, in which case the Compaq SC with it's fast individual processors is an attractive option for running an OpenMP application within a node.

Since we have SGI Irix systems, we use an all in one development tool called ProDev Workshop. This is a GUI tool for debugging, profiling and analysing the results of parallelisation, included as part of the standard Irix development environment. We trialled a copy of KAPPro toolset, which is probably its main competitor, but ProDev Workshop appeared to provide superior information, and didn't require us to pay an expensive software license.

The 5 case studies presented here are not the usual sort of case studies one sees on OpenMP. After all, OpenMP's biggest attraction is to gain the advantages of parallel processing at minimal programming effort. These case studies involved considerable intellectual effort, and hopefully the lessons learnt are of interest to OpenMP practitioners.

2 NMG — Spectral Magnetohydrodynamics Code

The NMG code was originally written for the CM5 by Olga Podvigina. It solves 3D Navier-Stokes equations for an electrically conducting fluid. Similar work modeling neutral fluids was published in [5]. It is a spectral code, computing a quantity like $\mathbf{v} \times \nabla \times \mathbf{v}$. This can be expressed using the Fourier transform $\tilde{\cdot}$ as $\widetilde{\mathbf{v} \times i\mathbf{k} \times \tilde{\mathbf{v}}}$. So we have 3 3D FFTs, followed by a cross product, followed by 3 inverse 3D FFTs followed by another cross product.

Initially the code was run on the 128 node CM5 "colossus", which was still in operation at the time at the University of Adelaide, to obtain a baseline result for the porting and optimisation efforts. A 64^3 element volume was simulated for 50 timesteps, which took 230s to run on the CM5. This corresponds to about 130MFlops of sustained performance.

The code was written in CMFortran, a forerunner of the HPF language. Being already written in a data parallel fashion, it seemed that this should be an easy code to parallelise with OpenMP. The first task was to port the FFT routines to available FFT libraries. The following libraries were used:

SGI Complib. The original mathematics library available for Irix. A parallel implementation using Irix sproc() is available.

SGI Cray Scientific Library (SCSL). A library available for Irix, originally sourced from Cray. A parallel implementation using Irix sproc() is available.

FFTW. the *Fastest Fourier Transform in the West*[4], an open source library available from http://www.fftw.org/. A parallel implementation using POSIX threads is available.

IBM ESSL. Available for IBM SP systems. Threaded versions available.

Compaq DXML. Available for Compaq systems. Threaded versions available.
JMFFT. Open source FFT library suitable for vector computers, available from http://www.idris.fr/data/publications/JMFFT[3]

The CM5 Scientific Subroutine Library (CMSSL) provided FFT routines that perform an m-dimensional FFT on n-dimensional data. Ranks and dimensions of each array can be extracted from the arguments passed, and an additional rank 1 array specifies which dimensions are to be transformed. In NMG's case, a 3D FFT is performed on 4D data (since each field has 3 components). In converting the code to using the 3D FFT routines in the above libraries, array sections need to be created:

```
do i=1,3
  call zfft3d(-1,n,n,n,fff(i,:,:,:),n,n,coef)
enddo
```

Creating these array sections is actually a quite significant component of the total runtime. One of the strategies employed was to have separate variables for each component of the field (eg fff1, fff2 and fff3), and unroll the above loop.

The cross products were originally written out in full using as:

```
h(1,:,:,:)=ider(2,:,:,:)*vf(3,:,:,:)-ider(3,:,:,:)*vf(2,:,:,:)
h(2,:,:,:)=ider(3,:,:,:)*vf(1,:,:,:)-ider(1,:,:,:)*vf(3,:,:,:)
h(3,:,:,:)=ider(1,:,:,:)*vf(2,:,:,:)-ider(2,:,:,:)*vf(1,:,:,:)
```

Since none of these arrays fitted into cache, and there is no cache reuse, performance of this loop limps along at the speed of the memory subsystem. A slight improvement can be had by performing loop fusion:

```
do k=1,n
  do j=1,n
    do i=1,n
      h(1,i,j,k)=ider(2,i,j,k)*vf(3,i,j,k)-ider(3,i,j,k)*vf(2,i,j,k)
      h(2,i,j,k)=ider(3,i,j,k)*vf(1,i,j,k)-ider(1,i,j,k)*vf(3,i,j,k)
      h(3,i,j,k)=ider(1,i,j,k)*vf(2,i,j,k)-ider(2,i,j,k)*vf(1,i,j,k)
    ...
```

This at least allows some cache reuse. Another improvement comes from replacing ider by its value computed on the spot:

```
h(1,i,j,k)=(0,j)*vf(3,i,j,k)-(0,k)*vf(2,i,j,k)
```

or even extracting real and imaginary components of vf.

None of the FFT library routines would execute in parallel if called from within a parallel region, so this required two parallel regions per timestep. What is even worse is that on Irix systems, POSIX threads is incompatible with the SGI sproc() call used by the OpenMP compilers. As a consequence, FFTW was not useful for parallel work on Irix.

An optimisation which we didn't end up trying due to limited time is to write our own 3D (or 4D even) specialised FFT routine, composed of 1D sequential

FFT routines (from any library), and parallelised using OpenMP, so it could be called from a parallel region. Even so, timing results indicate that the cross product code is actually the bottleneck, executing at a small fraction of peak speed.

As a result we turned to using the NEC SX5 computer, which has a superior memory subsystem. We found an FFT library written by France's *Institut du Développement et des Ressources en Informatique Scientifique* (Institute for Development and Resources in Computational Science) that gave good performance on the SX5. Profiling the code on the SX5 indicated that the FFT is the bottleneck on this system. Execution time results of the sample problem for the different systems are given in the following table:

Machine	No. CPUs	Time (s)
CM5	128	230
SGI Power Challenge (195MHz)	4	188
SGI Origin (400MHz)	1	190
Intel PIII (600MHz)	1	232
NEC SX5	1	42

One thing that is interesting to note is that the newer Origin system does not run any faster in parallel, and is no faster than the older Power Challenge system whose performance peaks at 4 processors. It is expected that this is because the code is ultimately limited by the performance of the memory subsystem. It is also interesting to note a year 2000 vintage PC performing as well as the original CM5.

In conclusion, even though efficient shared memory versions of FFT routines exist, their use is limited by the performance of the memory subsystem in existing SMP implementations.

3 Force Calculations in *N*-Body Problems

N-body simulations are dominated by the force calculation, which scales as N^2, where the rest of the computation, and memory usage scales linearly with N. One of the example problems developed for the SMP programming course features implementing a 2D N-body force calculation for an inverse square force (like gravity):

$$\mathbf{F}_j = \sum_{k \neq j} \frac{\mathbf{x}_j - \mathbf{x}_k}{|\mathbf{x}_j - \mathbf{x}_k|^3} \tag{1}$$

Fortran code for computing eq (1) is shown below:.

```
do j=2,N
   do k=1,j-1
      rx = x(j)-x(k); ry = y(j)-y(k)
      r=sqrt(rx*rx + ry*ry); ir3 = 1/(r*r*r)
      Fx(j) = Fx(j) + rx * ir3
      Fy(j) = Fy(j) + ry * ir3
      Fx(k) = Fx(k) - rx * ir3
      Fy(k) = Fy(k) - rx * ir3
   enddo
enddo
```

On traditional vector supercomputers, the inner loop can be effectively vectorised (The carried dependency on $\mathbf{F}_j$ can be simply resolved as a sum reduction, and usually compilers will do this automatically). However, the operation density of the inner loop is too low on SMP systems for practical values of N to overcome the thread creation overhead, and the flushing of the caches. So parallelising the outer loop becomes mandatory.

The first problem is to break the carried dependency on $\mathbf{F}_k$. This can be achieved by placing the last two assignment statements within a `CRITICAL` region, however the thread lock overhead absolutely kills any performance gain from parallelism. The answer to the problem comes from realising that total force computation is decomposable into independent pieces that can be summed up to give the final force. So we need to introduce a thread private force variable, which will remain in a processor's cache (bear in mind the linear dependence on N of memory requirements). Once the force has been computed between pairs of particles, for which one of the pair is located on the processor, the result can be summed into the shared force variable, within a `CRITICAL` section, as per the code snippet below.

Only one other feature needs commenting. The default STATIC scheduling of the DO workshare construct will lead to load imbalance due to the triangular work distribution. However, as is reasonably well known, OpenMP provides the GUIDED schedule which is optimised for this situation.

Performance of this particular piece of code was reported elsewhere[8], exceeding 16GFlops for 20,000 particles on a 40 processor SGI Origin. This exceeds 50% of peak floating point performance (each CPU has a peak of 800MFlops).

Exciting though this result is, it is worth noting that for more than a few thousand particles, one would employ a neighbourlist algorithm if the force was short range, or an Ewald summation for long range forces [1]. This changes the overall algorithm complexity to something more like $o(N \log N)$, and probably reduces the amount of gain from parallelism.

This technique was employed with a real world 3D *Discrete Particle* code used for Material Science research. These particles had van der Waals attraction, and the algorithm employed a neighbourlist to imporve performance. We achieved reasonable performance scaling to about 8 processors (about 4 times) for 250 particles.

```
C$OMP PARALLEL private(i,j,k,rx,ry,r,ir3,Fxk,Fyk) shared(x,y,Fx,Fy)
         do j=1,N
            Fxk(j)=0
            ...
C$OMP DO schedule(guided)
         do j=1,N
            Fx(j)=0
            Fy(j)=0
            do k=1,j-1
               ...
               Fxk(j) = Fxk(j) + rx * ir3
               ...
C$OMP CRITICAL
         do j=1,N
            Fx(j) = Fx(j) + Fxk(j)
            ...
C$OMP END CRITICAL
C$OMP END PARALLEL
```

4 HKondo — Lattice Gauge Theory Calculations

HKondo was a code presented by one of our users (who shall remain nameless). It performed some kind of lattice gauge calculation, which is not important for the present purposes. What is important is that the user desired to speed the program up, and that the code consisted of a monolithic block of 670 lines of Fortran 66 dialect and coding style. No indentation, no comments and a large number of goto statements. A large number of optimisations had already been applied to the code, which in part explained the codes crypticity. There was little to be achieved by trying standard sequential optimisation techniques.

An initial profile of the code indicated that most time was spent within two inner loops, however the loop count on these were depressingly small. The next loop out was then the loop to attempt parallelisation. However this loop of about 200 lines has dozens of array references and assorted other control logic. The compiler's automatic paralleliser was of no assistance here, the code was simply too complex.

In order to determine the data flow through this block, an unusual trick was resorted to. The entire loop was moved into its own subroutine, and used the `IMPLICIT NONE` directive. The first pass of the compiler gave me a list of variables that needed to be declared. The compiler listing were pasted into the code, and reformatted as declarations. Next, the parameter list for the subroutine needed to be determined. This involved a manual check as to whether a variable name had been referenced outside the subroutine using the text editor's search feature. It soon became clear whether a variable was imported into, exported from, or simply ignored by the code block. The conclusions could be tested by using Fortran's `INTENT` statement, and leaving the variable out of the parameter list. If a conclusion was wrong, the program would break. This led to a small

list of 35 variables that needed to be declared as SHARED, with the remainder declared as thread private. This was the most labour intensive part of the whole process, and took perhaps about 2 hours to perform.

After this was done, the compiler's paralleliser indicated 3 carried dependencies. The first two were of the nature of tracking maximum values, eg

```
if (nst2.gt.nlmax) nlmax=nst2
```

which can be easily handled by placing these in a critical region. The third carried dependency was a little more difficult:

```
afr(1,k1,k2,k3)=afr(1,k1,k2,k3)+ax
```

These might be handled via a critical section, but unfortunately were inside the innermost loop, and consequently the most expensive operation. The critical section idea lost in lock overheads, all the gains made by parallelisation.

A closer inspection of the code revealed that k1 was constant within the inner loop, and depended in a deterministic way on the loop index of the outer loop:

```
do ist=1,nst
 k1=v(ist); nt=...
 do j=1,nt
   k2=...; k3=...; ax=...
   afr(1,k1,k2,k3)=afr(1,k1,k2,k3)+ax
   ...
```

What needed to be done, therefore, was to gather all loop iterations with the same value of k1, and execute this on the same processor. Then the loop dependence carried by afr is simply broken without needing a critical section. To do this required a small precomputation:

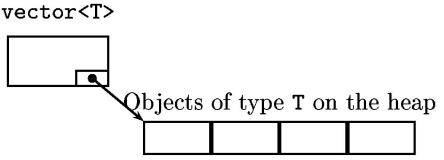
```
      integer istvec(maxk1,nst), nist(maxk1)
      do ist=1,nst
        k1=v(ist)
        nist(k1)=nist(k1)+1
        istvec(k1,nist(k1))=ist
      enddo
C$OMP PARALLEL DO SCHEDULE(DYNAMIC)
      do k1=1,maxk1
        do i=1,nist(k1)
          ist=istvec(k1,i); nt=...
          do j=1,nt
            k2=...; k3=...; ax=...
            afr(1,k2,k3,k1)=afr(1,k2,k3,k1)+ax
          ...
```

Note that the indices of `afr` needed to be reordered to make the memory references more cache friendly. DYNAMIC scheduling was used to improve load balancing. The resultant code achieved a speedup of approximately 3 on 4 threads, which was about the limit of its scalability. The quad processor APAC nodes proved the most effective platform in this case.

5 C++ Implementation of Ray Tracing

As a student project for an honours course taught by one of us, a student Tom Edlund decided to implement a parallel Monte Carlo ray tracing algorithm, choosing C++ for its object oriented nature, and used the VTK toolkit[6] for handling scene geometries.

Whilst the exact details of the algorithm are not germane here, the main issues with OpenMP related to the use of standard containers, in particular `vector`. A `vector` is a dynamically resizable array of some type of object. Clearly, it must consist of some kind of header for managing the object, with the actual storage taking place on the heap:



Firstly, lets consider what happens when the vector is a shared object. Vectors are very flexible objects that can be built by adding objects to their end with `push_back()`, resized, or have elements deleted from with their range. Any of these operations can caused a new space to be allocated for the vector, and the contained objects copied into the new space. If another thread is trying to access the vector, even just read only access, while the vector is being copied to a new location, disaster strikes. Even though the Standard Template Library claims its containers to be thread safe, they can only be used safely if treated like static arrays — namely their size remains fixed throughout a parallel region, and they are either readonly, or written to from within a critical region.

Now consider thread private vectors. As soon as a vector is resized to contain some data, it must call `new` to obtain some memory. The problem is that `new` must update some shared internal heap management variables, as the heap is shared between all threads. This problem is fixed by providing a thread safe version of `new`, which is easy enough to do in C++:

```
void *operator new(size_t s)
{
#pragma omp critical
  return malloc(s);
}

void operator delete(void *p)
{
#pragma omp critical
  free(p);
}
```

Obviously these critical statements can be a serious performance bottleneck, but by judiciously using `vector`'s `reserve()` method, one can minimise the number of calls to `new` and `delete`. Alternatively, one could set up private heaps for each thread, and supply appropriate `new` and `delete` operators to manage them.

Tom found that even with these thread safe memory allocation operators, his code did not run correctly in parallel. In the end, he had to write his own threadsafe vector class, which did work. This could be simply a problem with the SGI implementation of the standard template library, as we did not try this out on any other C++ implementation.

6 Finite Difference Time Domain

The FDTD code uses a Finite Difference Time Domain technique to find the solution of a vector field problem.

The porting and optimisation work was performed on the IBM SP2, then laterported to the NEC SX5 vector machine. The same OpenMP parallel techniques used on the IBM SP2 version, was employed the dual processor SX5 using the native NEC SMP parallellising compiler directives.

The first of the two main sections to be parallelized is shown below; the second section being similar in structure to the first but with slightly more complex terms. For relatively simple constructs as above, the FORALL multi-loop structure is fairly efficient for it is an optimized command. Initial testing which replaced the FORALL with a triple do-loop, slowed down this section by 3.5 times.

```
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gxy(I,J,K) = Cay(J)*Gxy(I,J,K) - Cby(J)*
( Fzx(I,J+1,K) - Fzx(I,J,K) +  Fzy(I,J+1,K) - Fzy(I,J,K) )
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gxz(I,J,K) = Caz(K)*Gxz(I,J,K) + Cbz(K)*
( Fyx(I,J,K+1) - Fyx(I,J,K) +  Fyz(I,J,K+1) - Fyz(I,J,K) )
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gyx(I,J,K) = Cax(I)*Gyx(I,J,K) + Cbx(I)*
( Fzx(I+1,J,K) - Fzx(I,J,K) +  Fzy(I+1,J,K) - Fzy(I,J,K) )
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gyz(I,J,K) = Caz(K)*Gyz(I,J,K) - Cbz(K)*
( Fxy(I,J,K+1) - Fxy(I,J,K) +  Fxz(I,J,K+1) - Fxz(I,J,K) )
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gzx(I,J,K) = Cax(I)*Gzx(I,J,K) - Cbx(I)*
( Fyx(I+1,J,K) - Fyx(I,J,K) +  Fyz(I+1,J,K) - Fyz(I,J,K) )
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gzy(I,J,K) = Cay(J)*Gzy(I,J,K) + Cby(J)*
( Fxy(I,J+1,K) - Fxy(I,J,K) +  Fxz(I,J+1,K) - Fxz(I,J,K) )
```

Various features of the code snippet above indicate that each of the six loops are fairly independent and there is no data dependency. There is a possibility of cache reuse which can be accomplished by loop fusion. Looking at the big picture, it appears that this code would benefit from coarse-grain parallelism, rather than fine-grain / loop level parallelism.

Thus the major step to parallelize the code is to use the parallel sections feature of OpenMP. There are a number of ways in which to construct parallel sections over this code, some will be more efficient than others. In order to achieve high efficiency, the first criteria is maximizing cache reuse, and the second criteria is matching the number of parallel sections with a multiple of the number of processors available.

The optimized code is shown below here with a 2-2-2 configuration denoting 3 parallel sections with 2 equations each:

```
!$OMP PARALLEL SECTIONS
      FORALL(I=0:IM,J=0:JM,K=0:KM)
Gxy(I,J,K) = Cay(J)*Gxy(I,J,K) - Cby(J)*
( Fzx(I,J+1,K) - Fzx(I,J,K) +  Fzy(I,J+1,K) - Fzy(I,J,K) )
Gyx(I,J,K) = Cax(I)*Gyx(I,J,K) + Cbx(I)*
( Fzx(I+1,J,K) - Fzx(I,J,K) +  Fzy(I+1,J,K) - Fzy(I,J,K) )
      end FORALL
!$OMP SECTION
   ...
!$OMP SECTION
   ...
!$OMP END PARALLEL SECTIONS
```

The 2-2-2 configuration take advantage of the symmetry of the pairing of the 6 field equations — to allow some cache reuse. The resultant 3 parallel sections is also sufficient to be implemented on the IBM SP2 by using 1 node. Within this node, 4 processors are available in a shared memory configuration. An OpenMP thread will ideally be processed by one processor. In this case, 3 OpenMP threads are required - specified by the environment variable *OMP_NUM_THREADS=3*. In addition, experience has shown that the computing job runs better when all 4 processors of the node are reserved by the batch queue, as the 4th processor takes over some of the system processing time.

Config.	Loop Type	OMP thrd.	Real Time	User Time	Sys Time
2-2-2	FORALL	1	1m4.78s	1m3.23s	0m0.45s
2-2-2	FORALL	2	0m44.69s	1m6.22s	0m0.47s
2-2-2	FORALL	3	0m28.25s	1m11.64s	0m3.20s
2-2-2	DO	3	0m30.80s	1m19.27s	0m3.71s
1-1-1-1-1-1	DO	3	0m32.60s	1m22.41s	0m3.92s

The results when the remainder of the code was parallelized is shown in table above. The size of the matrices are IM=120, JM=80, KM=120 for 100 iterations.

There is a clear improvement as the number of OMP threads are increased from 1 to 3. As noted before, changing the FORALL loops into triple Do-loops slows down the code.

The code was finally ported over to the SX5 vector machine, and after some manual optimization, ran 10 times faster (using 2 threads) than on the SP2. However, this was not only due to the vectorization. In fact the same principles of parallel sections was applied using compiler directives (very similar to OpenMP) native to the SX compilers. Thus the techniques of OpenMP can sometimes be transferred successfully to other non-OpenMP compilers.

References

1. M. P. Allen and D. Tildesley. *Computer Simulation of Liquids*. Clarendon, Oxford, 1987.
2. Rohit Chandra. *Parallel Programming in OpenMP*. Morgan Kauffman, San Francisco, CA, 2001.
3. Jalel Chergui. Transformées de Fourier rapides monoprocesseur sur NEC SX-5. Technical report, CNRS/IDRIS, Bâtiment 506, BP 167 91403 Orsay cedex, France, 2000. http://www.idris.fr/data/publications/fft-SX5.pdf.
4. M. Frigo and S. G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proceedings ICASSP*, volume 3, pages 1381–1384, 1998. http://www.fftw.org/.
5. Olga Podvigina. Spatially-periodic steady solutions to the three-dimensional Navier-Stokes equation with the ABC force. *Physica D*, 128:250–272, 1999.
6. Will Schroeder, Ken Martin, and Bill Lorensen. *The visualization toolkit : an object-oriented approach to 3-D graphics*. Prentice Hall, Upper Saddle River, N.J., 1996.
7. Marc Snir et al. *MPI: the complete reference*. MIT Press, Cambridge, MA, 1996.
8. R.K. Standish. SMP vs Vector: a head-to-head comparison. In *Proceedings HPCAsia 2001.*, 2001. http://parallel.hpc.unsw.edu.au/rks/docs/ps/smp-vs-vector.ps.gz.

OpenMP and NUMA Architectures I: Investigating Memory Placement on the SGI Origin 3000

Nathan Robertson and Alistair Rendell

Department of Computer Science
Australian National University
Canberra ACT0200, Australia
Nathanr@nathanr.net, Alistair.Rendell@anu.edu.au

Abstract. The OpenMP programming model is based upon the assumption of uniform memory access. Virtually all current day large scale shared memory computers exhibit some degree of Non-Uniform Memory Access (NUMA). Should OpenMP be extended for NUMA architectures? This paper aims to quantify NUMA effects on the SGI Origin 3000 system as a prelude to answering this important question. We discuss the tools required to study NUMA effects and use them in the study of latency, bandwidth and the solution of a 2-D heat diffusion problem.

1 Introduction

In the late 1980's a number of hardware vendors began offering small scale shared memory parallel architectures comprising a few CPU and memory boards linked via a shared bus. These systems quickly proved to be very popular not least because the cost premium of adding CPUs was relatively low compared to the cost of the initial system. With this new architecture and an expanding user base came a natural interest to develop applications that could run in parallel, and this was particularly true among the computational science community.

Although there are several ways to utilize multiple CPUs on a shared memory architecture, the idea of parallelizing critical loops by adding a few directives to the source code and letting the compiler do most of the work was particularly attractive. This was in part because computational scientists had used a similar approach to successfully exploit vector architectures. To facilitate this, hardware vendors began implementing and promoting their own set of shared memory parallel programming compiler directives. Unfortunately, although many of these directives provided virtually identical functionality their exact form was vendor specific. This situation existed for a number of years, until in the mid 1990's there was a move to develop a standard set of shared memory parallel processing directives that eventually gave rise to OpenMP for Fortran 77 [1] (released in 1997). The success of this standardization effort has since led to a number of updates and extensions such that the OpenMP model now encompasses Fortran 95, C and C++ [1].

A basic premise of OpenMP is a shared memory programming model in which all CPUs have equal access to all memory locations, i.e. the assumption of Uniform Memory Access (UMA). While this accurately reflected the early bus based shared

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 648–656, 2003.
© Springer-Verlag Berlin Heidelberg 2003

memory machines of the late 1980's and early 1990's, it is increasingly not true for large scale shared memory parallel architectures. Thus, virtually all current large scale parallel machines have some degree of Non Uniform Memory Access (NUMA), e.g. the SGI Origin 3000, HP GS320, Sun E15k and IBM Regatta.

For computational science the shared memory parallel programming paradigm has some definite attractions, especially now that there is widespread compiler support for OpenMP. For many of these applications, however, performance is also critical and an obvious question arises as to whether the existing UMA OpenMP programming model should be extended to account for current NUMA architectures, and if so how. There is also interest in developing OpenMP compilers that generate code for cluster systems [2,3] – which is an extreme NUMA environment!

NUMA support within OpenMP has received some discussion both in the literature and at various conference forums [4,5]. There are essentially two opposing views. At one extreme it is considered that issues relating to memory placement should be handled by the O/S through sensible page placement and migration. While at the other extreme it is advocated that the user should be provided with full control over all memory and thread placement. In the middle there are a number of hybrid schemes, such as the user initiated page migration model proposed by Nicolopolous et al [6].

Already some vendors, e.g. SGI [7] and Compaq [8], have implemented their own NUMA extensions to OpenMP. Unfortunately, as in the early days of shared memory parallel systems, these have different syntax but similar functionality. For application programmers trying to develop portable programs this provides a major disincentive to their use. In our view, however, it is premature to be advocating any specific NUMA extensions to OpenMP. Rather we believe it is first important to gather information for a variety of applications showing clearly the benefits or otherwise of different memory placements on overall performance. That is, to quantify what is the maximum likely difference between the performance obtained when using optimal memory placement versus that obtained when using the worst possible memory placement, and then to compare both these results with the typical performance obtained when running with the default memory placement provided by the O/S.

This paper is a first step towards this goal. Firstly we discuss the basic functionality that is required to develop application programs that can be used to investigate NUMA effects. Then we use this functionality to measure the different memory latencies and bandwidths on a NUMA platform – an SGI Origin 3000 system [9,10]. Finally we investigate the implication of NUMA on the performance on the SGI Origin of a typical 2-D heat diffusion program.

2 Basic Requirements

Our goal is to place processes or threads on specific CPUs within the NUMA hardware and have their associated memory reside at specific physical locations. To achieve this requires the ability to:

- bind a particular process or thread to a specific CPU
- allocate memory pages to specific memory locations without page migration

For verification it is also useful to be able to:

- determine the physical CPU that a process/thread is currently running on
- map any given virtual address to its physical address

In general, in a multi-user shared-memory parallel environment having users tie their threads or memory to specific hardware resources is undesirable since it substantially reduces the flexibility of the O/S to efficiently schedule jobs. Thus it is hardly surprising that for a given hardware platform finding out how to achieve these goals is not always easy!

The results presented here were obtained from an SGI Origin 3000 system [9]. To create threads we used the `pthreads` library and then bound them to specific CPUs using SGI's non-standard `pthread_setrunon_np` library routine. Memory was allocate and placed at specific locations using the "`dplace`" utility. To verify thread placement it is necessary to first issue a call to `schedctl(SETHINTS)` before obtaining the CPU id from the `get_cpu` macro defined in the `task.h` header file. To map virtual addresses to physical addresses we used the `va2pa` routine available at the SGI technical publication web site [11], although it should be noted that the bits are shifted by 33 places on the Origin 3000 compared to 32 places on the Origin 2000.

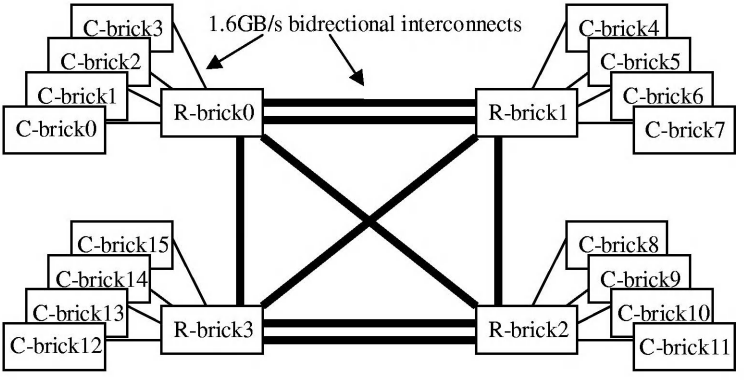


Fig. 1. Schematic diagram of the router and CPU bricks on the SGI Origin 3000 used in this work. (C-bricks contain 4 600MHz R14K CPUs and 4GBytes of memory).

The physical configuration of the SGI 3000 used in this work is given in Fig.1. It has 64 processors and 64GB of memory. Each CPU board, or C-brick, comprises 4 CPUs and 4GBytes of memory. The C-bricks are connected to router or R-bricks that are then connected together in a "bristled" hypercube topology. There are 8 connections to each R-brick and each is capable of a peak performance of 1.6GB/s bidirectional. On the machine used for this work the spare R-brick connections that would normally be used to build a higher dimensional hypercubes have been used to provided enhanced bandwidth between selected R-bricks. The configuration shown in Fig.1 has three different levels of NUMA, corresponding to the following memory accesses:

1. within the same C-brick, e.g. C-brick0 to C-brick0
2. one R-brick remove, e.g. C-brick0 to C-brick1
3. two R-bricks removed, e.g. C-brick0 to C-brick11

3 Measuring Memory Latency on NUMA Architectures

The aim here is to measure the memory latency associated with each different class of memory access. To do this we have developed a modified version of LMBench [12]. The basic idea behind LMBench is pointer chasing, i.e. a memory location is read and the contents used as a pointer to the location of the next memory address to be read. By designing each memory reference to be sufficiently well separated from all previous references it is possible to ensure that all references are to main memory and not to data that already resides in cache.

To achieve the above goal LMBench was modified to use `pthreads`. First a thread is created and bound to a specific CPU. This thread then allocates and initializes a large block of memory on the same C-brick (using `dplace`). The first element of this array is defined as a pointer referencing an element N_{sep} bytes further along in the array. This element in turns references another element N_{sep} bytes further advanced. This pattern continues until after some point a null pointer is used to terminate the sequence. After establishing this pointer series, the address of every element in the array is examined to verify that it is indeed located on the expected C-brick. The thread is then moved and bound to a CPU on a different C-brick, the pointer references followed, and the elapsed time taken recorded. (Note that we initialize the thread on one C-brick and then measure the pointer chasing time on another to ensure that that there are no caching effects resulting from the array setup phase). When measurement is complete on this C-brick the thread is moved to another C-brick and process repeated. A brief summary is given below:

```
Create thread and bind to specific CPU on given c-brick
Allocate memory on same c-brick and set up pointer array
Move thread to new c-brick
foreach c-brick in system
    bind thread to that c-brick
    pointer = initial_pointer
    start timer
    while pointer != NULL
        pointer = *pointer
    stop timer
    print c-brick and time
Next c-brick
```

The benchmark contains two parameters, the number of bytes separating each pointer reference (N_{sep}) and the total number of pointer references (N_{ref}). The results for N_{ref}=1000 and N_{sep}=16kbytes are given in Fig. 2. It is immediately obvious from this that the memory was located on C-brick 2. Accessing this memory from the C-bricks

connected to the same R-brick (i.e. C-bricks 0, 1 and 3) incurs a latency penalty of ≈ 50%, while those located two R-bricks away have roughly twice the latency.

Overall the results shown in Fig. 1 tend to indicate that for memory latency critical applications there is potentially a factor of two to be gained through careful placement of thread and memory.

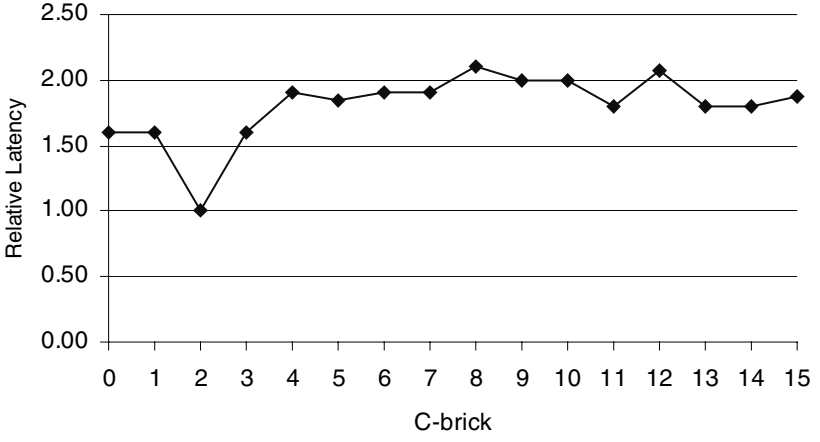


Fig. 2. Relative memory latency for accessing memory located on C-brick2 from all C-bricks. Results obtained on a non-dedicated Origin 3000

4 Measuring Bandwidth on NUMA Architectures

The aim here was to measure memory bandwidth contentions that may arise from having multiple threads simultaneously accessing various memory locations. To do this we have used a modified version of the STREAM benchmark [13]. This benchmark measures the time required to perform the following four vector operations:

```
A(1:N) = B(1:N)                          copy
A(1:N) = B(1:N) * const                  scale
A(1:N) = B(1:N) + C(1:N)                 add
A(1:N) = B(1:N) + C(1:N) * const         triad
```

for some large value of `N`. The results are reported in terms of Mbytes/sec, representing the amount of data that is transferred between memory and the CPU.

The basic STREAM benchmark was modified to create multiple threads and multiple sets of the A, B and C vectors. Each set of vectors was allocated entirely within the memory of a specific C-brick and was used by a unique thread. Treads

were then selectively bound to different C-bricks and synchronized before and after they had completed each vector operation. The results obtained with up to two threads and two different memory locations are given in Table 1.

Table 1. Memory bandwidth for Scale and Triad operations with explicit thread and memory placement for up to 2 threads/memory pairs. Results were obtained from a non-dedicated 64 SGI O3K with 600MHz processors

Exp. No.	— C-brick Pair 1— Thread	Memory	— C-brick Pair 2 — Thread	Memory	Scale Mbytes/Sec	Triad Mbytes/Sec
1-thread						
1.1	0	0	-	-	676	637
1.2	0	1	-	-	556	533
1.3	0	9	-	-	433	458
2-threads						
2.1	0	0	0	0	789	800
2.2	0	0	1	1	1196	1260
2.3	0	1	1	0	1044	1083
2.4	9	0	9	0	750	736
2.5	9	0	10	1	881	870

Although the results were obtained from a non-dedicated machine, we report the best observed performance from a number of experiments and believe the trends to be valid. Thus with a single thread we see maximum performance from experiment number 1.1 – corresponding to co-location of the thread and memory. Separating the thread and memory by a single R-brick decreases bandwidth (result 1.2) by about 20%. This decreases still further when the thread and memory are separated by two R-bricks (result 1.3). Since the interconnect can support data rates well in excess of the results obtained above, it seems likely that at least part of this bandwidth reduction may be due to contention on the network arising from the other jobs running at the time.

Considering now the results for two threads, if these threads plus their memory are all located on the same C-brick we see only marginal improvement in bandwidth compared to the single thread results, i.e. result 2.1 compared to result 1.1. Moving the threads and memory to different C-bricks (result 2.2) gives a significantly enhanced bandwidth of nearly twice that of a single thread (result 1.1). Interestingly having each thread reference memory on the C-brick of the competing thread (result 2.3) produces nearly the same bandwidth as when each thread is accessing local memory (result 2.2). In experiment 2.4 the threads were both located on the same C-brick but at the opposite side of the machine to their memory. This produces roughly the same bandwidth as having all threads and memory on the same C-brick (result 2.1). Finally having all threads and memory on different C-bricks is slightly better than having them all on one C-brick, i.e. result 2.5 compared to result 2.1.

In summary for two threads their appears to be the potential for the available memory bandwidth to vary by a factor of two between the best and worst possible thread and memory placement.

5 Memory Placement for 2D Heat Diffusion

The above results clearly show a difference between optimal and non-optimal memory and thread placement. Both cases are, however, extreme examples designed to illustrate these effects. We now consider a slightly more realistic problem – the solution of the Laplace equation on a uniform rectangular lattice with Dirichlet boundary conditions. A typical example of this occurs when calculating the steady state temperature distribution for a metal plate whose edges are held at a fixed temperature. The algorithm proceeds by discretizing the domain of the metal plate and then evaluating a new temperature for each grid point based on an average of the current temperatures of the four surround grid points. Normally the calculation would continue in an iterative fashion until some convergence criteria is reached, however in our benchmark we perform a fixed number of iterations. The algorithm is easily parallelized using a simple domain decomposition in which, for example, on two CPUs all the data points in one half of the domain are assigned to one thread while those in the other half are assigned to the other. Our parallelization was implemented using pthreads.

In a similar vein to the modified STREAM benchmark the threads and the memory corresponding to the grid point they will be updating were selectively placed on different C-bricks. In contrast to the simple STREAM benchmarks, however, updating the data points at the boundary between the two domains requires both threads to retrieve a small amount of data from the adjoining domain. The calculation was run for 25 iterations on a grid size of 1500 by 2048 elements. Results for two threads are given in Table 2. Although the benchmark was again run in a non-dedicated environment timing data was collected from a number of runs performed over a number of days. We report the best elapsed time as well as the average elapsed time and standard deviation. The thread and memory placements labeled as “IRIX” were obtained without using any explicit memory or thread placement.

The “Best” results show a difference of about 40% between the results obtained in experiment 2.2 compared with those from experiment 2.4. It is also not surprising, given the STREAM results, that the fastest performance was obtained when each thread was located on a different C-brick, and the arrays containing their grid points also resided on the same C-brick. It is also interesting that the results obtained without any explicit thread or memory placement are close to the worst. Suggesting that, at very least, a user who blindly uses this system without any thought to thread and data location is likely to achieve far from optimal performance.

Table 2. Effect of memory and thread placement for parallel 2-D heat diffusion benchmark using 2 threads. Results elapsed times obtained from a non-dedicated 64 SGI O3K with 600MHz processors

Exp. No.	— C-Brick Pair 1 — Thread	Memory	— C-Brick Pair 2 — Thread	Memory	Best Sec	Average Sec	Std. Dev.
2.1	0	0	0	0	3.2	3.8	0.63
2.2	0	0	1	1	3.1	3.7	0.49
2.3	0	1	1	0	4.0	4.1	0.08
2.4	9	0	9	0	4.6	5.0	0.45
2.5	9	0	9	1	4.4	5.4	0.94
2.6	IRIX	IRIX	IRIX	IRIX	4.3	4.7	0.23

6 Conclusions

We believe that quantitatively studying the effects of NUMA for a range of applications will provide valuable input into the debate on whether and how to extend the OpenMP paradigm to account for NUMA. This work is a first step in that direction. Undertaking NUMA studies requires some means to selectively bind threads or processes to specific hardware resources and to allocate memory at specific physical memory locations. One way to do this on the SGI Origin system has been discussed. Using the SGI we have developed benchmarks that can be used to accurately measure the differential memory latencies and the effects of bandwidth contentions. In both cases we have shown that the difference between good and bad memory/thread placement on the SGI Origin can be quite large (a factor of 2 or more). Moreover, our results have been obtained on a relatively small system with just 64 processors and three NUMA levels, going to a larger machine would be expected to significantly enhance these differences. The importance of these effects for a model application has been discussed. These results also show that simply relying on the O/S to perform wise thread and memory placement does not necessarily produce good results. Work is now in progress to extend these investigations to involve larger number of threads, more complicated algorithms and other NUMA architectures.

Acknowledgements. The authors are grateful to Martin Nichols and the SGI staff at the University of Queensland for their assistance in this work. The support of the Australian Partnership in Advanced Computing is gratefully acknowledged.

References

1. See http://www.openmp.org
2. A. Basumallik, S-J. Min and R. Eigenmann, *Towards OpenMP Execution on Software Distributed Shared Memory Systems*, pp 457–468, Int. Workshop on OpenMP: Experiences and Implementations (WOMPEI2002), Lecture Notes in Computer Science #2327, Springer Verlag, May 2002.
3. Omni OpeMP and SCASH. http://www.pccluster.org
4. D.S. Nikolopoulos, T.S. Papatheodorou, C.D. Polychronopoulos, J. Labarta and E. Ayguadé, *Is Data Distribution Necessary in OpenMP*, Proc. Supercomputing, Dallas, TX 2000, IEEE Press.
5. J. Bircsak, P. Craig, R. Crowell, Z. Cvetanovic, J. Harris, C.A. Nelson and C.D. Offner, *Extending OpenMP for NUMA Machines*, Proc. Supercomputing, Dallas TX 2000, IEEE Press.
6. D.S. Nikolopoulos, T.S. Papatheodorou, C.D. Polychronopoulos, J. Labarta and E. Ayguadé, *A Case for User-Level Dynamic Page Migration*, Proc. of the 14th ACM Int. Conf. on Supercomputing, pp 119–130. Santa Fe, NM, May 2000.
7. See Chapter 8: Tuning for Parallel Processing, Origin 2000 and Onyx2 Performance Tuning and Optimization Guide (document 007 3430 003), available at http://techpubs.sgi.com
8. Tru64 UNIX NUMA Overview, Part Number AA-NUMAG-DE, available at http://www.tru64unix.compaq.com/docs/base_doc/DOCUMENTATION/V51_HTML/NUMA/numa.pdf
9. See http://www.sgi.com/origin/3000/
10. J.R. Mashey, *NUMAflex Modular Design Approach – A revolution in Evolution*, http://www.cwi.nl/~robertl/mash/numaflex, posted in Aug. 2000 on comp.arc.news
11. See Appendix C, Useful Scripts and Code, Origin 2000 and Onyx2 Performance Tuning and Optimization Guide (document 007 3430 003), available at http://techpubs.sgi.com
12. L. McVoy, *LMBench – Tools for Performance Analysis*, http://www.bitmover.com/lmbench
13. J.D. McCalpin, *STREAM: Sustainable Memory Bandwidth in High Performance Computers*, http://www.cs.virginia.edu/stream

Poor Scalability of Parallel Shared Memory Model: Myth or Reality?

Mark Kremenetsky, Arthur Raefsky, and Steve Reinhardt

Supercomputer Applications, Silicon Graphics Inc.,
Mountain View, California 94043
{mdk,raefsky,spr}@sgi.com

Abstract. Large CFD models require memory sizes larger than can be supported by today's single 'node' computers. Using the memory of more than one node can greatly complicate the creation of a well-performing program. We believe that preserving globally addressable memory beyond the boundary of a single node enables diversity of programming methods and provides flexibility essential for optimum algorithm development. We isolate the effects of algorithm and implementation by porting the same parallel CFD algorithm in three styles: fine-grain shared memory, coarse-grain shared memory, and coarse-grain distributed memory.
Despite some fundamental differences in programming implementation, both distributed memory and coarse-grain shared memory code provide very close parallel performance due to algorithmic similarities. At the same time fine-grain shared memory code, despite use of the same programming paradigm as coarse-grain shared memory program, falls far behind due to unavoidable parallel performance penalties caused by Amdahl's law and some other limitations.

1 Introduction

It has long been recognized that the two necessary elements for achieving scalability in application performance are scalable hardware and software. Both these elements have existed for some time. Scalable hardware in the context of physically distributed memories connected through a scalable interconnection network has been commercially available since the 1980's. Such systems provide only an interconnection network and the burden of scalability then falls on software. As a result, scalable software for such systems will exist only in a message passing model.
There is another class of parallel architecture with scalable hardware support for cache coherence. These are generally referred to as Shared Memory Multiprocessor or SMP architectures. For SMP systems, the native programming model is shared memory and message passing is built on a top of shared memory model. This software environment provides multiple parallel programming methods and parallel paradigms.

This paper will demonstrate that the so-called "poor scalability" of shared memory programming paradigm is "myth" rather than "reality" and that the impediments to

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 657–666, 2003.
© Springer-Verlag Berlin Heidelberg 2003

software scalability are not the programming model but rather parallel application methods.

Within this context we implement a single key CFD algorithm in multiple styles (methods) to illustrate the benefits of a low-latency, high-bandwidth memory system available via multiple programming methods. The first style is classic OpenMP loop-level parallelism, the second is OpenMP directives used in a coarse-grain fashion with ghost cells, and the third is a distributed memory version implemented with MPI. We conclude that coarse-grain parallelism is required for strong scalability, and its implementation via a shared memory approach or distributed memory paradigm (MPI) provides similar parallel performance.

2 Parallel Programming Models (Paradigms)

This paper concerns itself with two different programming models, message passing and shared memory. The message passing model assumes the underlying parallel architecture is simply a collection of computers, or nodes, linked together by network. Each individual node knows nothing about the memory of other nodes in the system. The programmer must explicitly pass information between nodes by sending messages. In addition, a message cannot be received by a node unless that node explicitly posts a "receive". This last restriction is relaxed for so-called "one-sided" message passing as exemplified by Cray's shmem model or MPI-2 standard. One–sided message passing requires hardware support for good performance. In common two-sided message passing, both sending and receiving node must coordinate to successfully share data. Although each processor can execute a different program, most often the "Single Program, Multiple Data" (SPMD) programming style is used; all processors execute the same program, acting on different parts of data set. This requires an appropriate partitioning of the data and operations. The partitioning of the data must be done in such a manner that the workload is well balanced between processors, and communications and synchronizations are minimized.

In contrast, in the shared memory model every processor has direct access to the memory of every other processor in the system. This means a processor can directly load or store any shared address. The programmer can also declare certain pieces of memory to be private to the processors. Parallelism in a shared memory model exists in a form of multiple threads of execution. These threads typically fork from a master thread, and throughout the course of execution may join and refork a number of times. The fork/join execution model makes it easy to get loop level parallelism out of a sequential program. Unlike the message passing model, where the program must be completely decomposed for parallel execution, in a shared memory model it is possible to parallelize just at the loop level without explicitly decomposing the data structures.

Unfortunately loop level parallelism is rarely scalable because it typically leaves some constant fraction of sequential work in the program that, by Amdahl's law, can quickly decrease the gains from parallel execution. Loop level parallelism can easily lead to race conditions and frequent synchronization. False sharing when two or more processors want to write to the same cache line can be another negative factor. All these deficiencies created a "myth" about poor scalability of the shared memory programming model.

It is important, however, to distinguish between the style of parallelism (e.g. loop level vs. coarse grained) and the programming model. Programming in the distributed memory model can be done only with coarse grain parallelism. It is often much more difficult than programming with loop level parallelism in the shared memory model, but it provides a much higher level of scalability. The programmer must be aware of the location of the data in the local memories and has to move or distribute these data explicitly when needed. The partitioning of the data and all necessary communication has to be included explicitly in the program. The sequential program often needs significant changes in order to parallelize it.

But there is nothing in the shared memory parallel model that prevents a programmer from achieving coarse grained parallelism. In the remainder of this paper we will show that the level of parallelism exposed in a program is dependent on the algorithm and data structures employed and not on the programming model.
Therefore, given a parallel algorithm and SMP architecture, a shared memory implementation should scale as well as a message passing implementation.

3 CFD Solver Algorithm

The most widely accepted methods for studying the transient behavior of the compressible Euler and Navier-Stokes equations are implicit. Unfortunately when solving realistic multi-dimensional problems, the system of equations that need to be solved are often too large to apply direct methods to. Most codes resort to solving a modified system of equations and employ techniques such as the implicit approximate factorization [1,2] and the diagonal approximation [3,4], which reduce a block 3D system to a serial sequence of one-dimensional blocked or scalar systems. These techniques work adequately on structured orthogonal grids, but also have uncontrolled factorization errors. Besides, these techniques do not apply for problems defined on unstructured meshes.

Here we employ the Bi-CGSTAB [5] algorithm and use it to solve a linear system of original discretized equations without resorting to approximate factorization techniques.

3.1 Notation

Consider the dimensionless Navier – Stokes equations in the strong conservative form:

$$U_{,t} + F_{i,i} = \mathrm{Re}^{-1} F^{v}_{i,i} \tag{1}$$

where:

$$\mathrm{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho e \end{bmatrix} \tag{2}$$

The Euler flux F_i is given by

$$F_i = u_i U + p \begin{bmatrix} 0 \\ \delta_{i1} \\ \delta_{i2} \\ \delta_{i3} \\ u_i \end{bmatrix} \tag{3}$$

and the viscous flux F_i^v is given by

$$F_i^v = \begin{bmatrix} 0 \\ \tau_{i1} \\ \tau_{i2} \\ \tau_{i3} \\ \tau_{ij} u_i - \dfrac{\mu}{P_r(\gamma-1)}(a^2)_{,i} \end{bmatrix} \tag{4}$$

where the viscous tensor τ_{ij} is given by

$$\tau_{ij} = \lambda\, u_{k,k}\delta_{ij} + \mu\,(u_{i,j} + u_{j,i}) \tag{5}$$

where λ and μ are the viscosity coefficients.

3.2 Discretization

To define the discrete form of the full Navier-Stokes equations we use a general three-point implicit time-stepping method [3]

$$\Delta U^n = \frac{\theta\Delta t}{1+\phi}\frac{\partial}{\partial t}\Delta U^n + \frac{\Delta t}{1+\phi}\frac{\partial}{\partial t}U^n + \frac{\phi}{1+\phi}\Delta U^{n-1} + O[(\theta - \frac{1}{2} - \phi)\Delta t^2 + \Delta t^3] \tag{6}$$

where $\Delta U^n = U^{n+1} - U^n$ and $U^n = U\ (n\Delta t)$. The parameters θ and ϕ can be chosen to produce different schemes of either first or second order accuracy in time. We restrict ourselves to the first order in time scheme with $\theta = 1$ and $\phi = 0$ (although what follows can easily be extended to second order accuracy). Applying eq.(6) to eq.(1) with time step h = Δt results in:

$$U^{n+1} - U^n + h\,F_{i,i}^{n+1} = h\,R_e^{-1}(F_{i,i}^v)^{n+1}, \tag{7}$$

In order to evaluate (7) we must know the advective and diffusive flux vectors F_i and F_i^v at time t = (n+1) Δt. We see that Eq.(7) is nonlinear in U^{n+}. The nonlinear terms are linearized in Taylor series expansion about $\Delta U^n = 0$,

$$F_i^{n+1} = F_i^n + A_i^n \Delta U^n + O(h^2), \tag{8}$$

and

$$F_i^{v\,n+1} = F_i^{v\,n} + M_i^n \Delta U^n + O(h^2), \tag{9}$$

where the Euler Jacobian is $A_i = \partial F_i / \partial U$ and the diffusive Jacobian is $M_i = \partial F_i^v / \partial U$. The analytical form of A_i and M_i can be found in [2].
Using (8) and (9) we rewrite (7) as:

$$[I + h A_{i,i}^n - h M_{i,i}^n] \Delta U^n = h R_e^{-1} F_{i,i}^{v\,n} - h F_{i,i}^n, \tag{10}$$

For the space discretization of flux and Jacobian derivatives we use central differences. Finally, to handle strong shock waves we added a second and fourth-order explicit artificial dissipation terms to the right hand side of Eq.(10) and a second-order implicit diffusive term to the left hand side.

Such discretization leads to a large, sparse nonsymmetric system of linear equations that must be solved at every time step. Thus the unfactored set of linear equations that we solve using a parallel version of Bi-CGSTAB algorithm. With $A = [I + hA_{i,i}^n - hM_{i,i}^n]$ and $x = \Delta U^n$, and $b = hR_e^{-1} F_{i,i}^{v\,n} - F_{i,i}^n$ we rewrite (9) in the familiar linear equation form $Ax = b$.

3.3 Iterative Algorithm

Several iterative algorithms [6] have been proposed for the solution of nonsymmetric systems of linear equations. Here we employ an algorithm due to Van der Vorst [5] which is fast and smoothly converging. The Bi-CGSTAB algorithm is described below.

Bi-CGSTAB:

x_0 is an initial guess; $r_0 = b - Ax_0$;
y_0 is an arbitrary vector, such that
$(y_0, r_0) \neq 0$, e.g. $y_0 = r_0$;
$\rho_0 = \alpha = \omega_0 = 1$;

$v_0 = \rho_0 = 0$;
for i = 1,2,3…
$\rho_i = (y_0, r_{i-1})$; $\beta = (\rho_i / \rho_{i-1})(\alpha / \omega_{i-1})$;
$p_i = r_{i-1} + \beta (p_{i-1} - \omega_{i-1} v_{i-1})$;
$v_i = A p_i$;
$\alpha = \rho_i / (y_0, v_i)$;
$s = r_{i-1} - \alpha v_i$;
$t = A s$;
$\omega_i = (t, s) / (t, t)$;
$x_i = x_{i-1} + \alpha p_i + \omega_i s$;

if x_i is accurate enough then quit;
$r_i = s - \omega_i t$;
end

We note that the iteration loop of Bi-CGSTAB involves two matrix vector multiplications (Ap_i and As). There are also four inner products as well as vector updates which lead to an additional 12 N flops.

4 Implementation

We have implemented the algorithm described in the previous section entirely in Fortran90. Using orthogonal coordinates in three spatial dimensions, the structure of the matrix A is block-sparse. The global spatial structure of A is similar to any second order operator (e.g., Laplacian operator) but additionally A has a local dense 5 by 5 block structure. After computing the Jacobians A_i and M_i which involve mostly local computation with a bit of differencing, the heart of the algorithm involves fetching 5 element vectors from neighboring grid points followed by a local 5 by 5 matrix multiplying a 5 element vector at every grid point.

As is the case in most iterative algorithms, the computational time is dominated by the matrix vector multiplication. Our linear solver has no knowledge of the structure of the matrix and one need only modify the matrix_times_vector( ) subroutine to support other spatial structures (e.g. high order and upwind schemes).

4.1 Parallel Methods

The algorithm described in the previous sections has been parallelized as follows:

Fine-Grain Method. This mode implements loop level parallelism within a shared memory concept. Domain decomposition was applied through multithreading the

outer loops over the last spatial dimension using OpenMP parallel directives. All data structures are shared. Such an approach can be described as one-dimensional domain decomposition with edge dissection (non-overlapping domains, Fig.1a)

Coarse-Grain Method. This mode implements a shared memory paradigm with the use of OpenMP directives. In contrast to the Fine-Grain Mode, this approach is based on the Schwarz additive domain decomposition approach with overlapping local regions (Fig.1b). Major data structures are declared "private" which means that they are not explicitly shared between computational threads. There are only two parallel directives in this mode: the first opens the parallel region in the very beginning of the code and the second one closes the parallel region at the end of this code. In order to use a shared memory paradigm in combination with local data structures, we created a set of shared pointers to those local structures that allows any parallel thread to address private data from other domains. This unusual approach allowed us to implement the coarsest possible level of decomposition within a shared memory paradigm. As a result of this implementation we were able to avoid typical hazards of fine-grain (loop-level) parallelism such as frequent synchronization, false sharing of cache lines and almost completely eliminate a sequential component of parallel code.

Distributed Mode. This mode is also based on Schwarz additive domain decomposition but implements it within a distributed memory paradigm with the use of the MPI (Message Passing Interface) protocol. Algorithmically, Coarse-Grain and Distributed Methods are identical, but the implementation is very different.

The code was compiled with the Intel Linux compiler for the IPF/Linux platform. The OpenMP library was supplied by Intel and the MPI library is the SGI implementation of the MPI-1 protocol.

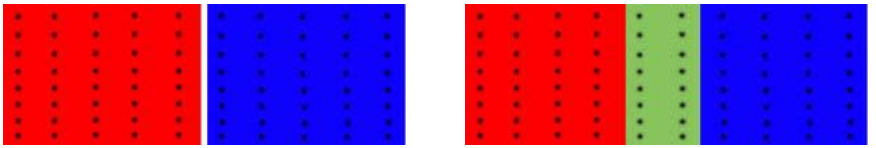


Fig. 1 (a). Partition of a domain in two subdomains

Edge dissection – non-overlapping regions
Vortex dissection – overlapping regions with one "ghost" cells layer

4.2 Flow Problem

We computed the three-dimensional transonic transient laminar flow, which includes such interesting gas dynamic phenomena as shock and rare fraction waves. We believe that such complex problems can be considered a reasonable test.

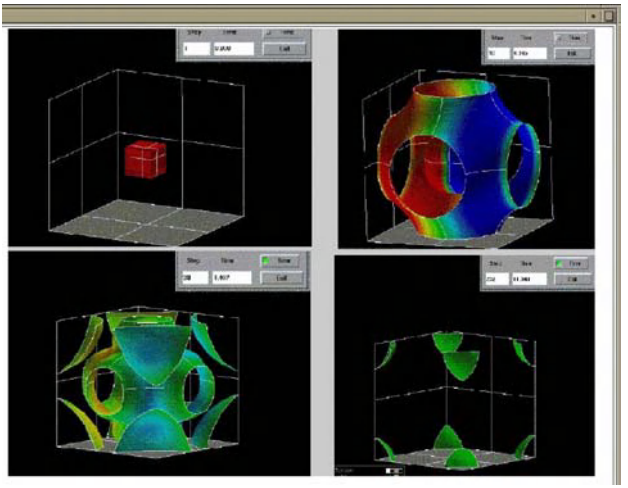


Fig. 2. Blast Waves transient flow (pressure iso - surface colored by velocity field)

The initial configuration consists of a high pressure and density region at the center of the cubic cell with the low pressure gas at rest. Periodic boundary conditions are applied to the array of cubic cells forming an infinite network. At the initial moment the high pressure volume begins to expand in the radial direction as the shock waves flows. At the same time the rare waves move to fill the void at the center of the cubic cell. When the expanding flow reaches the boundaries it collides with its periodic images creating a complicated structure of interfering shock waves (Fig.2). In the case of viscous gas, these processes create a nonlinear damped periodic system with energy being dissipated in time. Finally, the system will come to an equilibrated steady state.

5 Performance Results

The layout of our benchmark suite looked like this:

- We tested 4 sets of input data with variable size in order to evaluate the parallel scaling properties as a function of problem size. The first two dimensions of the computational grid were kept constant but the last dimension along which the parallel decomposition was performed was increased in powers of 2. All physical and computational parameters other than grid size, were fixed for all test cases.
- At first we tested our models on a well known multiprocessor ccNUMA system, the SGI Origin3000 with 128 MIPS R14000 processors running at 600 Mhz clock rate
- The next set of tests was performed on a new SGI system – Altix3300 with 64 Itanium2 processors running at 900Mhz clock rate. This system belongs to the same class of ccNUMA computers as the Origin3000, allowing global addressing from every CPU. The internal network of this system is based on NumaFlex link fabrics from SGI and provides extremely low latency and high bandwidth

communication layer that allows parallel applications to scale to the high count of processors.

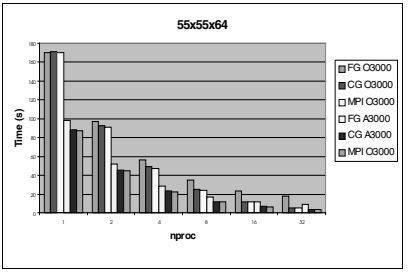


Fig. 3.

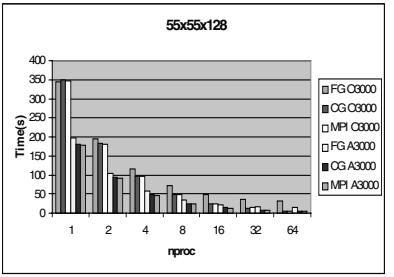


Fig. 4.

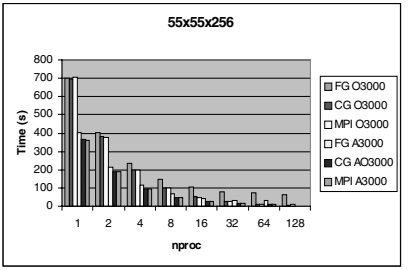


Fig. 5.

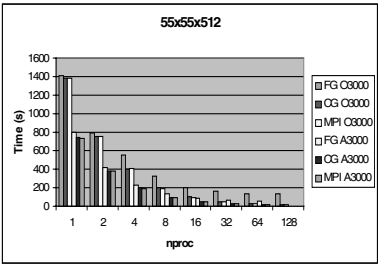


Fig. 6.

We observe the following about this algorithm:

- The easiest way to parallelize any technical application is through loop-level (fine-grain) parallelism within the shared memory paradigm. OpenMP directives provide all the necessary tools for such an approach. Unfortunately the parallel performance of such codes will always be limited by Amdahl's Law constraints.
- The coarse-grain parallel approach eliminates, or significantly reduces, the sequential part of a code, which in turn provides much better parallel scaling. Coarse-grain parallelism can be realized within either distributed or shared memory paradigms. Either memory model requires a complete redesign of the sequential code to expose the parallelism properly. This contrasts with the incremental approach possible with loop-level parallelism. Domain decomposition is one technique for exposing the parallelism.
- The coarse-grain shared memory and distributed memory code show approximately the same level of parallel scaling. Both codes have essentially identical algorithms, which proves that parallel performance of computer codes is not defined by the nature of parallel paradigm in use but rather is a function of a specific implementation algorithm.
- Large shared memory systems present a very capable platform for implementation and production use of both shared memory and distributed memory parallel programs.

6 Conclusions

We demonstrated that scalable programs can be easily written in a shared memory model under constraints of adequate parallel algorithm and data distribution. In order to prove this statement we implemented and tested various parallel modes of a complex CFD code and demonstrated good parallel scaling for a coarse grain parallel implementation in both shared and distributed memory paradigms.

References

1. Beam, R. and Warming, R.F., An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservative Law Form, J. Comp. Phys. 22(1976), 87–110
2. Pulliam, T.H., Efficient Solution Methods for the Navier-Stokes Equations. Lecture Notes for the Von Karman Institute for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Blades. Jan 20–24, 1986, Brussels, Belgium, pp.1–102
3. Warming, R.F. and Beam, R.M., On the Construction and Application of Implicit factored Schemes for Conservation Laws in Computational Fluid Dynamics, SIAM-AMS Proceedings Vol.11 ed. Herbert Keller, American Mathematical Society, Providence, Rhode island, 1978
4. Jespersen, D. and Levitt, C., Numerical Simulation of Flow Past a Tapered Cylinder. NASA Ames Technical Report RNR 90-021, October 1990,1–10
5. Van der Vorst, H.A., Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. SIAM J., Sci. Stat. Comput., Vol.13(2), PP.631–644, 1992

Asynchronous Execution of OpenMP Code

Tien-hsiung Weng and Barbara Chapman

Computer Science Department
University of Houston
Houston, Texas 77024-3010
{thweng,chapman}@cs.uh.edu

Abstract. This paper presents the transformation of OpenMP source code to a Macro-Task Graph, an internal representation of the parallel program as a collection of tasks, which later can be asynchronously scheduled for out-of-order execution and optimized for locality reuse. The transformation is based on array region analysis. We also show the potential benefits of targeting OpenMP code to a macro-task graph, instead of directly generating a multi-threaded program. We show experimental results for a Jacobi kernel and part of the POP code in OpenMP and compiled traditionally versus macro-dataflow execution model using the SMARTS runtime system on SGI Origin 2000.

1 Introduction

OpenMP [1] is a popular parallel programming interface for medium scale high performance applications on shared memory platforms. Strong points are its ability to support incremental parallelization, portability, and ease of use. However, there are several obstacles to scaling an OpenMP code to hundreds or thousands of processors, in particular latency of remote memory access, poor cache memory reuse, large numbers of barriers and false sharing of data in cache. However, it is up to the user to ensure that performance does not suffer as a result. Good data locality is generally needed to overcome these performance problems. An additional potential cause of performance degradation on ccNUMA (cache coherent Non-Uniform Memory Access) systems is that a thread may need to access a remote memory location; despite substantial progress in interconnection systems, this still incurs significantly higher latency than accesses to local memory, and the cost may be exacerbated by network contention. If data and threads are unfavorably mapped, cache lines may ping-pong between locations. However, OpenMP provides few features for managing data locality. In particular, it does not enable the explicit binding of parallel loop iterations to a processor executing a given thread; such a binding might allow the distribution of work such that threads can reuse data.

The method provided for enforcing locality is to use private variables. However, systematically applied privatization may require a good deal of programming effort, akin to writing an MPI program. Several approaches have been proposed to provide

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 667–676, 2003.
© Springer-Verlag Berlin Heidelberg 2003

performance with modest programming effort, including extending OpenMP by data distribution directives and directives to link parallel loop iterations with the node on which specified data is stored [2,3], first touch memory allocation, dynamic page migration by the operating system [3], user-level page migration [4], a combination of OpenMP and MPI [5], and compiler optimization [6]. Programming using data distribution directives has the potential to help the compiler ensure data locality; however, it means that programmers should be aware of the pattern in which threads will access data across large program regions. Hence the programming task remains more complex than with plain OpenMP. Moreover, there is no agreed standard for specifying data distributions in OpenMP; existing ccNUMA compilers have implemented their own data distribution directives. Thus, when they are used, performance may well be improved, but portability will be sacrificed. So OpenMP users may be forced to choose between simplicity and performance.

Our approach is to put work into compiler optimization and to use a runtime system that can better exploit data locality, provide higher degrees of parallelism, and reduce synchronization overheads. We target OpenMP to a data flow execution model realized using the SMARTS runtime system developed at Los Alamos National Laboratory.For this effort, we first evaluated SMARTS as a runtime system for OpenMP on SMPs and ccNUMA systems and then developed a Fortran API for SMARTS, which serves to pass information from compiler to runtime system. In the compiler analysis, we transform OpenMP to a macro-task graph. A macro-task graph is an intermediate representation (IR) of the OpenMP program where each node represents either a task or sub-iteration of a loop, called an *Iterate*, that may be assigned to a processor to be executed according to the precedence constraints represented by arcs of the graph. The assignment of a node to a processor can be permanent or hint. An arc represents a data dependence relation between a pair of nodes. The construction of this IR is based heavily on array region analysis. It describes or summarizes sub-regions of array that are accessed by a section of code and is commonly used for interprocedural side effect analysis, data dependence testing, array privatization, locality improvement, as well as communication optimization. It is applied to analyze the access patterns of the parallel program in order to make decisions on scheduling of the computation so that each processor may reuse data. However, it can be expensive to apply this testing between two call statements or between two different larger regions of code.

2 The SMARTS Runtime System and Our API

SMARTS [7](Shared-Memory Asynchronous Runtime System) was developed at Los Alamos National Laboratory to support both task and data parallelism. Written in Object-Oriented C++, it is originally designed to be the runtime system for POOMA (Parallel Object-Oriented Method and Application). The novel aspect is its macro-dataflow approach [8]. SMARTS supports the efficient execution of code, for which both the data and the work has been decomposed, on shared memory multiprocessors and ccNUMA systems by applying the data flow concept of exploiting temporal locality. It relies upon a prior partition of the original loop iteration space into so-called

iterates, and a partition of the arrays in the program into data-objects. The array partitions defined and used in each iterate must be determined by the compiler. Based upon the data accesses, a task graph is constructed that captures the dependences between iterates. When an iterate requires data that is (partially) created by another iterate, the original execution order must be preserved; similarly, anti- and output dependences between iterates impose an ordering on their execution. So long as these dependences are respected, iterates that are not involved in such relationships can be scheduled concurrently for out-of-order execution. Note that the execution behavior depends on both the data and the work decomposition. The decomposition of an entire loop into many sets of iterations may result in having many independent iterates available for execution at any given time. To reduce the overheads associated with thread creation, SMARTS creates threads at the beginning of the program and carries out a join at the end of the program.

A pool of parallel tasks is implemented by multiple work queues to avoid contention on a shared queue and to promote cache locality of the task queues. When the workload is well distributed across work queues, contention is minimized. When a slave thread is idle, it first grabs work from its own work queue for execution; but if its queue is empty, it may steal work from another work queue. SMARTS uses the first touch policy by default. Under this policy, the process that first writes or reads a page of memory causes that page to be allocated in the node on which the process is running. Two variants of affinity scheduling have been employed: soft and hard affinity scheduling. With soft affinity or hint affinity scheduling, the user or compiler can specify where an iterate is to be bound, but the scheduling is up to SMARTS with work stealing when there is an imbalanced work load. Hard affinity scheduling means that an iterate is statically bound to a thread by the compiler without work stealing.

```
call OMP_SET_NUM_THREADS(4)
!$OMP PARALLEL
  do k=0, ITER
!$OMP DO
    do j = 1, size-2
       do i = 1, size-2
          A(i,j)=(B(i,j-1)+ B(i,j+1)+
                  B(i-1,j)+ B(i+1,j)) /4.0
       enddo
    enddo
!$OMP ENDDO
!$OMP DO
     do n = 0, size-1
        do m = 0, size-1
           B(m,n) = A(m,n)
        enddo
     enddo
!$OMP ENDDO
  enddo
!$OMP END PARALLEL
```

Fig. 1. Standard OpenMP version of Jacobi kernel

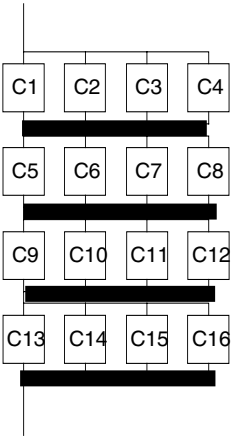


Fig. 2. Runtime execution model for Figure 1 (C stand for chunk or iterate and horizontal bold lines represent barrier)

Our strategy is to gather as much information as possible from the compiler to reduce runtime overhead. The information required includes iterate identification, read/write requests to data objects, mapping of specific iterates to thread and the type of mapping. The mapping decision is determined by generating the task graph (iterate dependence graph) and applying a heuristic scheduling algorithm at compile time.For hard affinity scheduling, each iterate is assigned a thread id and will be executed by that thread when its data becomes available.

To illustrate this, we show a simple OpenMP Jacobi kernel in Figure 1. Figures 3 and 4 show the corresponding code parallelized using calls to SMARTS via the API we have created. Figure 5 illustrates the iterate dependence graph constructed by the compiler (it may be compared with the execution model associated with the original OpenMP code in Figure 2). Based upon this graph, the compiler might assign the iterates to threads as follows: C1 to the master thread, C2 to thread 1, C3 to thread 2, and C4 to thread 3, then C5 to the master thread, C6 to thread 1, C7 to 2, C8 to 3 in order to reuse cache.

Figure 3 also illustrates the structure of the generated SMARTS code and our API. The main program is first executed by the master thread which creates additional threads, performs initializations, and passes read/write access information for iterates to the runtime system, along with iterate affinity, and iterate identification, etc. The call to SM_concurrency(n) creates an additional n-1 slave threads in which all iterates will be run. Information such as array partition information is passed via the API function SM_define_array; SM_startgen tells the scheduler that the master thread is starting a new data parallel statement. Information on read/write access of iterates to data objects is then passed to the runtime system via SM_def_readwrite.

```
call SM_concurrency(n)
call SM_define_array(info)
schedtype = hintaffinity
call SM_startgen()
do k = 1, iter
   do i1 = 1, sqit
      do j1 = 1, sqit
         call SM_def_readwrite(loop0)
         call SM_getinfo(loop0, affin, datinfo)
         call SM_handoff(0, schedtype, affin, datinfo)
      end do
   end do
   do i1 = 1, sqit
      do j1 = 1, sqit
         call SM_def_readwrite(loop1)
         ! handOff for loop 1
         call SM_handoff(1, schedtype, affin, datinfo)
      end do
   end do
end do
call SM_run()
call SM_end()
```

Fig. 3. Main program using SMARTS

Next, the API routine SM_handoff takes arguments such as iterate, type of affinity scheduling, and a thread ID that specifies the iterate mapping to slave threads and passes it to the runtime scheduler. SM_run is called to tell the scheduler that no more

iterates will be handed off and the scheduler starts to work. SM_getinfo obtains information such as iterate affinity and the bounds of each iterate from the compiler.

```
subroutine fortran_loop0(datinfo)
  use shared_DATA
  call SM_compute_bound(datinfo,lb1,ub1,lb2,ub2)
  do j = lb1, ub1
     do i = lb2, ub2
        A(i,j) = (B(i,j-1)+ B(i,j+1)+
                  B(i-1,j)+ B(i+1,j)) / 4.0
     enddo
  enddo
end subroutine fortran_loop0

subroutine fortran_loop1(datinfo)
  use DATA_jacobi
  call SM_compute_bound(datinfo,lb1,ub1,lb2,ub2)
  do n = lb1,ub1
     do m = lb2, ub2
        B(m,n) = A(m,n)
     enddo
  enddo
end subroutine fortran_loop1
```

Fig. 4. Parallel loops using SMARTS

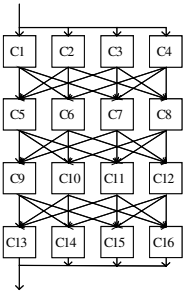


Fig. 5. Dataflow execution model associated with Figure 1

As soon as iterates have been handed off, the SMARTS runtime scheduler checks if their data are available. If so, the scheduler immediately schedules the iterate to the specified slave queue for execution. When the slave thread is idle, it grabs the iterate and the parallel loops routine associate with the iterate as Figure 4 will be invoked. In Figure 5, C1, C2, C3, and C4 correspond to subroutine fortran_loop0, and C5, C6, C7, and C8 correspond to subroutine fortran_loop1. At the end of the parallel region, SM_end is invoked to perform a join, which destroys the slave threads.

3 Transformation of OpenMP to Macro-Task Graph

Given an OpenMP source code, our aim is the compiler generation of semantically equivalent parallel code that is represented by a task graph for Macro-Dataflow computation and scheduling of the graph nodes for cc-NUMA systems. First, tasks are generated by following the semantics of OpenMP worksharing constructs. The grain size of an iterate is important. When it is too coarse, parallelism may be limited and load imbalance may be a problem; while too fine granularity will cause excessive overheads. After a suitable grain size is obtained, the compiler must determine any data dependences between pairs of nodes by array section analysis to generate the macro task graph. Where dependences exist, the compiler must further determine the reuse between the two nodes by applying array region analysis [9,10,11]. Thus a macro task graph ***G(N,E)*** consists of a set of nodes $\boldsymbol{N} = \{n_1, n_2, .., n_m\}$ connected by a set of edges ***E***, each of which is denoted by e_{ij}. Each node represents a task. A task can be a region of code such as basic block, sub-iteration of loop (parallel loop), proce-

dure, etc. The weight of node n_i, $w(n_i)$ is its execution time. Each edge e_{ij} represents a precedence constraint that must hold between node i and node j. The weight $w(e_{ij})$ is the level of reuse of data between the two end nodes in a percentage.

The task graph is generated by data dependence testing using regular section analysis. We use the example in Fig. 7 (a): A1, A2, A3, and A4 to illustrate our approach. Suppose that task 1 accesses areas A1 and A2; if we apply an immediate union, we get A'=A1∪A2, that is, a summary which also covers un-accessed regions shown as dark areas in (b). Now suppose task 2 accesses A3. The dependence relation between task 1 and task 2 is obtained by intersecting A3 and A', A''=A3∩(A1∪A2) as in (c); A'' is non-empty which causes a *false dependence*. But it is also equal to A''= (A3∪A1)∩(A3∪A2); this requires three operations, but the solution is exact as in (d). Therefore, we test whether the union is exact; if it is, we perform it immediately; otherwise, we delay the union and the two sub-regions are kept. But, we must make sure that the number of elements in the list does not exceed a threshold value.

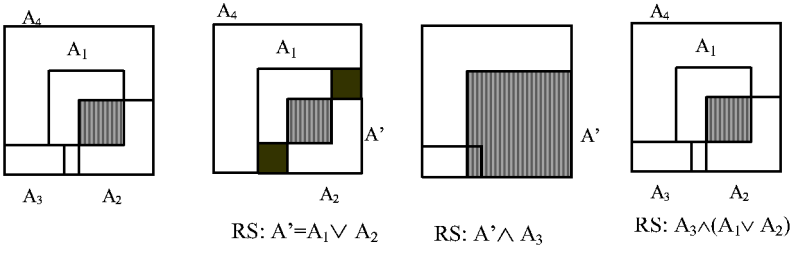


Fig. 6. Union and intersection operations

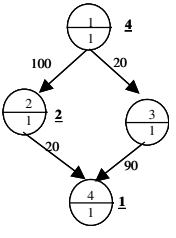


Fig. 7. Task graph

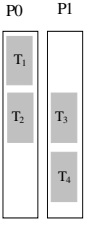


Fig. 8. Static scheduling, two processors

Fig. 8 shows the schedule reuse for the task graph in Fig. 7. We assume that first touch policy is applied. Task 2 is scheduled to p0 for data locality since its level of reuse is the highest of task 1's immediate successors. Task 3 can be scheduled on p1. Finally, task 4 is scheduled to p1 since the incoming edge from its parents task 3 indicates maximum reuse. The first touch policy can be used to obtain good memory access locality [14] even with a program that has irregular memory access patterns. By analyzing the memory access patterns in the program during graph generation, we can schedule tasks for reuse.

4 The Benefits of a Dataflow Execution Model for OpenMP

Although there is little potential for speedup with our Jacobi code, the above example already hints at the performance improvement possible if SMARTS is used as a runtime system for OpenMP. Specific benefits of targeting OpenMP to the dataflow execution model include cross-loop out-of-order execution, temporal data locality, and reduction of synchronization overheads.

4.1 Reduction of Synchronization Overheads

OpenMP employs a fork-join programming model, in which a master thread executes the sequential regions of the program. When a parallel region is encountered, it forks additional worker threads to share the work. The model is both flexible and simple, but it may incur significant synchronization overheads. In Figure 1, the first parallel DO construct requires a barrier to maintain program correctness since there is a true dependence between the first loop and the second loop. The corresponding runtime execution schema is shown in Figure 2, where the iterations performed by each thread are assumed to be a chunk. In this case, each chunk of the second parallel DO construct (C5, C6, C7 and C8) must wait until C1, C2, C3, and C4 have completed. On the other hand, in the dataflow execution model as shown in Fig. 5, each of the parallel loops in the second loop nest only wait for three iterates of the first parallel DO construct to finish. Thus this approach can reduce the synchronization overhead.

4.2 Data Locality

It is up to the programmer to write OpenMP code that makes good use of cache and avoids false sharing of cache lines by appropriately privatizing data, possibly padding shared data and blocking loops. Unfortunately, this requires a nontrivial effort. The SMARTS runtime system uses the first touch policy and affinity scheduling with or without work stealing to support locality. Work stealing is clearly useful for load balancing. But if it is too eagerly applied, it may lead to performance degradation, due to the potential conflict with cache reuse. So we assign cache reuse a higher priority than load balancing. We schedule iterates that use the same data to the same processors.

4.3 Out-of-Order Execution

Traditional OpenMP compilation will lead to execution of one parallel construct (DO or SECTIONS) in parallel. SMARTS allows an out-of-order execution within and between parallel constructs if and only if there are no data dependencies between the parallel constructs. Therefore it is necessary to compute the data dependencies between parallel constructs by applying the appropriate array section analysis and array data flow analysis. With the standard OpenMP execution model in Fig. 8, C17 can

only be executed after C13 finishes and C21 can only be executed when C17, C18, C19, and C20 have completed. A translation from OpenMP to SMARTS provides an opportunity for out-of-order execution. As illustrated in Fig. 9, under this approach C17 no longer needs to wait for C13. Further, the chunks can be reordered to permit cache reuse.

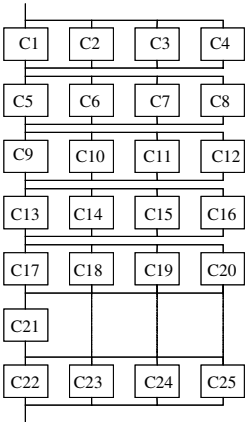


Fig. 9. Runtime execution model for OpenMP

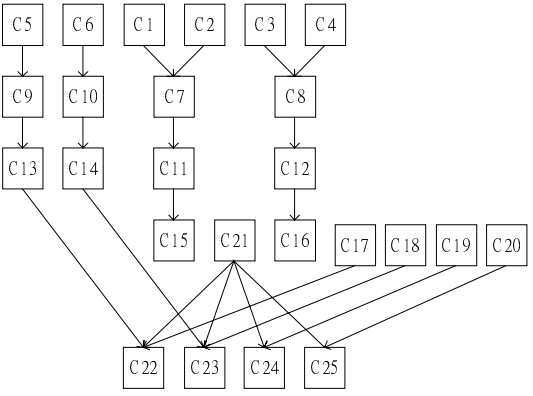


Fig. 10. Dataflow execution model associated with translated code of Figure 9

5 Experimental Results

We show results of comparisons between OpenMP and SMARTS versions of a simple application based upon the Jacobi kernel as well as one in which we translated part of a Fortran90 OpenMP version of the POP code to a corresponding parallel code based on SMARTS. POP (the Parallel Ocean Program) [13] is an explicit finite difference model developed by the Los Alamos ocean modeling research group. It makes extensive use of modules, array syntax, and WHERE constructs. Our target platform was an Origin 2000 at the National Center for Supercomputing Applications (NCSA). They were compiled with SGI's MIPSpro Fortran 90 compiler under the option –64 –Ofast –IPA and run on MIPS R10000 processors at 195 MHz with 32 Kbytes of split L1 cache, 4 Mbytes of unified L2 cache per processors and 4 Gbytes of DRAM memory. We made use of the strategies provided by SGI to obtain good performance under OpenMP. First touch allocation, where a datum is stored on the node where it is first accessed, was used in each case. We also set _DSM_MIGRATION (page migration) environment variable to OFF.

Different implementations of OpenMP on the available hardware platforms incur different overheads for OpenMP directives. Synchronization and loop scheduling overheads have been measured by [12]. The PARALLEL DO directive with REDUCTION clause has one of the worst overheads under SGI's MIPSpro f90 compiler on the SGI Origin 2000. Even though the overhead of the barrier directive itself

is minor, the time spent waiting for all threads to finish could be significant when large numbers of processors are involved.

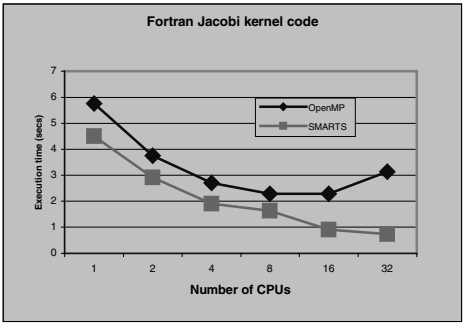


Fig. 11. Execution time for OpenMP Jacobi and translated Jacobi using SMARTS with array size of 2048x2048

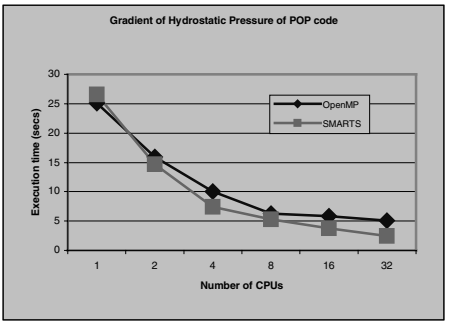


Fig. 12. Execution time for POP routine computing Gradient of Hydrostatic Pressure and translated code using SMARTS with array size of 2048x2048

We show results for a Jacobi computation based upon runs with a 2048 by 2048 double precision matrix in Fig. 10. With more processors, the barrier clearly increasingly affects the performance; from 8 processors on, it does not give good speed up. On the other hand, under data flow execution, the Jacobi program obtains better speedup up to 32 processors. When both programs are compiled with –O2, the difference between the two is much larger.

Fig. 11 illustrates our results on a routine from the POP code that computes the gradient of hydrostatic pressure at level k. For our measurement, we executed the code with an array of size 2048 by 2048. The number of data objects and read/write requests are greater than that of the Jacobi kernel. Hence, the initialization overheads incurred when passing information to the runtime system lead to a longer execution time for the translated code on one processor. This overhead is amortized by the function of the problem size and the number of iterations.

6 Conclusions

In this paper, we show that by using the SMARTS runtime system as an execution target for OpenMP, we can increase program performance by improving data locality, reducing synchronization overhead as well as exploiting out of order execution to maximize parallelism. An API has been designed to interface between Fortran and the SMARTS C++ library. We compare OpenMP with a translated code using SMARTS on two codes, clearly showing that the translated version outperforms the OpenMP code and also scales well. We can do even better by further developing the compiler to improve the analysis required for the translation. Our approach avoids the use of data distribution directives, which add complexity to the programming model and poten-

tially destroy portability. Our experimental results consistently show a benefit from this approach.

References

1. OpenMP Architecture Review Board, Fortran 2.0 and C/C++ 1.0 Specifications. At www.openmp.org.
2. B.Chapman, P.Mehrotra, and H.Zima, Enhancing OpenMP with Features for Locality Control. Proc. ECMWF Workshop Towards Teracomputing {The Use of Parallel Processors in Meteorology". Reading, England, November 1998.
3. J. Bircsak, P. Craig, R. Crowell, J. Harris, C. A. Nelson, C. D. Offner, Extending OpenMP For NUMA Machines: The Language, WOMPAT 2000, Workshop on OpenMP Applications and Tools, San Diago, July 2000.
4. D. S. Nikolopoulos, T. S. Papatheodorou, C. D. Polychronopoulos, J. Labarta and E. Ayguade, Is Data Distribution Necessary in OpenMP ? Proc. of the IEEE/ACM Supercomputing'2000: High Performance Networking and Computing Conference, Dallas, Texas, November 2000.
5. L. A. Smith and P. Kent, Development and Performance of a Mixed OpenMP/MPI Quantum Monte Carlo Code,Proceedings of the First European Workshop on OpenMP, Lund, Sweden, Sept. 1999, pp. 6–9.
6. S. Satoh, K. Kusano, and M. Sato, Compiler Optimization Techniques for OpenMP Programs, Second European Workshop on OpenMP (EWOMP 2000), Edinburgh, September 14–15, 2000.
7. S. Vajracharya, S. Karmesin, P. Beckman, J. Crotinger, A. Malony, S. Shende, R. Oldehoeft, and Stephen Smith, Exploting Temporal Locatlity and parallelism through Vertical Execution. ICS '99.
8. S. Vajracharya, P. Beckman, S. Karmesin, K. Keahey, R. Oldehoeft, and C. Rasmussen, Programming Model for Cluster of SMP. PDPTA '99.
9. P. Havlak and K. Kennedy. An implementation of interprocedural bounded regular section analysis. IEEE Transactions on Parallel and Distributed Systems, 2(3):350–360, July 1991.
10. V. Balasundaram, K. Kennedy, A Techniques for Summarizing Data Access and its Use in Parallelism Enhancing Transformations, Proceedings of ACM SIGPLAN'89 Conference on Programming Language Design and Implementation, Jun. 1989.
11. R. Triolet, F. Irigoin, P. Feautrier, Direct Parallelization of CALL Statements, Proceedings of ACM SIGPLAN '86 Symposium on Compiler Construction, Palo Alto, CA, July 1986, pp. 176–185.
12. J.M. Bull, Measuring Synchronisation and Scheduling Overheads in OpenMP, Proceedings of the First European Workshop on OpenMP, Lund, Sweden, Sept. 1999, pp. 199–105.
13. Parallel Ocean Program, http://www.acl.lanl.gov/climate/models/pop
14. D. S. Nikolopoulos and E. Artiaga and E. Ayguadé and J. Labarta. Exploiting memory affinity in OpenMP through schedule reuse. ACM SIGARCH Computer Architecture News, 29(5):49–55, 2001.

Virtual Reef: A Visualisation Framework for Marine Simulation Models

Stephen Jeffrey

Advanced Computational Modelling Centre,
University of Queensland, St. Lucia, 4072, Australia
sjj@maths.uq.edu.au
http://www.acmc.uq.edu.au

Abstract. A real-time application has been developed to enable immersive stereoscopic visualisation of the results of marine simulation models. Static datasets from external applications or dynamic output from real-time models may be interactively examined in a simulated underwater environment. The current work describes how the visualisation framework has been coupled to a morphogenetic model of coral growth and a biomechanical model of a fish. Coral structures have been modelled using the techniques developed by Kaandorp, and artificial fish have been constructed using methods adapted from the work of Tu.

1 Introduction

Mathematical models of complex systems are rarely complete; researchers are usually engaged in an iterative process of model refinement and output analysis. The technique used for assessing and/or comprehending model outputs usually depends on the nature of the simulation and also the size and complexity of the data. Visual techniques can be particularly useful, especially for three-dimensional datasets, as the human perception system is well adapted to performing graphical tasks such as pattern recognition. Graphical tools are therefore well suited to analysing outputs from environmental models given the spatio-temporal characteristics of many environmental processes.

The purpose of this paper is to describe the construction of a "virtual reef" that has been designed to facilitate the development of marine simulation models by using immersive visualisation. The graphical framework consists of an underwater scene and a range of navigation modes and visual effects. Datasets can be displayed either by first converting the data to a standard graphics format, or alternatively, the user may write their own C/C++ function for loading and rendering the data as desired. The conceptual layout of the framework is shown in Fig. 1.

Preliminary models of coral growth and fish movement have been implemented to demonstrate how model outputs may be examined in the graphical

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 679–687, 2003.
© Springer-Verlag Berlin Heidelberg 2003

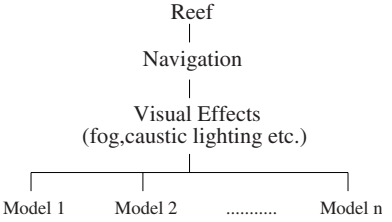


Fig. 1. Structure of the "virtual reef" application

environment. The coral and fish models are described in Sections 2 and 3, respectively, followed by implementation issues in Section 4. Concluding remarks are presented in Section 5.

2 Coral Growth Model

A coral growth model based upon the work of Kaandorp (See [1] and references therein) has been used. The model attempts to simulate the radiate accretive growth process exhibited by many corals, sponges and algae. A longitudinal cross section through a colony of *Montastrea annularis* (Fig. 2) demonstrates

Fig. 2. Longitudinal cross section through a colony of *Montastrea annularis*. Photograph taken from [2]

the process: growth occurs in layers whereby a new layer is added to the existing structure. The growth is radial in nature due to the process's dependence on light and/or nutrients for growth. Growth is a maximum at the centre of the growth axis where the surface area (through which light and/or nutrients are absorbed) is a maximum and progressively decreases towards the side of the colony as the effective absorption area decreases. The two-dimensional growth model is shown in Fig. 3 where the *surface elements* are the tangential edges along the new

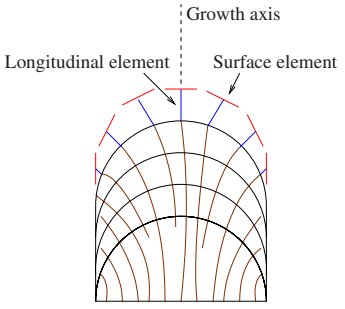


Fig. 3. Diagram shows the radiate accretive growth process. The lowest arc represents the initial state and the higher arcs represent successive growth layers. Diagram reproduced from [2]

outer surface, and the *longitudinal elements* show the growth between surface elements in successive layers. The basic growth model is given by:

$$l = s \cdot \sin(\alpha),\ 0 \leq \alpha \leq \pi \tag{1}$$

where l is the distance grown, s is the maximum growth and α is the angle between the surface tangent (at the point of interest) and the growth axis. Additional rules are progressively added to refine the model and are summarised in Table 1.

A three-dimensional model for radiate accretive growth can be constructed by a relatively straightforward extension of the two-dimensional model. In the 3D case, the 2D *surface elements* are replaced by triangles where each vertex represents the location of an individual polyp. The triangular structure is motivated by the fact that corallites commonly have a pentagonal and hexagonal arrangement which can be conveniently tessellated using triangles. The simulation starts with an *initiator* obtained by constructing a triangulated approximation to a sphere. Each vertex on the surface represents an individual polyp and is grown using a set of rules which are essentially 3D analogs of the 2D rules in Table 1. The early stages of the growth process are illustrated in Fig. 4. The structure on the left is the *initiator* which seeds the simulation. The middle structure is obtained after one growth iteration and it can be seen that one triangle has been subdivided into four smaller triangles. The subdivision is a result of the Insertion rule and it represents the emergence of new polyps. The structure on the right shows the colony branching. The structure has widened due to environmental factors and the effect is implemented by the Widening rule. The colony may widen if, for example, there is a predominant current resulting in the increased transportation of nutrients in a specific direction. Growth (for a heterotrophic species) will accelerate in the direction normal to the current. When the structure has widened sufficiently, growth may cease in the central area and two or more branches may emerge. The branches appear due to increased growth in regions with a relatively high surface area, through which light and/or nutrients are absorbed. This effect is implemented by one or more of the Contact, Light

Table 1. Rules used in the model describing two-dimensional radiate accretive growth

Rule	Description
Growth	Distance grown (Eqn. 1)
Continuity	Adjacent surface elements form a smooth surface (nearby polyps grow at similar rates)
Insertion	Insert a new surface element between adjacent surface elements that are sufficiently far apart (emergence of a new polyp)
Inhibition	Threshold below which growth is truncated (polyp activity ceases)
Widening	Growth is accelerated on the side of the structure. This rule may be used in basic models to approximate the effect of the Light and Concentration rules (all polyps within a region grow at a similar rate)
Association	Associate each surface element with a growth axis. A growth axis is defined for each region of surface elements that are actively growing (emergence of a new branch)
Deletion	Remove a surface element when adjacent surface elements are growing toward each other and are sufficiently close (polyp growth suppressed to avoid collisions)
Deregulation	Random component incorporated into the Growth rule (polyp growth is disturbed by surface irregularities or turbulence in the environment)
Contact	Growth is made a function of the local radius of curvature (polyp growth is influenced by the surface area through which it can absorb light &/or nutrients)
Light	Growth is made a function of the angle between the light source, the point of interest (on the surface) and the surface normal at that point. The light model provides a simple estimate of the available light. A realistic model is not used as it would be too expensive to compute (polyp growth is dependent upon light)
Concentration	Growth is made a function of the local nutrient concentration, computed using the discrete Laplace equation (polyp growth is dependent upon nutrient concentration)

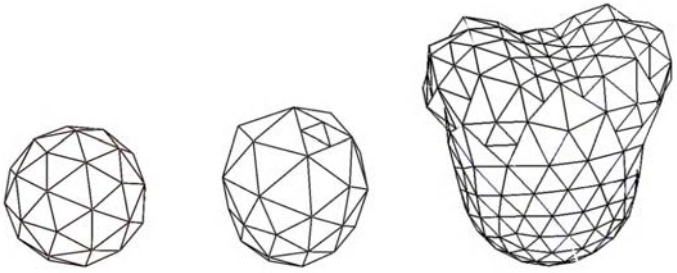


Fig. 4. Simulated radiate accretive growth in 3D. The structure on the left is the *initiator*, the middle structure is the form after one growth iteration and the structure on the right shows the colony branching. For further details, see the text

and Concentration rules. The fertile regions remaining after the branching then continue to grow along separate growth axes. Growth along these axes will be determined by each polyp's relationship with its new axis (as opposed to the axis prior to branching) and other environmental factors. Polyps are assigned to new growth axes by the Association rule.

The three-dimensional growth algorithm is built up from a set of rules similar to those used in the two-dimensional case. Further details on the growth algorithm may be found in [2].

3 Fish Model

Artificial fish have been constructed using a technique adapted from Tu's work [4]. The fish model consists of 3 distinct components: perception, behaviour and locomotion. The perception module is used to determine which objects are in the fish's field of view. An advanced recognition algorithm is not required as a list of all objects in the scene is known and a simple line of sight algorithm is used. The behaviour module takes input from the perception module and determines what action to take. A list of possible actions is constructed based upon the fish's knowledge of the location of other fish and its current desires and habits. Desires include the fish's intention to eat when hungry, flee from predators or school with other fish when appropriate. The candidate actions are ranked in order of priority and are executed by the locomotion module. Actions are prioritised to determine the most appropriate course to take in the current situation. For example, if the perception module has identified a nearby predator and the fish is hungry, the action selection mechanism should instruct the fish to flee. The desire to forage for food should be secondary to the fish's survival and executed after the fish has evaded the predator. The locomotion module consists of a mass-spring system representing a simplified model of the fish's muscle structure, and a set of motor controllers for activating the "muscles". Each motor controller is responsible for controlling the muscle actions required to implement a specific behaviour *e.g.,* swimming, gliding, turning *etc.*

The mass-spring system consists of 91 damped visco-elastic units arranged as in Fig. 5. The fish dynamics are simulated by numerically propagating the equations of motion over time. The governing equations are expressed as a set of coupled non-linear second order differential equations and solved numerically using a 4,5 Runge-Kutta algorithm with an adaptive step size [5]. In each time

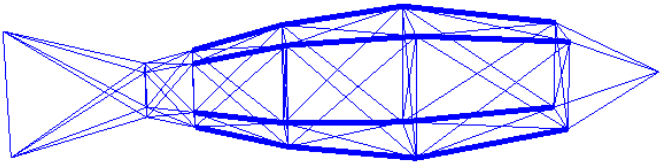


Fig. 5. The fish's internal structure. Each line represents a damped visco-elastic unit; muscles are shown as thick lines

step the motor controllers update the desired muscle lengths which then exert a force on the connecting nodes. The equations of motion are solved and the position of each node is updated.

The fish can be brought into a desired conformation by using an appropriate sequence of muscle actions. For example, a periodic relaxation and contraction of the tail muscles (muscles on either side are out of phase by 180°) make the tail oscillate [6]. The induced hydrodynamic forces propel the fish forward, thus simulating a swimming motion.

The fish body is made by mounting an external frame around the internal mass-spring system (Fig. 6). The offset between vertices in the external frame

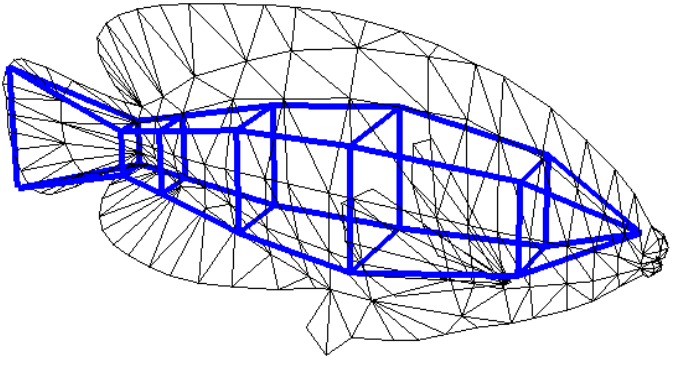


Fig. 6. The fish's body shape is given by mounting an external mesh relative to the internal frame

and internal nodes is maintained as the internal frame moves. The offset ensures the fish's external shape deforms to match the internal structure.

Work on the perception and behaviour modules is not yet complete; the fish are currently controlled using key strokes *i.e.,* the user sends commands directly to the locomotion module.

4 Implementation Details

The virtual reef has been designed for immersive stereoscopic visualisation. Its target machine is an SGI Onyx 3000, projecting three channels onto a curved screen 2.5m high with a 150° field of view. The software has been written in C++ and uses OpenGL Performer [7]. Visual effects such as fog and caustic lighting have been used to enhance the immersive experience. Caustic lighting effects improve the realism of the underwater scene and are applied using texture-based colour modulation [8]. The SGI hardware does not support multi-texturing so a multi-pass rendering algorithm is used to implement the caustic effects. Fog is used to simulate a turbid environment but cannot be applied in the usual manner as it does not work correctly in the multi-pass environment. Fog is applied to each pixel using a one-dimensional texture map, where the texture coordinate is the normalised eye-distance. The eye-distance to each pixel is computed by reading the depth buffer and reversing the perspective scaling.

The application has been demonstrated by using it to display colonies of coral generated by a morphogenetic growth model. Texture mapping has been used to make the coral visually appealing but its application presents some interesting challenges. The growth algorithm was described in Section 2 where it was noted that vertices in the triangulated surface represent the location of an individual polyp. To reflect this one-to-one correspondence between vertices and polyps, a texture map of a single corallite can be centred on each vertex comprising the surface. The method can be used on structures generated by the coral growth algorithm, but is unsuitable in practice as the original structures undergo further processing. To reduce the graphics load, polygon reduction and level-of-detail techniques are used, thereby removing the correspondence between vertices and polyps. Since vertices in the processed structures no longer have any biological significance, there is no need to incur the graphics overhead associated with centering a texture map on each individual vertex. In practice textures are applied using automatic texture coordinate generation based upon distances from multiple planes.

The application can simultaneously display model outputs from a number of diverse models as the graphics are independent of the underlying simulation algorithms. The examples presented in Sections 2 and 3 demonstrate the separation between models and graphics. The coral simulation is an external application that generates coral structures which are then added to the underwater scene at run time. The fish simulation is compiled into a dynamic library and linked at run time. Since both algorithms are independent of the visualisation application, alternate simulation models can be easily substituted.

5 Conclusion

A visualisation framework has been described which may be used to analyse the outputs of marine simulation models. Example applications have been developed and demonstrate how the graphical environment can assist the development of models by enabling researchers to interactively examine their results. In addition to its usefulness as a research tool, the virtual reef can be used for educational purposes or simply for entertainment.

A scene from the virtual reef featuring both the fish and coral models is shown in Fig. 7.

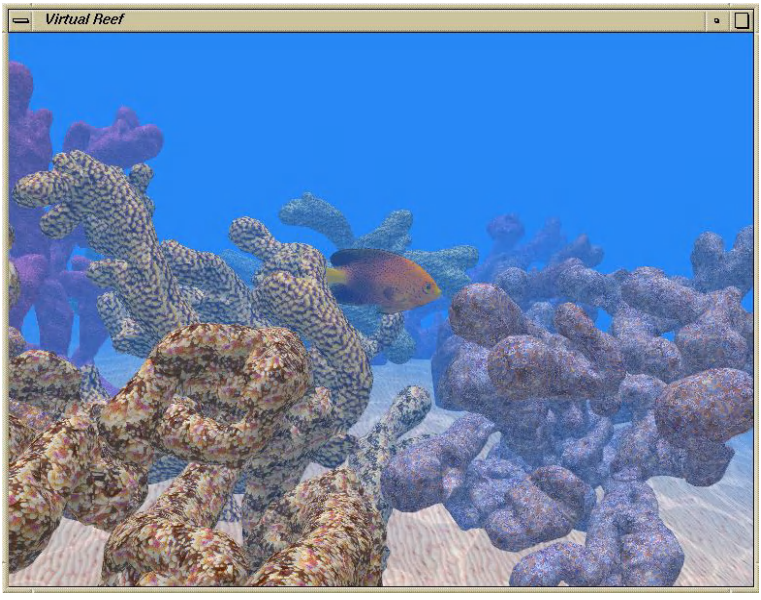


Fig. 7. Underwater scene taken from the Virtual Reef

The virtual reef is currently used for displaying dynamic datasets from both real-time and non-real-time simulation. Further development will focus on the display of observational datasets and the construction of a hydrodynamic model to simulate the flow of nutrients and its effect on fish movement.

Acknowledgements.

This work was carried out in the Visualisation and Advanced Computation Laboratory at the University of Queensland, using machines funded by the Queensland Parallel Supercomputing Foundation and the Australian Partnership for Advanced Computing. I would like to thank Dr Jaap Kaandorp for many useful discussions regarding the coral growth algorithm.

References

1. Kaandorp, J.A., Kübler, J.E.: The algorithmic beauty of seaweeds, sponges, and corals. Springer-Verlag, Berlin Heidelberg New York (2001)
2. Kaandorp, J.A.: Fractal Modelling: Growth and Form in Biology. Springer-Verlag, Berlin Heidelberg New York (1994)
3. Kaandorp, J.A., Sloot, P.M.A.: Parallel simulation of accretive growth and form in three dimensions. BioSystems. **44** (1997) 181–192
4. Tu, X.: Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception and Behaviour. Lecture Notes in Computer Science, Vol. 1635. Springer-Verlag, Berlin Heidelberg New York (1999)
5. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. The Art of Scientific Computing. Cambridge University Press, Cambridge (1992)
6. Webb, P.W.: Form and Function in Fish Swimming. Scientific American. **251** (1984) 58–68
7. OpenGL Performer: Silicon Graphics, Inc. Mountain View, USA
8. Dorbie, A.: Peremptory Performer: Aqua demonstration. Internet site: www.dorbie.com

Modelling Developmental Regulatory Networks

Tommy Krul[1], Jaap A. Kaandorp[1], and Joke G. Blom[2]

[1]Section Computational Science, Faculty of Science, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{tkrul, jaapk}@science.uva.nl
[2]CWI (Center for Mathematics and Informatics), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
joke.blom@cwi.nl

Abstract. This paper introduces a model for simulating regulatory networks that is capable of reproducing spatial and temporal expression patterns in developmental processes. The model is a generalization of the standard connectionist model used for modelling genetic interactions, where the terms for the regulation of gene products and the diffusion term have been separated. This model can be coupled with biomechanical models of cell aggregates and used to study the formation of spatial and temporal expression patterns of gene products during development in cellular systems.

1 Introduction

One of the most important goals in biology is to understand the chemical and cellular processes that govern the development of a fertilized egg cell to a fully-grown organism. Currently the genetic regulation and the specification of the body plan in development in a number of model organisms, as for example *Drosophila*, is known in great detail [4, 14]). The development of the body plan is typically controlled by large networks of regulatory genes. And although there are many examples available of how individual genes affect the developmental process, there are no cases available where a line of causality can be mapped from the genomic sequence to a developmental process [5]. An important issue is therefore to understand the structure and dynamics of regulatory networks. And in addition to experimental observations, the only other option to study how genetic information is translated into a spatial-temporal expression pattern that eventually gives rise to the complete morphology of the fully-grown organism, is to use simulation models.

A number of different models have been proposed to capture the notion of gene networks. And although most of these models were originally created for the reverse engineering of genetic networks [1, 2, 13, 15, 16], it is usually trivial to adapt these models for simulation purposes. To date, the most frequently used is the Boolean network model, introduced by Kauffman [11] in 1969 and used, among others, by Jackson et al. [9] in 1986 to show that pattern formation in both space and time is possible when using active feedback loops and limited

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 688–697, 2003.
© Springer-Verlag Berlin Heidelberg 2003

communication among cells. But although these methods generally possess great expressiveness, they use a high level of abstraction, which limits their biological explanatory power. Also, examining gene expression data reveals that genes usually spend a great deal of their time at intermediate concentration levels. This means that discritizing gene expression levels tends to result in a significant loss of information. Furthermore, there are some concepts in control theory that seem indispensable for genetic regulatory systems that either can not be implemented using discrete variables or that have radically different dynamic behavior. This includes amplification, subtraction and addition of signals, maintaining equilibrium using negative feedback and smoothly varying the period of a periodic phenomenon like the cell cycle. More recently, continuous models, also referred to as connectionist [12] or additive models [6], have been proposed. These models closely resemble neural-like connectionist architectures or biochemical networks of interacting chemicals [10]. Reinitz and Sharp [16] have successfully used a system of differential equations to reproduce the formation of stripes of expression of the pair-rule gene eve in the *Drosophila* embryo. Even more recently, Salazar-Ciudad et al. [17, 18] used a very similar model to: (1) compare diffusion and direct-contact induction processes as mechanisms of pattern formation, (2) identify the possible range of behavior of real gene networks and (3) suggest causal mechanisms to generate known patterns. With these models it was also demonstrated that different types of body segmentation patterns during the development of insects, a process which is basically controlled by an identical set of genes, can be explained by 'rewiring' the regulatory network. But a significant limitation of the models by Reinitz and Sharp and Salazar-Ciudad et al. is that cells are modelled as static discrete lattice sites on a grid. In for example the work by Salazar-Ciudad et al. the *Drosophila* embryo is represented by a linear cellular automaton, while in the actual developing organism the spatial and temporal patterns of gene products during development will be formed in an aggregate of cells with a usually highly complex geometry.

In this paper we describe a model for simulating regulatory networks that is capable of reproducing spatial and temporal expression patterns as found during embryonic development. The model is based on the standard connectionist model proposed by Mjolness et al. [12] and is to some extend similar in approach to the models used by Salazar-Ciudad and Reinitz and Sharp. Yet, our model does not represent cells as static discrete lattice sites on a grid. Cells are modelled as independent objects, treating the intra- and extracellular environment as explicitly separated chemical spaces. Basically this model can be combined with a biomechanical model of a cell aggregate. In our current version cells are represented as point shaped sources and sinks in the space of the extracellular matrix. As a test case we use the formation of stripes of expression of the pair-rule gene eve in the *Drosophila* embryo.

2 Methods

2.1 The Case Study: Eve Stripe Formation in *Drosophila*

As a proof of concept we tried to replicate the basic pattern forming processes that are known to occur during early embryonic bodyplan formation and selected the well- studied regulatory network for the generation of Eve stripe two of *Drosophila* for this purpose [3, 4, 14, 20]. The bodyplan of *Drosophila* is, like all arthropods, segmented. It contains 14 segments, of which three make up the head, including its antennae and mouth parts. The next three segments make up the thorax and each of these segments will develop a set of legs. The last eight segments will develop into the abdominal. The initial gradients that guide the developmental process are a result of the presence of mRNA molecules in the anterior and posterior poles of the embryo. These are deposited in the egg by the mother before it was fertilized. After fertilization the anterior mRNA molecules activate the transcription of the Bicoid protein, while the posterior mRNA activates the translation of the Nanos protein. In the initial phase of the developmental process of *Drosophila* all cells are in a syncytium. This means that cells lack a cell membrane and it is therefore that during this stage all gene products can diffuse freely throughout the embryo. Diffusion of the Bicoid proteins forms a gradient going from a high concentration on the anterior pole to a low concentration on the posterior pole. The Nanos protein forms a gradient in exactly the opposite direction, going from a high concentration on the posterior pole to a low concentration on the anterior pole. In turn a Hunchback protein gradient is formed through the activation of Hunchback by Bicoid in the anterior region and the repression of Hunchback in the posterior pole by Nanos. This leads to the formation of a high concentration of Hunchback in the anterior region with a sharp cut-off towards the posterior. Also Caudal mRNA molecules can be found throughout the embryo and as Bicoid represses its transcription a final posterior-anterior gradient is formed.

Once the main anterior-posterior gradients are formed, other proteins get activated that produce more finely tuned patterns. The Eve protein is expressed in a pattern of seven stripes. It has been shown [3, 4] that the expression of each of these stripes is regulated independently. For example, the expression of Eve stripe two is regulated by the activating and repressing influences of four proteins: Bicoid, Hunchback, Giant and Krüppel. The binding of Bicoid and Hunchback stimulates the transcription of Eve, while the expression of Giant and Krüppel have a negative influence on its translation. Right in between a gap of high levels of Giant and Krüppel the expression of the second Eve stripe finally becomes a narrow band of cells. This is also illustrated in figure 1.

2.2 The Model

Biological cells are modelled as independent objects that can absorb and exude chemical substances from and into the extracellular space. The model makes no assumption on the shape of cells and treats them as point shaped sources

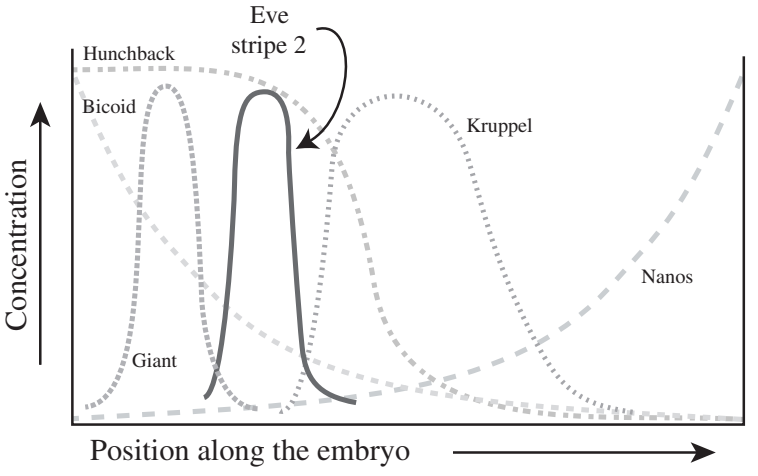


Fig. 1. The location of Eve stripe two is specified by four proteins. Bicoid and Hunchback activate Eve, while Giant and Krüppel have a repressing influence. Trapped between high concentrations of Giant and Krüppel Eve stripe two narrows down to a band of only a few cells thick. Modified after [3]

and sinks on the extracellular matrix. The extracellular matrix is represented by a grid that is overlaid on the cells and the space between them. Furthermore, protein levels are maintained within each cell. Paracrine protein levels are also maintained in the extracellular matrix, where they are allowed to diffuse. Note that we assume that these gene substances do not go into direct chemical interaction with each other. This means that we prohibit the formation of Turing-like reaction-diffusion patterns and that in our model chemical patterns can only arise through (modelled) genetic regulation. The model allows proteins to activate and inhibit each other and also makes a distinction between proteins that stay within a cell and only function for intracellular regulation (non-paracrine factors) and proteins that can diffuse into the extracellular matrix and are members of signaling pathways and exercise short- and long-range effects (paracrine factors).

For our model we adapt the equations as used by Salazar-Ciudad et al. [17, 18] to explicitly separate the intra- and extracellular environment. This separation is accomplished by removing the diffusion term from the intracellular equation and introducing extra equations that model the extracellular diffusion of paracrine gene substances. This separation does not necessarily preserve the exact dynamics of the equations as proposed by Salazar-Ciudad et al. but does preserve its pattern forming ability. The extracellular matrix is represented by a simple grid on which the diffusion equation is solved. This allows gene substances that are exuded from cells to diffuse through the extracellular matrix. The change in concentration for all intracellular gene products over time, whether paracrine or non-paracrine, is described by:

$$\frac{\partial g_{ij}(t)}{\partial t} = \frac{\phi(h_{ij})}{k_j + \phi(h_{ij})} - \lambda_j g_{ij}(t)$$

$$\text{Where } h_{ij} = \sum_{k=1}^{N_g} W_{jk} g_{ik} + h_j \text{ and } i = 1, .., N_c \text{ and } j = 1, .., N_g \tag{1}$$

Here g_{ij} is the intracellular concentration of gene substance j within cell i. Here k_j, λ_j and h_j are respectively the binding rate constant of gene j, the intrinsic rate of degradation of gene j and the activation threshold level of gene j. All enhancers and protein binding sites are described by the matrix W. Each element W_{ab} of W characterizes the regulatory effect of one gene on another by a single real number for each possible pair a and b. A positive value for W_{ab} meaning that gene b activates gene a while a negative value represses gene a. In the case that W_{ab} is zero, gene b has no effect on gene a. N_c is the total number of cells in the system and N_g is the total number of genes. ϕ is introduced to ensure that inhibiting interactions do not lead to negative concentration values. In our case ϕ is chosen to be:

$$\phi(x) = \begin{cases} x & \forall x > 0 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

In the model we make a distinction between paracrine (diffusing) and non-paracrine (non-diffusing or transcription) factors. Non-paracrine factors stay within the boundaries of a cell, while paracrine products are allowed to be transported through the cell membrane and diffuse through the extracellular matrix. The dynamics of the extracellular proteins is governed by a set of partial differential equations:

$$\frac{\partial c_j(\boldsymbol{x}, t)}{\partial t} = \mathcal{D}_j \nabla^2 c_j(\boldsymbol{x}, t) - \lambda_j c_j(\boldsymbol{x}, t) \tag{3}$$

Here c_j is the concentration of the paracrine gene product j, $\mathcal{D}_j$ is the diffusion velocity and λ_j is the rate of degradation of c_j. Equations (1) and (3) are coupled by the constraint

$$g_{ij}(t) = c_j(\boldsymbol{x}_i, t) \tag{4}$$

which defines the exchange of paracrine gene products between the intra- and extracellular environment. Here $c_j(\boldsymbol{x}_i, t)$ expresses the concentration of the extracellular paracrine gene substance j at the location $\boldsymbol{x}_i$ of cell i at time t. The implicit assumption here is that the timescale of the rate of exchange of gene substances between the extracellular matrix and the intracellular environment of a cell is much smaller than the timescale at which the extracellular diffusion of substances occurs.

For the spatial discretization of the equation (3) on the extracellular grid we use a straightforward finite difference approximation. As an integrator we

use in all cases an explicit forward Euler scheme. Because of the fact that cells can be positioned freely throughout the extracellular matrix there is no longer a one-to-one correspondence between grid points and biological cells. We therefore need to interpolate gene substances from the cell locations to the grid and vice versa. First the concentration c_{ij}^{new} of gene substance j at position $\boldsymbol{x}_i$ of cell i is determined. This is done by calculating the average of the intracellular concentration $g_{ij}(t)$ and the extracellular concentration $c_j(\boldsymbol{x}_i, t)$, which in turn is determined using linear interpolation. Then the mass m_{ij} of the absorbed or exuded substance is determined by multiplying the difference between the intracellular concentration $g_{ij}(t)$ and the newly-found average concentration c_{ij}^{new} with the volume v_i of the corresponding cell. Finally, all that is left to do is setting the new concentration values. The new intracellular concentration of gene j of cell i becomes c_{ij}^{new}, while the new concentration values in the grid can be found by linearly distributing the concentration $m_{ij}/volume_{gridcell}$ over the neighboring lattice points.

The algorithm starts by integrating the time dependent diffusion equation (3) over one time step. Thereafter the new grid values are used to update the paracrine protein concentration levels at the positions of the cells using the interpolation method described earlier (4). Then the intracellular variables are advanced one timestep using equation (1). Finally all paracrine gene substances are updated once again.

Model parameters consisted mainly of regulatory relationships between genes and activation threshold levels and were tuned to correspond roughly to known interaction properties of the regulatory network of *Drosophila*. For reasons of simplicity only a subset of the proteins known to be involved in generation of Eve stripe two were taken into account (see also figure 2). All systems were simulated until an equilibrium was reached.

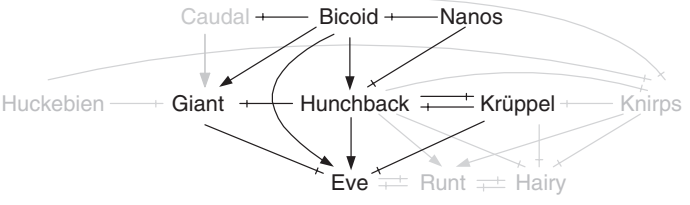


Fig. 2. A selection of important regulatory interactions between genes involved in the generation of Eve stripe two. An arrow indicates a positive regulatory relationship, while a line crossed at its end indicates a repressive relationship. Black interactions were taken into account, while interactions colored gray where not simulated. Modified after [3]

3 Results

As can be seen in figure 3 for a one- and in figure 4 for a two-dimensional domain we are able to reproduce expression patterns that resemble the specification of Eve stripe two. Differences when compared to biological data in the expression of the Giant and Krüppel protein are in part due to the fact that not all proteins involved in the specification of Eve stripe two have been taken into account. The Krüppel protein should only be expressed in the exact center of the embryo. This deviation probably comes from the absence of the repressive influence of the Knirps (kni) protein in the posterior region. Another difference with biological data comes from the Giant protein. This may be caused by the absence of the repressive influence of the Huckebien (hkb) protein on both of the poles. This last deviation is nevertheless quite significant. Giant does not form sharp borders, resulting in a band of cell expressing Eve stripe two that is too wide when compared to biological data. In two dimensional simulations this error becomes even more dominant. Yet, it is important to note here that the regulatory relationships were tuned by hand. And it should therefor come as no surprise that it is likely that a much better match can be found when using a more sophisticated parameter tuning method.

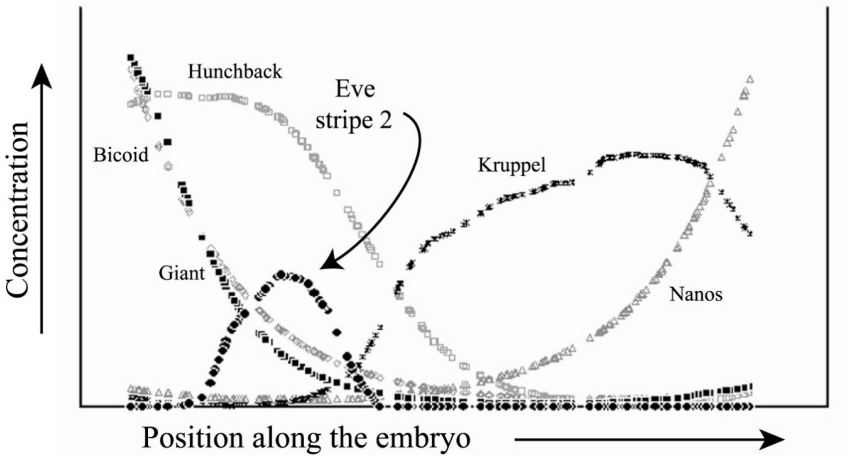


Fig. 3. One-dimensional concentration gradients showing an approximation of the specification of Eve stripe two. The Bicoid, Nanos and Hunchback gradients are shown in gray, while the Giant, Krüppel and Eve gradients are black. When comparing this figure to figure 1 some significant differences are observed. These difference can in part be explained by the absence of certain proteins that do have regulatory relationships with some of simulated proteins but were not taken into account during the simulation

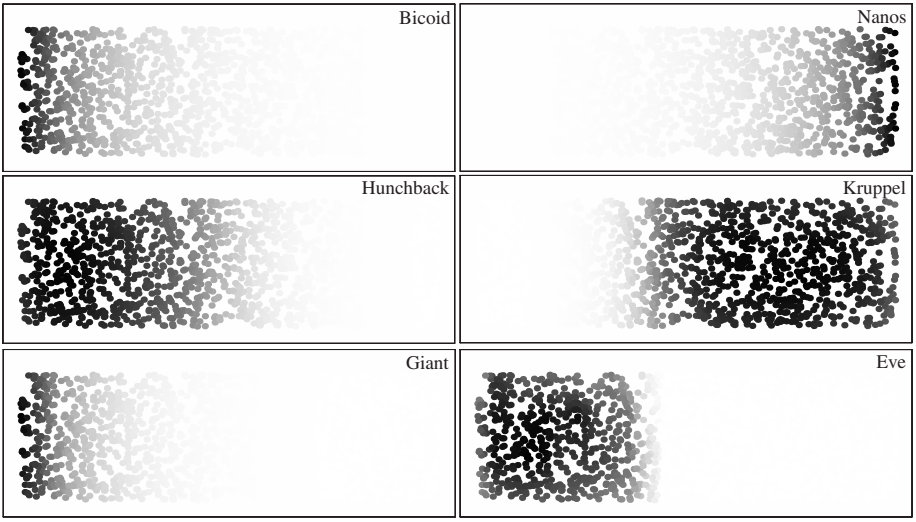


Fig. 4. Two-dimensional concentration gradients for six proteins involved in the specification of Eve stripe two. These graphs were produced using the same parameters as used for figure 3

4 Discussion

The examples in Figure 3 and 4 show that our model of regulatory networks is capable of forming relatively complex and one- and two-dimensional spatial and temporal morphogene expression patterns that correspond to morphogene gradients generated during early embryonic development of *Drosophila*. This model was coupled with a model of cells (or in the case of *Drosophila* nuclei) that can be positioned freely throughout the extracellular matrix, meaning there is no longer a one-to-one correlation between grid points and biological cells, as in many previous models [15–18]. This combination results in a more flexible model that can be used to simulate spatial and temporal expression patterns in a variety of cellular systems. The flexibility of the model mainly comes from the fact that we separated the terms that model the genetic regulation from the term that equates the diffusion of gene substances. This in contrast to many of the previous projects where connectionist models where used to simulate gene regulation [6, 12, 16–18]. Yet this simple modification allows us to perform much more realistic simulations. For example, it elegantly solves the strange correlation found in many previous connectionist models for gene interactions were a grid cell corresponds directly to a biological cell, meaning that increasing the accuracy of the generated pattern by using a finer grid, automatically results in an increase of the total number of simulated cells. In our model the biological cells and grid points are separated so that using a finer grid will result in an increased accuracy of the pattern. There is a catch however. We model cells as volume-less objects in the extracellular matrix, meaning that gene substances are

allowed to diffuse throughout the space unhindered by cells. Yet this also means that decreasing the size of the grid increases the effect of the error introduced by this simplification. We also showed that the model can be extended to more dimensions. Simulations have been carried out on both one and two-dimensional domains using exactly the same parameters for the regulatory network. Currently we are extending these simulations to three dimensions, a major problem here are the computational requirements. Because of the separation between the cells and the extracellular space it is possible to extend the model in such a way that it also models the effect of physical interactions between cells and their environment. One of our future plans is to combine our model of regulatory network for development with a bio-mechanical model of moving and interacting cells. In the simulation results shown in this paper, the model parameters describing regulatory relationships between genes and activation threshold levels, were tuned to correspond roughly to known interaction properties of the regulatory network of *Drosophila*. Currently we are investigating a method based on genetic algorithms to do a guided search through parameter spaces (see also [15, 17, 18]).

References

1. Akutsu, T., Miyano, S., Kuhara, S.: Inferring qualitative relations in genetic networks and metabolic pathways, Bioinformatics **16** (2000) 727–734
2. Brazma, A., Schlitt, T.: Reverse engineering of gene regulatory networks: a finite linear model ()
3. Carroll, S.B., Grenier, J.K., Weatherbee, S.D.: From DNA to diversity: molecular genetics and the evolution of animal design, (2001)
4. Davidson, E.H.: Genomic regulatory systems: development and evolution, Academic Press, London (2001)
5. Davidson, E.H. et al.: A genomic regulatory network for development Science **295** (2002) 1669–1678
6. D'haeseleer, P., X. Wen, Fuhrman, S., Somogyi, R.: Linear Modelling of mRNA expression levels during CNS development and ingury, Pacific Symposion on Biocomputing (1999)
7. Glazier, J.A., Graner, F.: Simulation of the Differential Adhesion Driven Rearrangement of Biological Cells , Physical Review E **47** (1993) 2128–2154
8. Graner, F., Glazier, J.A.: Simulation of Biological Cell Sorting Using a Two-Dimensional Extended Potts Model, Physical Review Letters **69** (1992) 2013–2016
9. Jackson, E.R., Johnson, D., Nash, W.G.: Gene networks in development, Journal of Theoretical Biology **119** (1986) 379–396
10. Kaneko, K., Yomo, T. Isologous deversification: a theory of cell differentiation. Bull. Math. Biol. **59** (1997) 139–196
11. Kauffman, S.A., J. Theoretical Biology **44** (1974) 167–190
12. Mjolness, E., Sharp, D.H., Reinitz, J.A.: connectionist model of development. J. Theoretical Biology **152** (1991) 429–453
13. Murphy, K. and Mian, S., Modelling gene expression data using dynamic Bayesian networks, Technical report, Computer Science Division, University of California, Berkeley, CA. (1999)

14. St Johnston, D, Nüsslein-Volhard, C.: The origin of pattern and polarity in the *Drosophila* embryo, Cell **68** (1992) 201–209
15. Reinitz, J., Kosman, D., Vanario-Alonso, C.E., Sharp D.H.: Stripe forming architecture of the gap gene system. Developmental Genetics **23** (1998) 11–27
16. Reinitz, J., Sharp, D.H.: Mechanism of formation of eve stripes. Mechanisms of Development **49** (1995) 133–158
17. Salazar-Ciudad, I, Newman, S.A., Sole, R.V.: Phenotypic and dynamical transisitions in model genetic networks I. emergence of patterns and genotype-phenotype relationships, Evolution and Development **3** (2001) 84–94
18. Salazar-Ciudad, I, Sole, R.V., Newman, S.A.: Phenotypic and dynamical transisitions in model genetic networks II. application to the evolution of segmentation mechanisms, Evolution and Development **3** (2001) 95–103
19. Turing, A.M., The chemical basis of morphogenesis, Transactions R. Soc. Lond B. **237** (1952) 37–72. Reprinted in Bull. Math. Biol. **52** (1991) 153–197
20. Wolpert, L., Beddinton R., Brooks J., Jessel, T.: Principles of development, Current Biology Ltd, Oxford University Press (1998)

A Biomechanical Imitative Model for Pulsatile Morphogenesis

L.V. Beloussov and V.I. Grabovsky

Faculty of Biology, Moscow State University, 119992, Moscow, Russia
lbelous@soil.msu.ru

Abstract. We propose a model which imitates the morphogenesis of several species of the hydroid polyps and permits the derivation of the geometry (surface curvature) of each developmental stage from that of a preceding stage. The model is based upon two experimentally verified assumptions. First, neighboring cells are assumed to compress each other laterally in a regular and species-specific pulsatile manner. It is this pressure, and/or an active cell reaction to it, which changes the curvature of a cell layer. Secondly, cell layers are assumed to have quasi-elastic properties tending to smooth out their curvature. With our model, the different pulsatile patterns of cell-cell pressure are reproduced and the elasticity parameters are modulated. The main principles of the model can also be used for interpreting the morphogenesis of other groups of animals. A suggested model emphasizes the self-organizing properties of a "stressed geometry" of embryonic rudiments.

1 Introduction

The aim of this work is to suggest, by means of an imitative modeling, some “rules of morphogenesis”, that is, more or less universal algorithms permitting the derivation of each next stage embryonic shape from a preceded one. Contrary to many up to now developed models (e.g. [1]) we use the approach which can be called geometro-mechanical. This means, that we use the formative properties of the mechanically stressed bodies of a definite geometry (curvature of a surface). Such an approach is supported by the existense of the regularly arranged and stage-dependent mechanical stresses, both tensile and pressure ones, within embryonic tissues [2]. In this paper, a geometro-mechanical approach is used for interpreting the formation of buds in several species of lower invertebrate animals called hydroid polyps, or hydroids. Hydroids buds start their development from very simple geometrical shapes which become complicated later on in a quite monotonous manner, by changing nothing more than their local curvatures. Their shapes specificity is provided by quite regular temporal patterns of so called growth pulsations (GP), that is, the periodical (taking place each several minutes) extensions-retractions of the growing tip. At the extension GP phase a lateral cell-cell pressure of a turgor origin is substantially increased. At the retraction GP phase which always follows the extension phase the intracellular turgor pressure is diminished and so does the lateral cell-cell pressure [3].

We imitate the development of three hydroids species: *Dynamena pumila., Obelia loveni and Campanulina lacerata.* In all the mentioned species the shapes of

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 698–707, 2003.
© Springer-Verlag Berlin Heidelberg 2003

just initiated buds can be roughly approximated by a hemispherical surface adjoining a cylindrical stem. The growing buds increase their transversal diameters to a different extent. This increase is the greatest in *Dynamena* samples, not so great in *Obelia* and almost absent in *Campanulina* whose buds remain quite narrow. As observed in a longitudinal projection, the buds always take a trefoiled (tripartite) shapes, with one central and two lateral protuberances. While in *Obelia* and *Campanulina* samples the lateral protuberances are not so pronounced and contribute to nothing more than tentacles, in *Dynamena* samples they develop to a much greater extent, deviate from a central lobe and produce new hydranths.

GP specificity is expressed in the duration ratios of the extension/retraction phases [4]. This ratio is the greatest in *Dynamena* samples, takes smaller values in *Obelia* colonies and is the smallest in *Campanulina* (Fig. 1). Correspondingly, we define the *Dynamena* GP as broad ones, those typical for *Obelia* as intermediate and those exhibited by *Campanulina* as narrow. The GP patterns correlate in a simple manner with the shape of the arisen buds: the broader are GP, the greater the rudiments are transversely extended, and vice versa. One of the main aims of our model will be to reproduce these correlations.

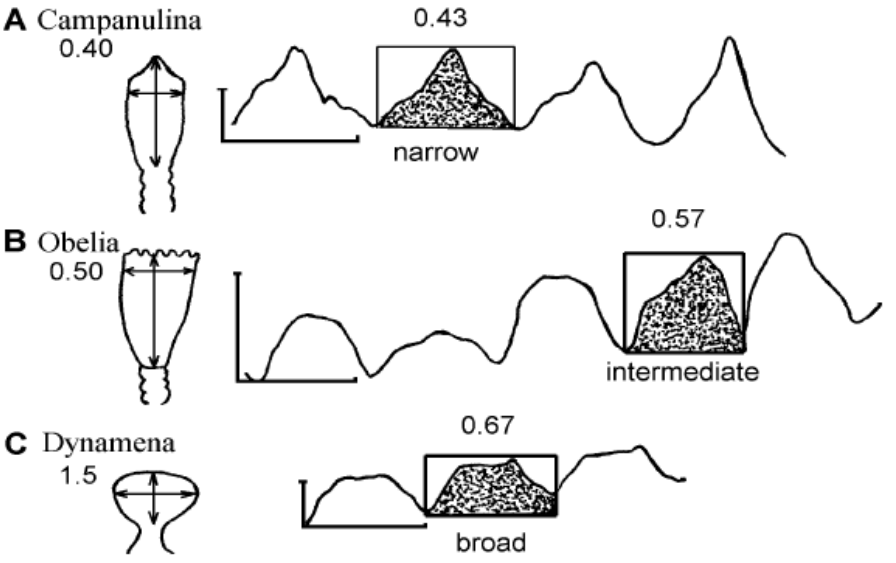


Fig. 1. A correlation between shapes of the *Campanulina* and *Obelia* hydranths (A, B) and *Dynamena* growing tip (C) on one hand and the patterns of their GP on the other. Left figures show transversal/longitudinal diameters ratios of the corresponding rudiments. Figures above GP graphs give the ratios of the hatched squares to the rectangular ones. This is a rough measure of a "broadeness" of GP.

2 Description of the Model

2.1 Main Assumptions

The model is based upon the following suggestions derived from the biological data:

1. A surface of a rudiment consists of a finite number N of kinematically independent units, each of them corresponding to a group of 2 – 4 cells of a real rudiment. We take this value because it fits the smallest diameters of the morphological elements (protuberances, invaginations) produced during morphogenesis of the given species (see also Results).
2. Any element *i* is compressed laterally by its neighbors, *i-1* and *i+1*. In the case of a non-zero local curvature, the pressure force applied to *i* acquires a radial component directed towards increasing the local curvature. Therefore, *i* should be shifted centrifugally. Correspondingly, we introduce either a purely passive, or the passive-active centrifugal force f_k. (Fig. 2A).
3. We assume that f_k is increased during the GP extension phase and is decreased (up to negative values) during the retraction phase. We express this effect by introducing a coefficient K of a force f_k. We also schematize GP patterns by dividing each model GP into 5 steps and ascribing to each step the different K values. We explore separately the effects of a constant (non-pulsating) regime and of three above mentioned types of pulsations: broad, intermediate and narrow. They were approximated by the following sets of K values:

Constant regime	1;1;1;1;1;
Broad pulsations	(in most cases) 1;1;-0,2;1;1
Intermediate pulsations	-0,2;0,3;0,8;1;0,7
Narrow pulsations	-0,2;0;0,2;1;0

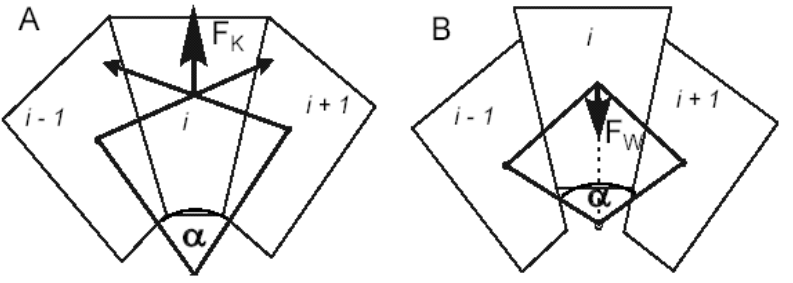


Fig. 2. The forces which increase and smooth out the curvature. A: a curvature-increasing force F_k acting upon a cell *i* due to the lateral pressure exerted by the neighboring cells *i-1* and *i+1* . B: a quasi-elastic force F_w acting to any cell of a layer independently of its neighbors.

4. At the same time, we assume that a quasi-elastic force f_w exists which is constantly (in a non-pulsating manner) applied to an entire cell layer and directed towards the diminishment of a local curvature (Fig. 2 B).
5. Due to a high viscosity (friction) of the rudiments' tissues there are no accelerations within a system and the forces are taken proportional to the linear shifts Δ of the elements per time unit:

$$\Delta_i = f(f_k, f_w)$$

where Δ_i is the resulted shift of an element i and f is the function connecting the value of Δ_i with those of the forces f_k and , f_w.

2.2 Technical Description of the Model

The model is based upon a 2-dimensional set, its dimensionality taken as [2, *No**], where *No* is a number of kinematically independent units within a model rudiment. Each element of a set [*a, b*] contains a coordinate *a* (*x* or *y*) of an element *i*. For example, an element [1, 12] contains an X-axis coordinate of an element No12 within an euclidean space, while an element [2, 12] contains an Y-coordinate of the same element.

We take two initial configurations (arrangement of elements), both of them 2-dimensional (as exemplifying medial longitudinal sections through the rudiments' bodies), which we call cylindrical (Fig. 3A) and curved (Fig. 3B). In the both cases a first 1/3 of the elements' set is used for constructing a "left" wall of a rudiment. The second 1/3 creates a "cap", namely, a semi-circle, its beginning being joined with the end of a "left" wall while its end being joined with the beginning of a right wall. The last 1/3 contributes to the right wall itself. In the case of a cylindrical configuration the both walls are straight lines.

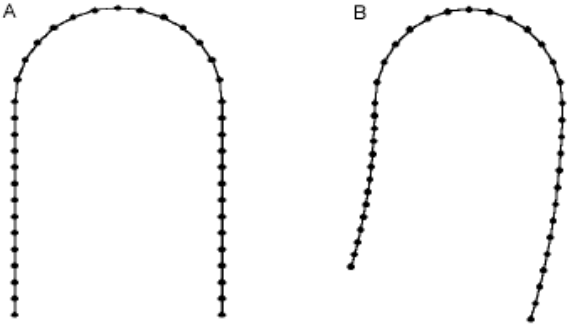


Fig. 3. Cylindrical (A) and curved (B) initial rudiments

In the case of a curved rudiment the left wall is an arch of a radius *R*, while the right one an arch of a radius $R_1 = R + D$, where *D* is a diameter of a "cap". Usually we take $R : D = 4$.

Then, according to the model algorithm we are transforming the coordinates of each element. The algorithm includes three following steps:

1. Change of an element's coordinate according to: (a) its momentary coordinate; (b) the coordinates of two neighboring elements, and (c) the value of *K* coefficient given as a parameter;
2. Correction of the newly obtained coordinates in relation to: (a) the coordinates of an average position of the given element and the two neighboring ones; (b) the value of a *W* coefficient given as a parameter;
3. Rearrangement of the obtained coordinates for avoiding the distances between neighboring points to override the limits of an initially taken interval. This is a condition limiting the mutual cells shifts within a real rudiment.

When thus obtained new set of coordinates of a given element replaces the initial one, the first step of a GP (out of 5) is fulfilled. Then the program returns back to point 1 for fulfilling the second step of GP, etc. After performing all the 5 steps, the program may be either stopped or starting a new GP. The number of GP can be set automatically installing a parameter *No of steps*.

At each step of the program, the transformations of the coordinates obey the following formalisms (Fig. 4A).

Step 1 can be generally expressed as

$$(i)'_{x,y} = f_1\{(i)_{x,y};\ (i-1)_{x,y};\ (i+1)_{x,y};\ K_n\}$$

where

$(i)_{x,y}$ are coordinates of an element i (i = 1,2,3… *No*);

K_n is the parameter of each step of a pulsation (n = 1,2,…5);

f_1 is the function connecting the values;

In our model we take the function f_1 as

$$(i)'_{x,y} = (i)_{x,y} - S \cdot K_n \left\{ \frac{(i-1)_{x,y} + (i+1)_{x,y}}{2} - (i)_{x,y} \right\}$$

Where

$$S = \frac{[i-1, i+1]}{[i-1, i] + [i, i+1]}$$

(in square brackets the euclidean distances between the points are given).

The parameter *S* is introduced for avoiding too sharp bending of a rudiments' surface. As one can see, under slight bending it is close to 1 while under increased bending it approaches zero.

After fulfilling this step, the initial elements' coordinates (i) are exchanged by the new ones (i)'. This step imitates the main model assumption, namely, the passive-

active movement of each element of a rudiment's surface towards the local curvature increase.

Step 2: corrections of the new coordinates (*i*)' according to the value of *W* parameter and the initial coordinates of the point *i* and its nearest neighbors (*i-1* and *i+1*). Formally the corresponding transformations of the coordinates can be expressed as

$$(i)''_{x,y} = f_2\{(i)'_{x,y};\ (i)_{x,y};\ (i-1)_{x,y};\ (i+1)_{x,y};\ W\}$$

The function f_2 looks like

$$(i)''_{xy} = (i)'_{xy} - W \cdot ((i)'_{xy} - (\overline{\overline{XY}}))$$

where $\overline{XY}$ are the averaged coordinates of all the three points: *i, i-1* and *i+1* (the corresponding point is denoted by a triangle, Fig. 6B). This step imitates the mechanical elasticity of a rudiment expressed by *W* parameter. The greater be its value, the more elastic should be the rudiment's tissue (Fig. 4B).

Step 3. Now the distances between the neighboring elements are tested and if they do not correspond to the formal limits the coordinates are rearranged.

The program language is Visual Basic 6.0. The program was implemented on IBM PC.

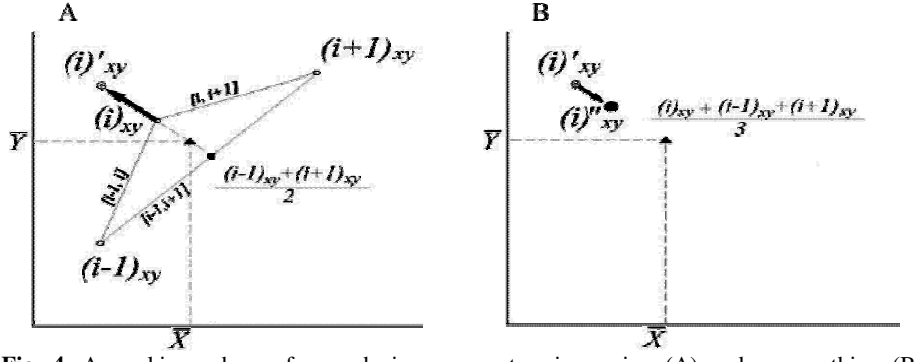


Fig. 4. A working scheme for producing a curvature-increasing (A) and a smoothing (B) component of the cell shifts.

3 Results

Fig. 4 exemplifies a generalized scheme of shape formation under different W values and different pulsatile regimes. Firstly, one can distinguish two different regimes with respect to the effect of the *W* values on the shapes produced. With low *W* values the trajectories of the imaging points cross each other, so that the points rapidly converge into dense "spots" (Fig. 5 C, G, L). This corresponds to the early developmental

stages of some embryos, but not to those which morphogenesis we aim to model. On the other hand, under very high elasticity values the initial shapes always gradually unfold towards a stationary state of a slightly concave line (Fig. 5H), or give a "sand-glass" shape (Fig. 5M) which again has little or no relation to the real morphogenesis. There are the intermediate zones of *W* values (correlated with pulsatile patterns) into which the required shapes are fairly well reproduced. We call them the "morphogenetic zones", MZ (densely outlined in Fig. 5). The shapes reproduced within MZ, are rather sensitive to the pulsatile patterns, and this sensitivity fits the empirical data (4). The shapes produced under constant regime (Fig. 5A,B) are rather poor and non-realistic. Those employing "broad" pulsations possess the apically extending rudiments (Fig. 5 D, E) and wide stems which are straight even if emerged from the curved rudiments (Fig. 5 F). This is typical for *Dynamena*. The "intermediate" pulsations give the rudiments without an apical extension (Fig. 5 J) and having S-shaped stems (Fig. 5 K). Those are features characteristic for *Obelia*. Lastly, under "narrow" pulsations we get quite narrow shapes (Fig. 5 N, N_1) which are straight even if produced by curved rudiments (Fig. 5 O). It is remarkable, that the shapes emerged from the axi-symmetrical initial rudiments rapidly (after a restricted number of iterations) come to the stationary states (this is illustrated by thin contours of the frames D, E, I, J, N_1). On the other hand, the shapes produced by the curved rudiments exhibit more prolonged "growth" (frames B, F, K, O) which is just typical for the stems of hydroid polyps. Some more details of the shapes formation under different pulsatorial regimes are given in Fig. 6. As one can see, the resulted shapes are quite sensitive to the changes in the pulsatorial regimes, especially, to the changes in the amplitude of "troughs": surprisingly, an increase or a decrease of a single trough within GP can produce rather global shape transformations. Meanwhile, the modeled shapes are biologically realistic.

Another important developmental parameter (not included in Fig. 5) is associated with the kinematically independent units. We found, that the morphogenesis is quite non-sensitive (robust) to the total number of such units: the initial rudiments cut from below and including slightly more than the apical hemisphere exhibit roughly the same morphogenesis as the complete rudiments.

That corresponds to the high regenerative capacities of hydroid polyps, which are able to reconstruct roughly the same shapes from little pieces of their tissue. On the other hand, the morphogenesis turned out to be very sensitive to the density of the units' arrangement: under high density (single units are of the scale of the single cells diameters) the shape changes are going slowly, so that quite monotonous wavy patterns are produced. In the opposite, if the units are too large, all the rudiments, irrespective of the pulsatorial regimes, come to smooth spheroid shapes. We speculate, that the optimal density of the kinematically independent units (employed in our model) is partly determined by the inherent properties of a tissue, and partly self-selected within the course of morphogenesis, as providing a most rapid formation of macroscopic shapes.

We are well aware that our model is up to now very much simplified as compared with the biological reality. Obviously, a number of the control parameters should be increased. One can also assume that they are evolved during development We suggest however that the main assumptions of the above presented model will be included in these, more perfect constructions.

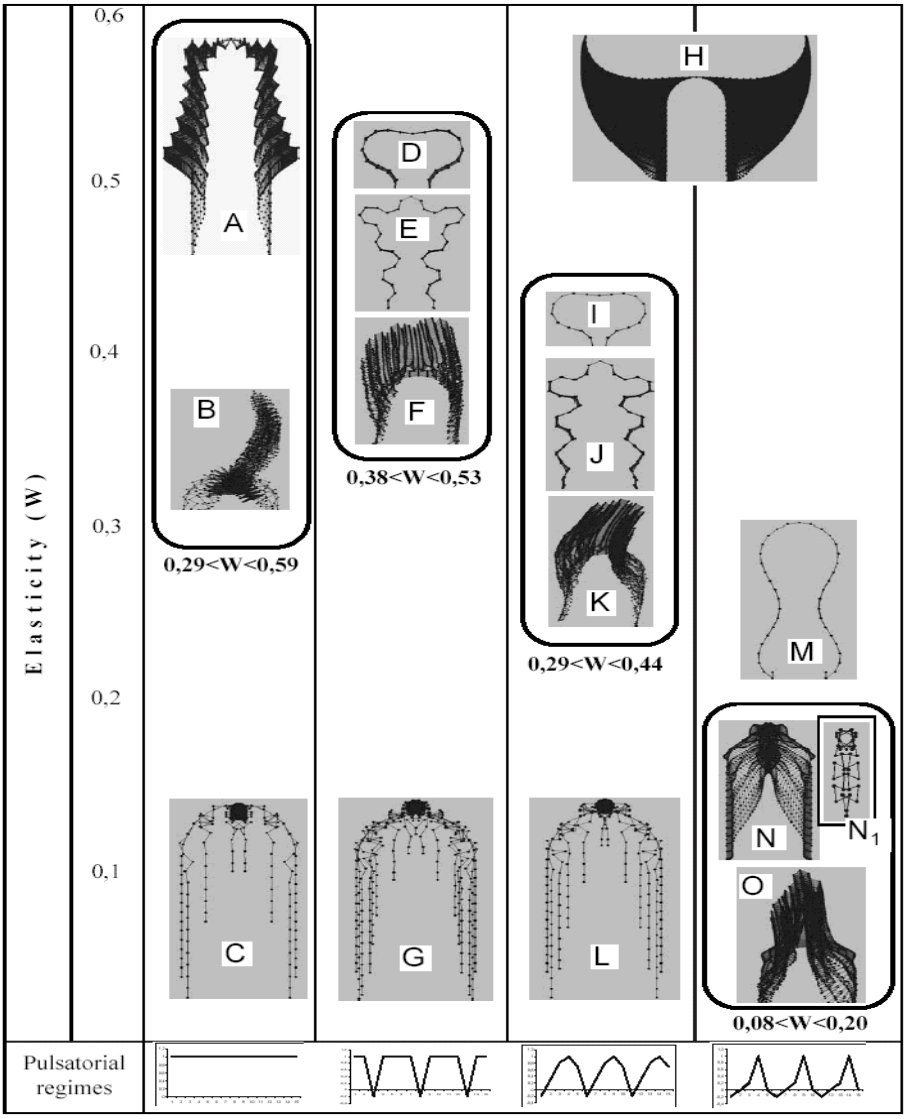


Fig. 5. Generalized scheme of the main model shapes, as depending upon the pulsatorial regimes and the quasi-elasticity of tissues. Columns from left to right: constant regime, broad, intermediate and narrow growth pulsations (imaged below). Vertical axis: values of a quasi-elasticity coefficient (*W*). Densely outlined are the morphogenetic zones. Within MZ, the lowest frames B, F, K, O show the typical resulted shapes of the curved rudiments, while the upper shape(s) A, D, E, I, J, N, N_1 are those of cylindrical rudiments. Frames D, E, I, J, M, N_1 (shown by thin lines) give just the final configurations, all of which are stationary. On the contrary, black frames give the apposition of all the intermediate configurations.

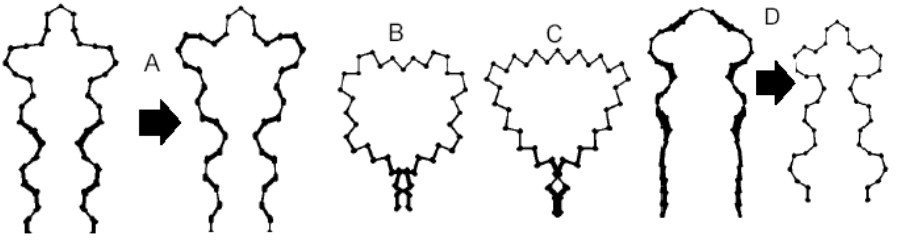


Fig. 6. Shapes modeled within MZ under different pulsation regimes if starting from a cylindrical configuration. A: "broad" GP. Left frame: 50 iterations (=10 GP), right frame: 75 iterations (=15 GP), stationary configuration. B: "broad" GP, but trough at the 3rd step is changed from –0,2 to +0,2. C: same" trough is +0,3. Note the change of an arisen shape from quasi-quadrangular to quasi-triangular. These are observed in real *Dynamena* colonies. For A–C W = 0,48. In all the cases *Dynamena*-like shapes are obtained. D: "intermediate" GP, producing *Obelia*-like shapes. W = 0,39. Left frame: 5 iterations, right frame: 15 iterations, stationary configuration.

4 Conclusions

The model presented is related to one of the most fundamental biological problems, namely to the conceptual gap between genetic factors which influence shape formation and the shapes themselves. The main difficulty here is in understanding, how the genetic factors which are *per se* uniformly distributed within embryonic space, are able to control both local and holistic morphological (that is, spatially non-uniform) properties of the developing organisms. One can see, that the both main parameters used in our model (elasticity W and the pulsatile patterns) are indeed spatially uniform and may be hence directly linked with genetic factors. As to the pulsatile patterns, this is shown directly (5). Same seems to be plausible in respect to the dimensions (density) of the kinematically independent units. Meanwhile, as shown by the model, such values can largely affect numerous morphological features. We suggest therefore, that the genetic control of development can be considered as a parametric modulation of some universal morphogenetic rules.

Acknowledgements. This work was supported by the Russian Fund for Fundamental Investigations (RFFI), grant # 02-04-49004.

References

1. Meinhardt, H.: Models of Biological Pattern Formation. Academic Press, London (1982).
2. Beloussov, L.V., Saveliev, S.V., Naumidi, I.I., Novoselov, V.V.: Mechanical Stresses in Embryonic Tissues: Patterns, Morphogenetic Role, and Involvement in Regulatory Feedback Int. Rev. Cytol. **150** (1994) 1–34.
3. Beloussov, L.V., Labas, J.A., Kazakova, N.I., Zaraisky, A.G. Cytophysiology of Growth Pulsations in Hydroid Polyps. J. Exp. Zool. **249** (1989) 258–270.
4. Beloussov, L.V., Labas, J.A., Kazakova, N.I. Growth pulsations in hydroid polyps: kinematics, biological role and cytophysiology. In: Rensing, L. (ed) Oscillations and Morphogenesis. Marcel Dekker, N.Y., Basel, Hong Kong (1993) 183–193.
5. Wyttenbach, C.R.: Genetic Variations in the Mode of Stolon Growth in the Hydroid Campanularia flexuosa. Biol. Bull. **137** (1969) 547–556.

Using Computational Plant Science Tools to Investigate Morphological Aspects of Compensatory Growth

David Thornby, Michael Renton, and Jim Hanan

Centre for Plant Architecture Informatics, University of Queensland, Brisbane, QLD
{dft, mrenton, jim}@cpai.uq.edu.au
http://www.cpai.uq.edu.au

Abstract. Models of cotton plant architecture expressing several physiological hypotheses about plant resource use and responses to damage are incorporated in the traditional research cycle to investigate the phenomena of compensation for defoliation. Two separate approaches to modelling the uptake and allocation of carbon are used: a detailed bottom-up physiology model expressing ideas about local control, and a top-down, canonical approach where qualitative knowledge about plant responses to defoliation are modelled as flows between plant physiological compartments. The two models provide contrasting methods for developing explanations for the underlying pattern of responses observed in the plants.

1 Introduction

The basic tool in a computational plant scientist's tool kit for studying biological effects related to plant architecture is a system for expressing models of plant development on a component-by-component basis. L-systems [1]allow such models to be expressed. Well-established techniques for developing models from relationships observed in real plants [2] are employed. These models can be used to express multiple hypotheses about plant behaviours, producing visual simulations and data output that capture emergent properties of the interactions among a number of hypotheses. We use this approach to investigate aspects of plant resource use and damage responses. This paper will show how the addition of a simulation phase to the normal investigative protocol of hypothesis and experimental phases aided in developing our understanding of responses to defoliation in cotton. We also aim to illustrate some of the differences between canonical and explicit approaches to modelling plant physiology, highlighting strengths and weaknesses of each.

Background It is well established that cotton crops have a high ability to compensate for significant levels of damage of various forms [3, 4], including defoliation [5, 6], particularly when damage occurs early in the season. That is, cotton crops are able to produce 'normal' yields (seed mass, lint mass and quality) following the removal of quite significant quantities of leaf tissue. A

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 708–717, 2003.
© Springer-Verlag Berlin Heidelberg 2003

considerable body of work quantifying crop-wide yield compensation under a variety of conditions already exists. However, the morphogenetic mechanisms of the compensatory response and the architectural variables of the response (for example, leaf area or stem length changes) are not well understood.

We used L-systems [1, 7] to develop our models of cotton growth and damage responses. Full text of the model code and visualisations are available at www.cpai.uq.edu.au.

2 Initial Model

An initial model of cotton seedlings was developed to aid in the generation of morphological hypotheses concerning main stem extension in cotton and its response to defoliation. This initial model links hypothetical physiological mechanisms with morphological outcomes. Leaves near the top of the plant tend to supply their photosynthetic output upwards, towards the apex [8]. Therefore, the central hypothesis of the model is that leaf photosynthetic output near the top of the plant makes a significant contribution to the development of new metamers, defined here as an internode, leaf, and axillary bud, by the apex (Hypothesis 1). Consequently, any significant defoliation near the top of the plant would affect the period of time between the appearance of successive metamers of the main stem (the plastochron), slowing development (Hypothesis 2). It has been shown that plants may respond to defoliation by increasing photosynthetic output from remaining or newly produced leaf tissues for a period of time [9, 10] (Hypothesis 3). The expansion of new leaves and internodes may also at some point be limited by reduced availability of photosynthate (Hypothesis 4) [9, 10].

Apical development In the model, apical development is controlled by a resource pool. The upper leaves contribute to the pool, while apical development consumes the resources, once the pool has passed a threshold value. This process may occur multiple times in one time step (a day), to allow for development of more than one set of parts. As new parts are produced, they are assigned a number identifying their node of attachment on the main stem.

Mechanism for accumulating photosynthate to fuel apex development The amount of resources each leaf adds to the apex resource pool is a function of the size of each leaf, its proximity to the apex, and its current rate of photosynthesis. Thus, each leaf within a user-defined distance from the apex contributes resources to the pool. The apex checks its resource pool after all the relevant leaves have contributed resources for the day.

Mechanisms for modelling the effects of defoliation In the defoliated model, leaf area is removed in the appropriate locations and times, and a signal from the wound site causes a rate modifier to be increased. After a user-defined time period, it is reset. The rate modifying variable alters the rate of photosynthesis and of expansion of new leaves and internodes, ultimately changing their fully expanded size.

Initial model output Morphological predictions were made from the initial model for leaf area, plant stem length, and internode number (Fig 1). Consis-

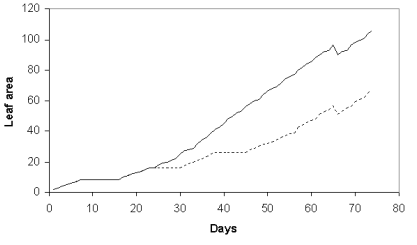
tent differences between the two plants are apparent, in particular, for leaf area per plant. The difference becomes apparent soon after defoliation, and a reduction in rate of increase in leaf area around steps 38-45 is primarily responsible for the significant difference throughout the rest of the simulation. Significantly, the increase in photosynthetic output following defoliation results in the plant developing new metamers more rapidly for a period of time (from defoliation to about step 32). Then, as the photosynthetic advantage is lost, the defoliated plant's plastochron increases compared to the control plant, with the result that although the plastochrons are later identical, the defoliated plant consistently lags behind. These observations led us to derive new morphological and morphogenetic hypotheses:

Hypothesis 5: Defoliated plants will manifest reduced leaf area at the end of the experiment compared to control plants.

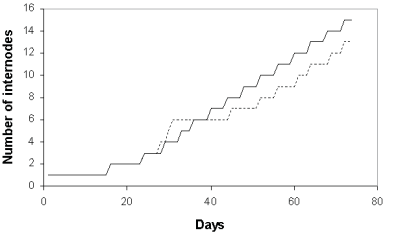
Hypothesis 6: Defoliated plants will manifest reduced main stem length at the end of the experiment compared to control plants.

Hypothesis 7: That there would be a period where the main stem plastochron of defoliated plants would be shorter, then longer, than for undefoliated plants, resulting in fewer main stem nodes on defoliated plants.

Hypothesis 8: That leaves and internodes produced soon after defoliation would be smaller than the corresponding components on undefoliated plants. An initial experiment was designed to test these morphological hypotheses.



a)



b)

Fig. 1. Initial model predictions: a) Leaf area per day per plant. b) Number of main stem internodes per plant. Solid line: control plant; dotted line: defoliated plant

3 Experiment

Materials and Methods The experimental design consisted of a control group and three treatments: the removal of one of the cotyledons, the first true leaf, or both a cotyledon and the first true leaf. Twelve plants were subjected to each treatment. *Gossypium hirsutum* 'Sicala 40' was chosen due to its upright growth habit, reducing 'lodging' of the plants as they mature, (L. Wilson, pers. comm.) and because it shows architectural characters broadly typical of normal-leaf cotton varieties.

The plants were grown in a glasshouse, providing approximately 80% of full outdoor irradiance and photoperiod of approximately 15 hours day/9 hours night. Relative humidity inside the glasshouse was maintained at approximately 60%. The plants were sown into plastic containers (20 cm diameter, 20 cm deep) containing a medium made up of composted pine bark (60%), coarse sand (20%) and peat (20%).

The plants were allowed to grow until the majority of first true leaves were fully expanded, at which point the vigorously growing, straight-stemmed plants were assigned randomly into treatments. Defoliation was of whole laminae, using secateurs, and petioles were left intact. Plant architectures were measured using a SAC Freepoint G12 sonic digitiser and Floradig software.

Results There were no significant differences in main stem node number between treatments, either during (Fig 2) or at the end of the experiment (Fig 3). Leaf area (Fig 3 b) was not significantly affected by any of the treatments applied, thus showing full compensation.

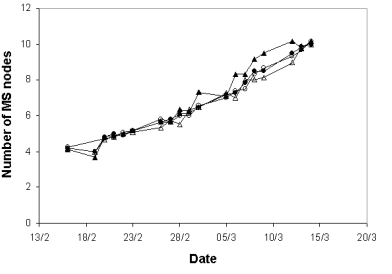


Fig. 2. Mean number of main stem internodes present per day during the experiment. • control plants; ∘ remove one cotyledon; △ remove first true leaf; △ remove both

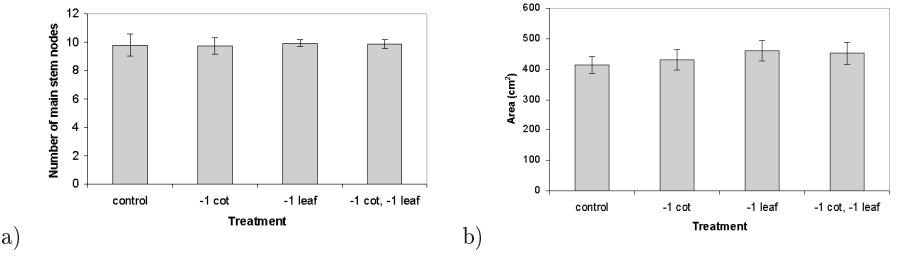


Fig. 3. Plant growth at the end of the experiment: a) Mean number of main stem internodes per plant. b) Mean lamina area. Error bars are one standard error

Discussion Several aspects of the cotton plants' growth and development did not fit with hypotheses generated using the initial model. Firstly, main stem plastochron was not perturbed by defoliation, at least at the levels tested in experiment one. It follows that main stem meristem activity is either not carbohydrate-limited, or not limited at the levels tested in experiment one. Secondly, the size of new leaves and internodes was not simply reduced by defoliation, but increased by up to 20% for a short period of time following the removal of true leaves. Conversely, the removal of a cotyledon at the seedling stage decreased the size of newly produced leaves and internodes for a short period. This may have been due to reduction in supply of some nutrient or nutrients from the cotyledons, or a change in the production of some regulatory signalling compound. Leaves and internodes at positions 3 and 4 were not fully expanded at the time of defoliation, therefore should have exhibited a size difference between treatments once fully expanded. The first metamers with different sizes were not measured as present at the time of defoliation, except as primordia. Leaves already in the rapid expansion phase were not affected by the signalling and/or resource changes caused by the defoliation treatments.

Thirdly, for leaf area and main stem length, the treated plants were able to compensate fully - that is, no significant differences were found between treatments for these morphological characters. Therefore, main stem development and growth rates (and thus absolute growth rate in pre-flowering, unbranched seedlings) are not a function of the amount of leaf area left on the plants after defoliation. Where the initial model predicted that defoliated plants would be unable to carry as large a reproductive biomass as undamaged plants [11], the data from the experiment suggests otherwise.

4 Improved Physiologically Explicit Model

To capture the results of the initial experiment, either quantitative (parameter changes to existing mechanisms) or qualitative (changes of mechanism) changes must be made. Reasonable quantitative changes did not allow the model to simulate the responses observed in the data; in order to have the model compensate fully, the rate modifier needed to be adjusted upwards to above 2, and to last for several weeks - well beyond any level likely to be observed in real plants. Within the experiment's limits, the main stem appears to develop obligately (according to a predetermined plan), not facultatively (according to current resource levels). This appears to be a significant contributor to the ability of these plants to compensate for defoliation in the tested indices such as leaf area (Fig 3 b). Therefore, we made qualitative changes to the physiological hypotheses underlying the model, by replacing the explicit mechanisms for plastochron dependence on levels of substrate carbon (Hypotheses 1-3) by the hypothesis that the plastochron is determined only by accumulated day-degrees [7].

Changes made to the initial model The value of the day-degree requirement of each successive main stem plastochron is determined using values appropriate to the current length of the main stem, in node numbers. While carbohydrate

availability was removed from the decision-making process concerning main stem meristem activity, carbohydrate production, transport and utilisation are used as important drivers of the successful development of plant parts in the new model. Local levels of available carbohydrate are stored for each main stem node position. Substrate carbohydrate is produced in the leaves according to grams per square centimetre values for cotton leaves given by Constable in his examinations of transport of radioactive-labelled carbon in cotton plants [12, 13]. Leaf productivity increases to a peak at about 7 days after production, and then declines with age [12].

Carbohydrate is transported upwards and downwards between main stem node positions, with values transported in each direction determined by each node's distance from the apex.

Future potential maximum sizes of parts are determined in the primordial phase. When the apex develops, it produces a primordium, rather than a fully differentiated set of plant organs (internode, bud, and leaf). The primordium carries a parameter that affects the size of the leaf and internode that it will eventually become. For each day that perturbed conditions exist in the plant, the potential maximum size of future metamers is adjusted.

Successful expansion of new parts (leaf, internode, flower bud, boll) to full size depends on local availability of substrate carbohydrate. Reproductive structures have the highest priority over local supplies of substrate carbohydrate and will be abscised if insufficient carbohydrate is available over a length of time.

Output from the improved explicit model By changing the model in the ways specified above, the plant is able to compensate more or less fully for defoliation applied in the simulation. The results for leaf area (Fig 4) show high levels of compensation for defoliation. Number of main stem nodes over time is exactly the same for each treatment.

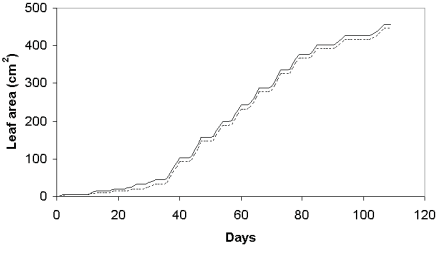


Fig. 4. Leaf area per plant, predicted by improved physiologically explicit model. Solid line: undefoliated; dotted line: two leaves removed at start.

Discussion Some aspects of the initial physiology model remain: notably, the effect of defoliation on the rate of photosynthesis. This illustrates one of the shortcomings of the morphological approach to experimentation: that is, we did

not measure the rate of photosynthesis and so could not evaluate any changes to photosynthetic output following defoliation. Therefore, while our morphological data did not dispute the theory of an increased photosynthetic rate, neither do we have direct information on the phenomenon. The physiologically explicit model leads us to believe that such a mechanism is at least possible, and indicates that it would be a partial explanation for the level of compensation found in the experimental results. We therefore retained it as one of the physiological hypotheses present in the model. However, we were not able to improve that aspect of our model by, for example, more accurate (data-supported) estimation of rate parameters. The parameters for changing the value of the rate modifying variable, and for its effects on the model plant's behaviour, remain fitted by hand.

There are two alternatives for improving our ability to produce numerically accurate simulations of the interactions between cotton's physiology and morphogenesis: either improve the physiologically explicit model by gathering more directly physiological data, or model the physiology at a more abstract level.

5 The Canonical Model

Where we encountered a need for missing pieces of information on aspects of cotton's physiological processes, the construction of a physiologically explicit model became difficult and less rigorous than we would have liked. In order to address this, we decided to model the physiology at a more abstract level [14]. The canonical approach [15] is particularly useful for modelling the qualitative and quantitative behaviour of physiological systems where detailed knowledge about the system is unavailable, when gaps exist in detailed knowledge, or when a detailed physiological mechanism is not needed. Similar canonical models have been effective in modelling birch tree responses to damage [15], forest stand dynamics [16, 17] and dry matter partitioning in trees [18].

Construction First we decided which variables would be important for our model and assigned each one a 'compartment' x. We then used a series of differential equations to represent flows of resources between compartments. The equations are of a simple, robust power law form: $flux = kx^h$, where k and h are fitted constants.

In our case, the compartments and flow equations represent different aspects of the carbohydrate physiology system in cotton: leaf biomass, stem biomass and storage (Fig 5). We also added a 'response' compartment and a flow for 'lost' carbon resource (representing carbon that is actually lost from the system as well as that which becomes fixed in structural components other than leaf and stem). Resources flow into the system and between the compartments, altering the state variables corresponding to each compartment.

Various aspects of physiology are simplified or implied. For example, the existence and characteristics of the storage pool are only qualitatively known, and the exact character of the response compartment is unknown. The structure

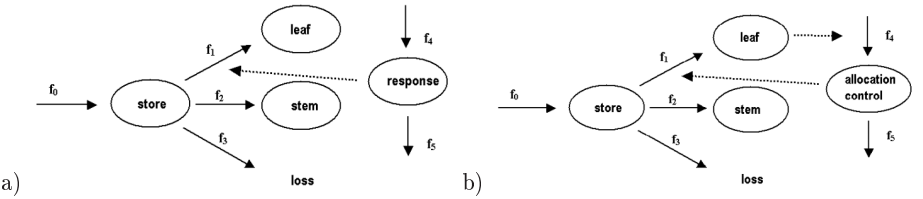


Fig. 5. Flow diagrams for the canonical model. The ellipses represent compartments. The solid arrows represent flows between compartments. The dotted arrow represents the influence that a compartment has over a flow. a) The 'presence' version. b) The 'absence' version

of the canonical physiology model is based on qualitative assumptions about the system.

In the case of our model, the primary aim was to demonstrate the process of compensation, to assist in constructing hypotheses – and therefore further experiments – about compensation, and to simulate the effects of presuming these hypotheses to be true. Clearly, in order for the cotton plant to compensate for damage in a mathematical model, there must be a response - a mechanism for changing the behaviour of the system so that it returns to the same state as an undefoliated system. The basis of our model's response mechanism is the adjustment of allocation patterns away from storage towards leaf and stem growth.

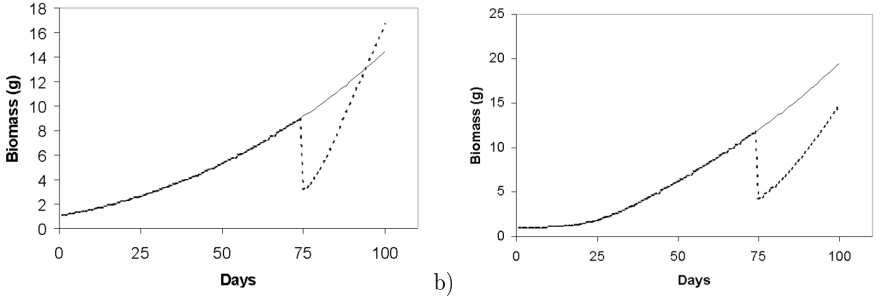
This response was modelled in two different ways. Firstly, in the 'presence' version, a defoliation event stimulates the production of a response, which could be a cascade of compounds in real plants. The response compartment alters the system's allocation strategy while it is present, and decays after its initial stimulation. Secondly, in the 'absence' version, an 'allocation control' compound is produced consistently at a rate proportional to the leaf biomass pool and decays at a constant rate; upon the removal of some portion of the leaf mass (i.e. defoliation), the size of the 'allocation control' pool drops rapidly, causing the response variable to increase in inverse proportion. The change in the response variable alters the allocation pattern, again away from storage towards leaf and stem growth.

The values of the constants for these equations were determined by evolutionary optimisation algorithms; the model was fitted to a subset of the data from the experiment. The model was solved using the Euler method for differential equations.

The model displays near-complete compensation for leaf biomass (Fig 6), which fits with the experimental data.

6 Conclusions

Expressing our hypotheses in a dynamic model has enabled us to investigate their interactions in a concrete manner. The model that assumed increased leaf production rate was tested against an experimental data set and required adjust-



a) b)

Fig. 6. Leaf biomass per plant, predicted by the canonical model. Solid line: undefoliated; dotted line: defoliated 65% at day 75. a) presence version; b) absence version

ment. The new model incorporated obligate development of the main stem, as this appears to be a significant contributor to the ability of these plants to compensate for defoliation in the tested indices such as leaf area and plant height. Uninterrupted development of new metamers on the main stem allows for replacement of lost leaf area without depending significantly on the remaining leaf area after defoliation, therefore the model is able to compensate for the lost leaf tissue. This physiologically explicit model is well suited to testing specific aspects of physiology. However, it is very sensitive to incomplete or incorrect knowledge (whether qualitative information or quantitative data).

The alternative, canonical model is particularly effective at demonstrating compensation for defoliation. Unlike the physiologically explicit model, it simulates compensation using a whole-plant controlling mechanism - the adjustment of resource allocation away from storage and into vegetative parts. Because it is mechanistic at a global level, it can be parameterised to fit a set of observations relatively easily. While not giving insights into interacting physiological hypotheses at a local level, it does allow us to investigate other aspects of the system. While both 'absence' and 'presence' models achieve compensation, only the 'presence' model can simulate over-compensation. If experimental results showed overcompensation, this would lend support to the 'presence' mechanism at the expense of the 'absence' mechanism.

We intend to investigate the issue further by producing a hybrid model: one that uses a canonical model to determine physiological parameters, which would then be passed to an architectural model for allocation to structure. The level of interaction between the two models may be flexible; would probably be the major part of the problem of getting the two together. It is our hope that the structural model will provide a method for improving the canonical model's realism, while the canonical model should help avoid the problems of inconsistent or missing data we have observed in constructing the structural model with explicit physiology.

References

1. Prusinkiewicz, P.: A look at the visual modelling of plants using L-systems. Agronomie: agriculture and environment 19, 211–224 (1999)
2. Hanan, J.S. and Room, P.M.: Practical aspects of virtual plant research, In Michaelewicz, M.T. (ed), Plants to ecosystems. Advances in computational life sciences. CSIRO, Melbourne, 28–44 (1997)
3. Sadras, V.O. and Fitt, G.P.: Resistance to insect herbivory of cotton lines: quantification of recovery capacity after damage. Field crops research 52, 127–134 (1997)
4. Mascarenhas, V.J., Cook, D., Leonard, B.R., Burris, E. and Graves, J.B.: Late season beet armyworm (Lepitoptera: Noctuidae) infestations on cotton: defoliation, fruit damage, and yield loss. Florida entomologist, 82(2), 218–229 (1999)
5. Longer, D.E. and Oosterhuis, D.M.: Cotton regrowth and recovery from early season leaf loss. Environmental and experimental botany, 41, 67–73 (1999)
6. Longer, D.E., Oosterhuis, D.M., and Withrow, M.: Cotton seedling recovery from partial and complete defoliation. Arkansas Farm Research, 42, 10–11 (1993)
7. Hanan, J.S.: Virtual plants – integrating architectural and physiological models. Environmental modelling and software, 12(1), 35-42 (1997)
8. Ashley, D.A.: ^{14}C-labelled photosynthate translocation and utilization in cotton plants. Crop science 12, 69–74 (1972)
9. Meyer, G.A.: Pattern of defoliation and its effect on photosynthesis and growth of goldenrod. Functional ecology 12, 270–279 (1998)
10. Verkaar, H.J.: Are defoliators beneficial for their host plants in terrestrial ecosystems? – a review. Acta Botanica Neerlands 37(2), 137–152 (1988)
11. Ashley, D.A., Doss, B.D. and Bennett, O.L.: Relation of cotton leaf area to plant growth and fruiting. Agronomy journal, 61–64 (1964)
12. Constable, G.A.: Section 7: Distribution of ^{14}C photosynthate in cotton, in Carbon fixation and distribution in cotton: implications of single-leaf measurements to plant performance, Ph.D. Dissertation, 82–122 (1981)
13. Constable, G.A. and Rawson, H.M.: Effect of leaf position, expansion and age on photosynthesis, transpiration and water use efficiency of cotton. Australian journal of plant physiology, vol 7, 89–100 (1980)
14. Renton, M., Hanan, J. and Kaitaniemi, P.: The inside story: including physiology in structural plant models. Proceedings of the international conference on computer graphics and interactive techniques in Australiasia and South-East Asia (GRAPHITE2003), 95–102 (2003)
15. Kaitaniemi, P.:A canonical model of tree resource allocation after defoliation and bud consumption. Ecological modelling 129, 259–272 (2000)
16. Martin, P.G.:The use of canonical S-system modelling for condensation of complex dynamic models. Ecological modelling 103, 43–70 (1997)
17. Torres, N.V.: S-system modelling approach to ecosystem: application to a study of magnesium flow in a tropical forest. Ecological modelling 89, 109–120 (1996)
18. Voit, E.O. and Sands, P.J.: Modeling forest growth II – biomass partitioning in Scots pine. Ecological modelling 86, 73–89 (1996)

3D Visualization of Complex Surfaces Using Dynamic Delaunay Tessellation

M.L. Gavrilova and J. Pivovarov

Dept. of Computer Science, University of Calgary
Calgary, AB, Canada T2N1N4
{marina,juraj}@cpsc.ucalgary.ca

Abstract. The need for robust and reliable rendering mechanism is very high in the areas of computer simulation, biological modeling and computer graphics. Specific applications, such as modeling of growing forms, require tailored approaches to visualization. The paper describes the software developed for maintaining dynamic changes and visualization of complex 3D surfaces, utilizing the Delaunay tessellation as an underlying data structure. Some qualitative analysis of visualization is presented and the algorithm for skeletonization based on the developed model is discussed.

1 Introduction

The need for robust and reliable rendering mechanism in the areas of computer simulation, biological modeling and computer graphics is hard to underestimate. Due to particular specifics of each of the applications and expected outcomes requiring visualization, numerous techniques have been developed so far. For instance, a number of data structures such as TIN (Triangulated Irregular Networks), ROAM (Real-Time Optimally Adapting Meshes), quadtree-based meshes were developed for terrain visualization in GIS (Geographical Information Systems) [2,3,5,7]. Smoothing, culling and error metrics algorithms are applied to improve visual features of the model and achieve a more realistic representation [7,11].

In the area of image processing, the algorithms for investigating connectivity problems and background skeletonization have been intensively investigated. They rely on a number of techniques, ranging from peeling to building medial axis using Voronoi diagram based methods [1]. These methods often serve as basic algorithms for research on template matching and morphing, which in turn require efficient visualization techniques for algorithm verification.

In solid modeling, a number of problems that typically addressed are: how to represent a model in the computer, how to construct or reconstruct a surface from discrete point data, and how to extract features of a model. None of these are possible to address without appropriate visualization techniques [8,9,13].

In the following, we describe the specifics of the software developed for 3D dynamic Delaunay tessellation visualization and discuss the variety of problems that this software will assist in solving.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 718–726, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Relevant Research

Connections have been growing recently between the modeling community and the computational geometry. Many questions in molecular modeling can be understood geometrically in terms of arrangements of spheres in two and three dimensions. Problems include computing properties of such arrangements such as their volume and topology, testing intersections and collisions between molecules (particles), finding offset surfaces (related to questions of accessibility of molecule subregions to solvents such as water), data structures for computing interatomic forces and performing molecular dynamics simulations, and computer graphics algorithms for rendering models of growing forms accurately and efficiently (taking advantage of their special geometric structure). One of the very relevant areas is morphogenesis. Methods for the analysis of and comparison of growth forms are very important in this area, while the 3D methods seem to still remain an underexplored field [16].

One of most commonly used data structures, possessing a number of important properties from both geometrical and application-oriented perspectives and can be applied to the above problem, is the generalized Voronoi diagram and its dual, Delaunay triangulation (tessellation). Shortly, the structure represents the proximity information of a set of entities (points, objects) in a general d-dimensional space. The subdivision of a space by Voronoi regions provides a convenient background for point location, nearest-neighbor queries and studying of the specific physical properties of the modeled systems, such as distribution of voids in molecular biology, grow patters in growing systems modeling and properties of clusters in granular materials simulation [12,13]. The corresponding triangulation (tessellation) possesses properties of being regular, satisfying min-max criterion as well as a number of other local optimization criteria, which makes it a desired choice for surface representation, meshing and model visualization.

For instance, a solvent-accessible triangulated surface generator for molecular graphics and boundary element applications was developed by Zauhar [16]. A 3D Delaunay tessellation was built to study and visualize the structure of fluids by Medvedev [13]. The algorithm for the rapid display of three-dimensional Delaunay meshes on standard raster displays, without the use of special purpose graphics hardware, was devised by Karasick [10]. Method for visualization of 3D surfaces and curves, based on Delaunay property of the underlying tessellation, was devised by [Baker]. In this paper, we describe the first prototype implementation of the software that allows to build and dynamically maintain the 3D model of the growing entity (such as coral, for instance), compute the medial axis of the model and perform fast visualization.

3 Preliminaries

Consider a set of objects S in R^d.

Definition 1. A *generalized Voronoi diagram (GVD)* for a set of objects S in R^d is the collection of generalized Voronoi regions

$$GVor(P) = \{\mathbf{x} \mid d(\mathbf{x}, P) \leq d(\mathbf{x}, Q), \forall Q \in S - \{P\}\},$$

where $d(\mathbf{x}, P)$ is the distance function in R^d between point $\mathbf{x} \in R^d$ and object $P \in S$.

Thus, a generalized Voronoi region of a generalized Voronoi diagram of *n* sites is obtained as the intersection of $n-1$ half-spaces. The boundary of the generalized Voronoi region consists of up to $n-1$ facets, where each facet is a $d-1$ dimensional face. A *1*-dimensional face is called a generalized Voronoi edge, and a *0*-dimensional face is called a generalized Voronoi vertex.

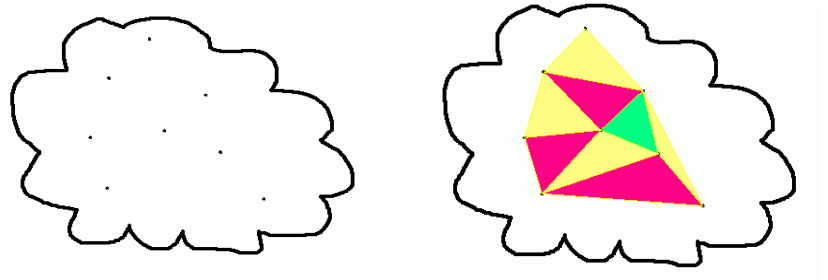
Definition 2. A *generalized Delaunay triangulation* is the dual of the generalized Voronoi diagram obtained by joining all pairs of sites whose generalized Voronoi regions share a common generalized Voronoi edge.

Definition 3. A *generalized Delaunay tessellation (DT)* corresponding to the generalized Voronoi diagram is a collection of *d*-dimensional simplexes such that for each generalized Voronoi vertex

$$\mathbf{v} = Vor(\mathbf{p}_1) \cap Vor(\mathbf{p}_2) \cap ... \cap Vor(\mathbf{p}_{d+1})$$

there is a simplex $(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{d+1})$ in the generalized Delaunay tessellation.

Fig. 1 illustrates the above concepts in 2D space.



Original point set 2D Delaunay Triangulation

Fig. 1. Point set (left) and corresponding Delaunay Triangulation (right).

In 3D, the Delaunay tessellation contains edges (1-dimensional) and facets (2-dimensional triangles). The Voronoi dual of a facet of DT is a segment (a Voronoi edge). For each edge of DT connecting two sites, the dual is a Voronoi facet, which is a part of the bisecting plane in 3D separating the two sites.

4 Software for Dynamic 3D Delaunay Tessellation

VisualizeDT is a program that visualizes the 3D Delaunay tessellation. The latest version of the program was compiled with Microsoft Visual C++, although it also runs under UNIX environment. The 3D Delaunay tessellation is build by utilizing the incremental construction algorithm, first presented in [6]. The DETRI software is used for building the initial 3D Delaunay tessellation [4,14]. The state of the art

program is written to enable dynamic changes in the Delaunay tessellation, computing statistics and performing 3D visualization was created (Fig. 2).

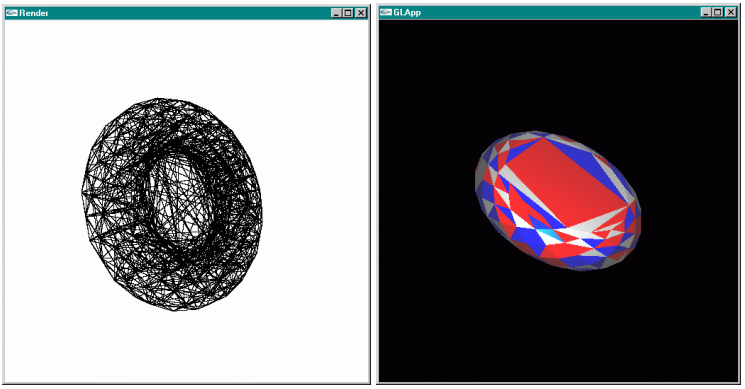


Fig. 2. Edges of DT of a torus (left), surface of DT (right).

The original DETRI program was made to run in two modes, one of them is for computing the tessellation from scratch, and the other mode is for reading the cached tessellation from a binary file, and then doing some reporting with it. The program creates an initial state; one that is highly dependent upon static sizes of the inputs. To alleviate this assumption, we made the software more "object-oriented". The algorithm state was abstracted and stored in a C structure. Some of the functions were changed to work with this state, rather than a set of global variables. This has the advantage too that more than one model could be manipulated. Finally, the state can be left around as long as is necessary while someone "interacts" with the model by adding points etc.

In the original DETRI code, the initialization happened, then the 3D Tessellation was computed, then the clean up occurred. Thus it was difficult to run just the "creation" portion of the software in order to be left with a model to work with. In the new organization of an abstracted algorithm state, the algorithm can be paused before cleanup takes place, again to facilitate interaction with the model.

4.1 Description of Data Formats

Using the modified code for Delaunay tessellation involves running the program twice. The first time it is run, the 3D Delaunay tessellation is computed and saved in a binary file format. By convention, the set of points are stored in an ascii file named without an extension. The binary file produced is saved in a file named as the point set file, but with a `dt` extension. Ex: `file` --> `file.dt`. The second time the program is run, it reads the binary file containing the tessellation and creates an ascii file outputting one or more of the tetrahedra, triangles, edges and vertices. More specifically, the following four types of files are used:

file

This ASCII file contains a sequence of 3D points "x y z" on each line.

file.DT.fl

This ASCII file has a different format based on which options are specified in the `-C "print ..."` command-line option. Basically, it contains a mish-mash of tetrahedrons, triangles, edges and vertices.

- A tetrahedron is displayed as "T a b c d" where *a,b,c,d* are 1-based indexes into the list of points.
- A triangle is displayed as "f a b c" where *a,b,c* are 1-based indices into the list of points.
- An edge is displayed as "e a b" where *a,b* are 1-based indices into the list of points.
- Finally a vertex is "v a" where *a* is a 1-based index into the list of points. All the above objects are part of the 3D Delaunay Tessellation of the 3D Point Set.

file.info

This ASCII file has some stats from the DT computation. It contains statistics of how many tetrahedrons, triangles, edge and faces were found, time requires per each operation, a number of comparisons performed.

file.voronoi

This ASCII file contains the Voronoi edges of the dual Voronoi graph. The format is one line segment per line "x1 y1 z1 x2 y2 z2" where one segment exists for each pair of tetrahedrons that share a face. A circumscribing sphere is computed for each tetrahedron T1, T2 with centers $c_1 = (x_1,y_1,z_1)$ and $c_2 = (x_2,y_2,z_2)$.

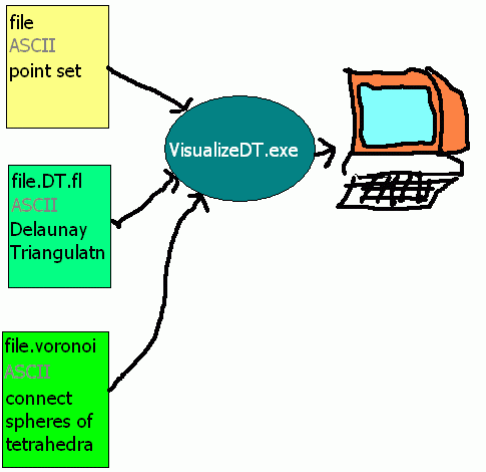


Fig. 3. Running the VisualizeDT program.

4.2 VisualizeDT Program

The VisualizeDT program uses information from the files **file**, **file.DT.fl** and **file.Voronoi** in order to create visual representation of Voronoi diagram and Delaunay tessellation (see Fig. 3).

The VisualizeDT program has 4 modes of viewing (see Fig 4):

– the set of points, highlighted (mode 1),
– the edges in the tessellation (mode 2),
– the triangles/faces of the tessellation (mode 3) and
– the dual Voronoi edges of the tessellation (mode 4).

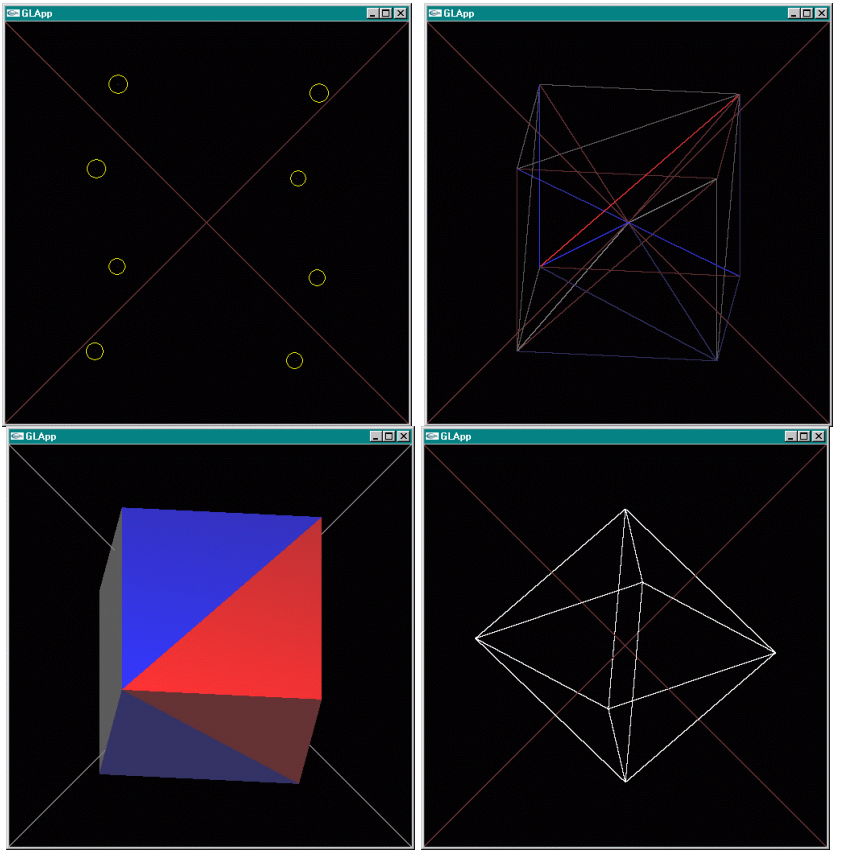


Fig. 4. Four modes of viewing in VisualizeDT program.

5 Dynamic Update of Delaunay Tessellation

The algorithm has been modified so that it can add extra points dynamically to the point set (see Fig. 5). The detailed algorithm description follows.

Algorithm for dynamic modification of the Delaunay tessellation and Voronoi diagram

1. *(Preprocessing)* Construct the Delaunay tessellation (using DETRI code).
2. *(Main loop)* Insert a new point in the Delaunay tessellation.

   2.1 Perform a point location to determine the tetrahedron *T* for a new point to be inserted. The inserted point splits *T* onto four new tetrahedra. This results in 8 new quadrilaterals (four based on faces and four on edges).

   2.2 Compute the inscribed sphere for the eight new quadrilaterals, using the function *Inscribed_sphere*.

   2.3 If the inscribed sphere is not empty, i.e. contains Voronoi sites of the original set *S, GOTO 2.4*
   Else GOTO 2.5

   2.4 Perform the *swap* operation. In 3D, the operation involves swapping the facet for an edge or an edge for a facet. For each of the newly introduced quadrilaterals (based on new facets and new edges), perform *Inscribed_sphere* test. GOTO 2.3
3. Update the corresponding Voronoi diagram. Compute dual Voronoi edges for all new DT facets and edges, using coordinates of the obtained inscribed spheres to determine the Voronoi vertices.

The complexity of the algorithm in the worst-case is $O(n^3)$. But on average, only a constant number of swaps is required at each step, i.e. $O(1)$ per insertion.

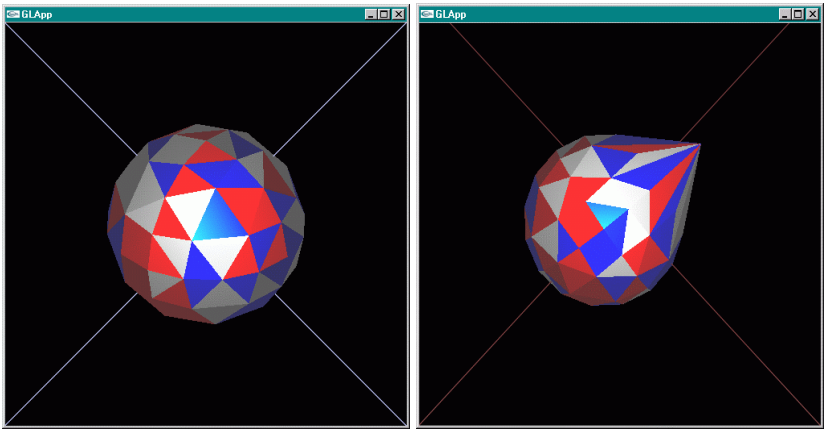


Fig. 5. Spherical surface (left), extra point dynamically added (right).

6 Algorithm for Building a 2D Medial Axis Based on a Voronoi Diagram

An algorithm to build the 2-dimensional medial axis for any randomly shaped object in the plane was developed and implemented. The algorithm also allows visualizing the Delaunay triangulation, Voronoi diagram and medial axis in 2D. The algorithm outline is as follows:

1. Compute the coordinates of a Voronoi edge for a given DT edge: find two adjacent triangles, find two corresponding Voronoi vertices.

2. Assign a weight to each DT edge XY using either approach *a* or *b*:

   Approach *a:* use the edge length.

   Approach *b:* compute the shortest path between X and Y along the edges of the input curve.

3. Start removing DT edges (and corresponding Voronoi edges) with smaller weights first. Use visualization to evaluate the termination condition.

The program then displays the Voronoi diagram, Delaunay triangulation and the medial axis. The medial axis shape can be drawn interactively or read from a file. In Fig. 6, the resulting medial axis is identified by blue font, the Delaunay triangulation edges by yellow font. The implementation of a 3D version of the algorithm is currently on its way.

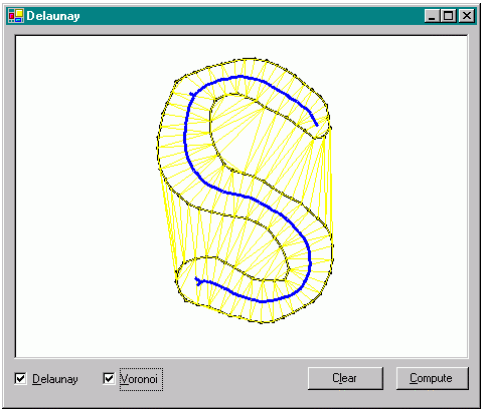


Fig. 6. 2D Medial axis and Delaunay triangulation.

7 Conclusions

The efficient visualization of growing surfaces is an area, highly relevant to morphogenesis. The presented methods for visualization of 3D surfaces in the presence of dynamic updates are the first step on the way to a better understanding of complex biological structures, and will provide an efficient framework for testing new developments in that area.

References

[1] Baker, T.J. (1989) Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation, Engng. Comput., 5 p. 161.
[2] Blow, J (2000) Terrain Rendering at High Levels of Detail, in Proceedings of the Game Developers' Conference, San Jose, California.
[3] De Berg, M. and Dobrint, K. (1995) On levels of detail in terrains, Proc. 11th ACM Symposium on Computational Geometry June 1995.
[4] DETRI code www. Geom..umn.edu/software/cglist/GeomDir/#DETRI
[5] Duchaineauy, M, Wollinshy, M. (1997) ROAMing Terrain: Real-Time Optimally Adapting Meshes, IEEE Visualization '97 Proceeding.
[6] Edelsbrunner, H. and Shah, N. (1996) Incremental topological flipping works for regular triangulations, Algorithmica, vol. 15, pp. 223–241.
[7] Hesse, M and Gavrilova, M. (2003) Quantitative Analysis of Culling Techniques for Real-time Rendering of Digital Elevation Models, WSCG.
[8] Kaandorp, J.A. and Sloot, P.M.A. (1997) Parallel simulation of accretive growth in 3-dimensions. BioSystems 44(3):181–192.
[9] Kaandorp, J.A. and Sloot, P.M.A. (2000) Growth and form of sponges and corals in a moving fluid. In Pattern formation in biology, vision and dynamics, pp. 24–41, World Scientific.
[10] Karasick, D. Lieber, L. R. Nackman, and V. T. Rajan (1997). Visualization of Three-Dimensional Delaunay Meshes. Algorithmica, 19(1-2):114–128.
[11] Lindstrom, P and Pascucci, V. (2001) Visualization of Large Terrains Made Easy, IEEE Visualization 2001, California, October 21–26, 2001
[12] Luchnikov, V.A, Gavrilova, M.L., Medvedev, N.N., Voloshin V.P. (2002) The Voronoi-Delaunay Approach for the Free Volume Analysis of a Packing of Balls in a Cylindrical Container, Future Generation Computer Systems, 18, pp. 673–679, Elsevier.
[13] Medvedev, N.N. (2000) Voronoi-Delaunay method for non-crystalline structures, SB Russian Academy of Science, Novosibirsk.
[14] Mücke, E., Saias, I. and Binahi Zhu (1996) Fast Randomized Point Location Without Preprocessing in Two- and Three-dimensional Delaunay Triangulations. 12th Annual Symp. on Comp. Geometry.
[15] Prusinkiewicz, P (1994) Visual Models of Morphogenesis. In Artificial Life, 1(1/2):67–74.
[16] Zauhar, R.J. (1995) SMART: A solvent-accessible triangulated surface generator for molecular graphics and boundary element applications, J. Computer-Aided Molecular Design 9, 149–159.

CODELAB: A Developers' Tool for Efficient Code Generation and Optimization

Dragan Mirković[1] and S. Lennart Johnsson[1]

Department of Computer Science
University of Houston
Houston, TX 77204
mirkovic@cs.uh.edu, johnsson@cs.uh.edu

Abstract. In this paper we describe CODELAB, an integrated development environment (IDE) for efficient code development and optimization. The main idea behind the CODELAB IDE is to bring together several useful code development and optimization tools into an integrated and user friendly environment with a goal of simplifying the production and maintenance of high performance software libraries. We give an overview of the tool structure and provide examples that illustrate the use and the efficiency of the IDE.

1 Introduction

Over the last 30 years, microprocessors have enjoyed a steady increase in performance inherent in Moore's law. The complexity of microprocessor architectures, however, has also been steadily increasing and the efficient application of scientific computing techniques requires constantly growing amounts of specialized knowledge in a variety of diverse scientific areas such as numerical analysis, algorithm theory, computer architectures, and programming languages. These requirements have put the high performance code development out of reach for many researches, since in most cases, simple implementations achieve only a small fraction of the peak performance on any given advanced architecture.

Most of the scientific software today is written in high-level programming languages and relies on efficient programming methods, good optimizing compilers, and use of hardware–tuned software libraries to achieve high performance. A number of other tools have been developed either to simplify the code writing or to help the optimization. The most popular are scripting (or Very High Level) languages (VHLLs) like MATLAB, Python, or Tcl/Tk. The primary advantage of these languages over a third generation language like C or C++ is that fewer lines of code must be written to complete a given task. They provide the user with a clear and simple expression of complex computational algorithms and allow for easy construction of graphical user interfaces. A further increase in productivity arises from the fact that scripting languages are usually interpreted which removes the compile-link-run cycle delays. The performance of applications developed with VHLLs, however, strongly depends on the granularity of

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 729–738, 2003.
© Springer-Verlag Berlin Heidelberg 2003

the code and on the quality of numerical and graphical libraries hidden behind the interpreter function calls. In general, long loops with a large number of simple numerical or graphical instructions perform poorly due to high interpreter overhead. To improve the performance in these situations most VHLLs need to be extended by adding procedures written in some lower level language. In many cases use of tightly-integrated tools can cut down the development time. Current trend is to join together compilers, libraries and supporting tools into a single, integrated development environment (IDE), allowing the user creating, compiling, linking, assembling and debugging within a single application.

Even with all these tools one problem still remains. Someone has to update the libraries and optimize them for the hardware that changes very rapidly. The advanced optimization techniques like loop overhead reductions through unrolling, cache optimizations, instruction level parallelism and use of special vector instructions produce code that is very difficult to maintain. The problem is further exacerbated if the library is to be portable since most of these optimizations are platform specific. The expensive and time consuming nature of these optimizations makes them nearly impossible to do by hand and new approaches are needed to address this problem. The current trend to achieve portable performance for scientific software libraries is to design software that can automatically adapt to different architectures either during the installation, or at runtime, or both. Examples of this approach include linear algebra libraries like ATLAS [1] or PHiPAC [2], fast Fourier transform libraries like FFTW [3], UHFFT [8, 9, 10] or Signal Processing libraries like SPIRAL [13]. These packages usually provide many ways of doing required operations, and use empirical timings or performance estimates in order to choose the best method for a given architecture. The implementation of different ways of performing the same operation can, in some cases be parametrized and the same code can be used on different architectures with parameters selected either at installation/compile time or at run time. More often the only way is to have different source code implementations for the same operation and select the one that performs the best on the given underlaying architecture. These multiple sources can either be hand–written and tuned or automatically generated, where the automatic code generation has a significant advantage from the developers point of view.

There are very few general tools for library code development and most of them are developed by the architecture vendors and are useful only for a restricted set of architectures. These include performance measuring tools and simulators that can efficiently track the flow of instructions and data through the maze of memory hierarchies and instruction pipelines of modern microprocessors and code generators that can produce tens of thousands of lines of computer code in seconds, visualize the code structure and automatically perform sophisticated optimizations. The low level performance measuring tools are hardware–specific and there are very few portable solutions, like PAPI [11] or PCL [12].

In this paper we describe an IDE that joins together several code development tools into a single application with the goal to simplify development and tuning of modern adaptive software libraries.

2 Overview of the CODELAB Structure

The purpose of the CODELAB project is to create a general code generation/performance measurement tool that would enable advanced code developers to easily generate, modify and test the performance of large amounts of code. The CODELAB project is a generalization of our effort in development of an adaptive FFT library [8, 9, 10] and as such it has inherited many of the tools developed in the UHFFT project. In UHFFT we have developed an efficient automatic method of generating and optimizing the FFT kernels. The code generator used in the UHFFT is based on the abstraction of the numerical algorithm and most of the manipulations and optimizations are done using a general abstract expression algebra. This approach can can be easily applied to other similar, numerical algorithms.

The schematic structure of the CODELAB IDE is shown in Figure 1.

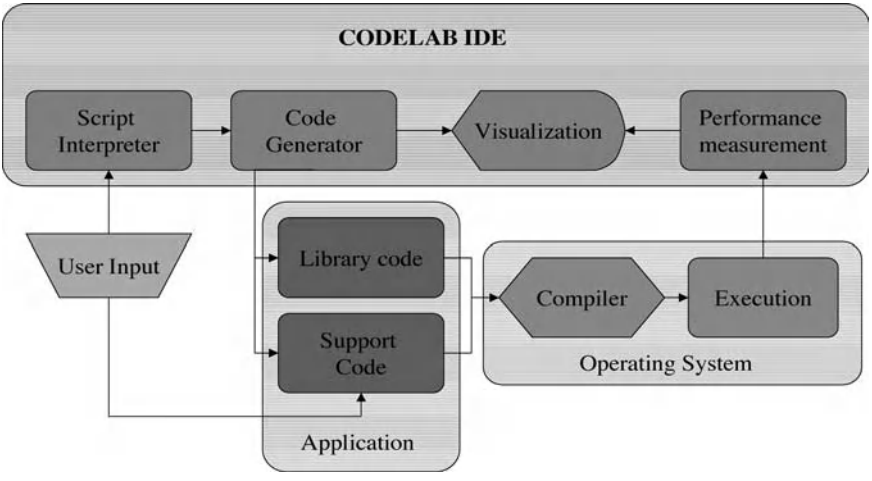


Fig. 1. High–level structural overview of the CODELAB integrated development environment.

2.1 Applications Suitable for the CODELAB Environment

The basic assumption in CODELAB is that the application consists of two principal parts. The first part is *the library code* which contains a large number of highly optimized computational kernels needed in the application. We assume that the kernel algorithm can be parametrized by a small set of parameters like kernel size, dimension, precision, order, blocking factor, factorization options,

etc. By varying these parameters the same code generator code can produce a wide variety of computational modules.

The second part is *the supporting code* that binds the library routines together such that the function required by an application can be computed. This part can be either hand-written or automatically generated. The supporting code also contains an appropriate search method which automates the selection of the most optimal available combination of computational modules from the associated library.

There are many important applications that satisfy the requirement of the CODELAB environment and we list here some of them

- Various numerical linear algebra routines. Most of the linear algebra algorithms can be blocked in many different ways and the blocking factor has a strong influence on cache utilization and on the performance of the implementation.
- Discrete Fourier and other linear transforms that can be factored or decomposed in order to obtain a fast algorithm by using a divide–and–conquer approach. These factorizations are not unique, and there are, in general, many different ways to compute the same transform. Here different factorizations not only change the blocking of the algorithm, but may also have a different number of floating point operations.
- Finite element methods (FEM). We are developing a general Lagrangian finite element code generator that can produce a finite element bases of arbitrary order and dimension and a variety of associated element and macroelement stiffness matrices. For a given FEM discretization the order of the basis and the size of the macroelement (i.e., a connected collection of elements) define the blocking factor and the cache utilization.
- Multiprecision calculations. A code generation can be efficiently used to implement a high precision mathematical library with a fixed length of the mantissa. The length of the mantissa can be implemented as a parameter in the code generator and a whole library implementing a variety of mathematical functions can be generated in a short amount of time.
- Various signal and image processing kernels can be implemented as a combination of fast Fourier transform and linear algebra algorithms.

2.2 The Scripting Language

The first step in library code development is efficient and simple algorithm descriptions. For this purpose we use a simple scripting language with constructs for standard algorithm elements, like variables, constants, expressions, repetitions and branching, real and complex arithmetic, matrix and vector algebra, trees, graphs, and lists.

The main purpose of the scripting language is to simplify construction of different code generators. In general, a code generator for a given algorithm can be written directly in C, and the script part can be bypassed. Scripts, however, provide for a more compact description of different algorithms.

The user provides a script for the code generator and some of the supporting code for the application. The code generator produces a library of computational kernels and a portion of the support code. The computational kernels contain, in most of the cases, optimized straight line code.

2.3 The Code Generator

The script which describes the algorithm is either interpreted or translated into the code generator which can produce a large variety of computational kernels depending on the input parameters. The first step in code generation is the *algorithm abstraction* in which a portion of the application algorithm is transformed into a list of expression structures. Hence, the basic structure in the code generator is called **Expression**. An expression is a polymorphic structure that takes one of the following forms:

- **Constant Expression**: A constant can be a real constant (**REALCONST**), an integer constant (**INTEGERCONST**), a Zero constant(**ZEROCONST**) or a constant with a value equal to one (**ONECONST**). All constants are stored as doubles and additional information are used in order to facilitate simplifications of expressions.
- **Variable Expression**: This is an abstraction for the variables used in different algorithms. The interpretation of the variable depends on its type, which is a composition of two attributes:
  - o relevance to the algorithm, i.e., input, output or temporary.
  - o type of data stored, i.e., integer, real, imaginary or complex.
- **Sum Expression**: This is an abstraction for composite expressions of the form $\sum_{i=0}^{N} e_i$, where e_i are expressions.
- **Product Expression**: This is an abstraction for composite expressions of the form $\prod_{i=0}^{N} e_i$, where e_i are expressions.
- **Negative Expression**: This is an abstraction for the change of sign. It is also used for subtraction, e.g, the statement *SumExpr(expr1, NegExpr(Expr2))* amounts to subtracting *expr2* from *expr1*.
- **Assign Expression**: This is an abstraction for expressions of the form $v = e$ where v is a variable and e is an expression.
- **Complex Expression**: This is the abstraction for complex arithmetic. it consists of two expressions, one for the real part and one for the imaginary part.

The four basic operations on Expressions are the sum (**SumExpr**), the product (**MultExpr**), the quotient (**DivExpr**), and the sign change (**NegExpr**). These four operations are used in the generation of Expression Lists. The derived expressions may contain some redundancies which are removed to reduce the total number of arithmetic operations in the expression list. Some of the simplifications include:

- Cancellation of terms in sums, e.g, $(tmp3 + tmp5) + (tmp6 - tmp3)$ would yield $tmp5 + tmp6$.

- Trivial multiplications are eliminated, i.e., multiplications by ± 1, $\pm i$ and 0.
- Common constants are factored out, e.g., $C2 * tmp5 + C2 * tmp7$ would yield $C2 * (tmp5 + tmp7)$.
- Constants are folded, e.g., $C2 * C3 * C4 * tmp5$ would yield $C5 * tmp5$, where $C5 = C2 * C3 * C4$.
- Multiple sign changes are simplified, e.g., $-(-tmp2)$ would yield $tmp2$.
- Complete complex algebra is included, i.e., the generator is able to deal with complex expressions.

From the basic structures described above, other more complex structures are derived. The Expression Vector (**ExprVec**) is a vector of expressions. Each expression may be of one of the defined types for an expression. The Expression Matrix (**ExprMat**) is a two dimensional structure of expressions. There are a number of operations that are used to manipulate these structures. These operations obey the rules of vector addition and multiplication, e.g., to add two vectors together, they must be of the same size.

The code generator can also provide information about the number of arithmetic operations in the generated code, efficiency of simplifications that are performed, number of registers or temporary variables needed for execution. The abstraction of the kernel algorithm results in a list of expressions and the structure of the generated code can be visualized either by printing the list of expressions or by plotting the direct acyclic graph (DAG) of the algorithm, see Figure 2. The application code is subsequently compiled and executed and the performance measurements can be collected and visualized. There are several advantages of the automatic code generation in library code development. The code that is optimized for modern architectures is, in general, difficult to write since it may contain large portions of completely unrolled loops (straight line code). If these are generated automatically the developer is relieved of a tedious and error prone task of writing them by hand. Moreover, the code generation introduces an additional level of abstraction and allows for separation of the algorithm description from the algorithm implementation. This means that the same algorithm once described in the code generator scripting language can have a number of different instances of implementation, depending on the various options selected during the optimization and scheduling phase.

Example: FFT code generator As an example, we describe the structure of the code generator for fast Fourier transforms used for fast evaluation of the Discrete Fourier Transform (DFT), which is a matrix–vector product of the form $\mathbf{y} = W_n \mathbf{x}$, where $W_n = (w_{lk})_{l,k=0}^{n-1}$ is the DFT matrix with elements $w_{lk} = \omega_n^{lk}$, and $\omega_n = e^{-2\pi i/n}$ is the principal nth root of unity. The fast evaluation is obtained through factorization of W_n into the product of $O(\log n)$ factors

$$W_n \mathbf{x} = (A_1 A_2 \dots A_r)\mathbf{x}, \tag{1}$$

where the matrices A_i are sparse and $A_i \mathbf{x}$ involves $O(n)$ arithmetic operations. The factorization (1) for a given n is not unique, and possible variations may

```
tmp0i = Im(x[0])+Im(x[is]);
tmp0r = Re(x[0])+Re(x[is]);
tmp1i = Im(x[0])-Im(x[is]);
tmp1r = Re(x[0])-Re(x[is]);
Re(y[0]) = tmp0r;
Im(y[0]) = tmp0i;
Re(y[os]) = tmp1r;
Im(y[os]) = tmp1i;
```

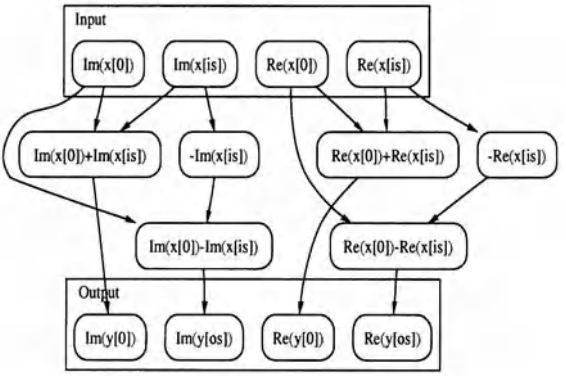


Fig. 2. The list of expressions for the size 2 complex-to-complex fft and the corresponding DAG.

have properties that are substantially different. The code generator used for building the library of FFT kernels in our example uses mixed–radix, prime factor, split–radix, and Rader's algorithms. (see [5, 6, 10] for a more detailed description of these algorithms). The FFT code generator is implemented using a fixed set of factorization rules in a recursive fashion. An additional layer of software has been added to the code generator library to facilitate building of expression lists related to the fast Fourier transforms. They include:

- **GenDFTExprMat**: this function returns the DFT expression matrix that consists of complex constant expressions. It takes in the DFT size and the transform direction as its input arguments.
- **GenFFTExprList**: this function generates the expression list for the FFT of size n. It takes in as arguments the input vector (to be transformed) and the direction of the required transform and then calls the appropriate FFT algorithm, depending on the size of the input vector. Pseudo code for this functions is as follows:

GenFFTExprList(ExprVec *u)
⋮
$n =$ Length of vector **u**
if $n = 2$ apply DFT to the vector **u**
else if n is prime call RADER'S algorithm
else {
choose factor r of n
if r **and** n/r are co–prime call PRIME FACTOR algorithm
else if $n > 8$ **and** n mod $4 = 0$ call SPLIT–RADIX algorithm
else call MIXED–RADIX algorithm
}

- **IxFFT**(r, m): this function evaluates the tensor (kroneker) product of the identity matrix of size r with the FFTExprList of size m. This operation is also known as the parallel FFT.
- **FFTxI**(r, m): this function evaluates the tensor (kroneker) product of the FFTExprList of size r with the identity matrix of size m. This operation is also known as the vector FFT.
- **Mixed–Radix Algorithm**: This returns a list of expressions which is an abstraction for the Mixed–Radix FFT algorithm.
- **Split–Radix Algorithm**: This returns a list of expressions which is an abstraction for the Split–Radix FFT algorithm.
- **Prime Factor Algorithm**: This returns a list of expressions which is an abstraction for the Prime Factor FFT algorithm.
- **Rader's Algorithm**: This returns a list of expressions which is an abstraction for Rader's FFT algorithm.

2.4 Performance Measurements and Tools

The CODELAB IDE has an integrated infrastructure for detailed performance measurements. These measurements provide a valuable feedback about critical points of the generated code and point to ways of improving performance. All of the generated code can be automatically instrumented for simple or advanced performance measurements using hardware performance counters. A simple visualization of the performance is included in the IDE. The low level performance measurements are implemented as an interface to the PAPI library [11].

Some of the specific uses of the performance data in the IDE include:

- Parameter specification for the code generator. One of the assumptions in code generation is that the kernel algorithm can be parametrized by a small set of parameters. Some of these parameters, like blocking factors or factorization options, are performance related and performance information is needed in the selection process.
- Building of performance models for CODELAB applications. Many adaptive applications make use of either direct performance measurements or need a performance model as a part of the application. Easy access to the performance data is essential in building a performance model.
- Interactive code optimization. Some of the code optimizations have to be done manually and this is usually a very tedious and error prone task. Interactive access to the performance data provided in the CODLAB IDE can alleviate some of the difficulties in this process.

As an example, in Figure 3 we show performance maps for some of the codelets from the FFT library. These are given with respect to input and output strides. The collection of codelets constitutes the core of the FFT library and most of the transform CPU time is spent executing the codelets that constitute the transform plan. Each codelet, smaller in size than the whole transform, acts on a subset of data defined by the input stride. The output vector is also,

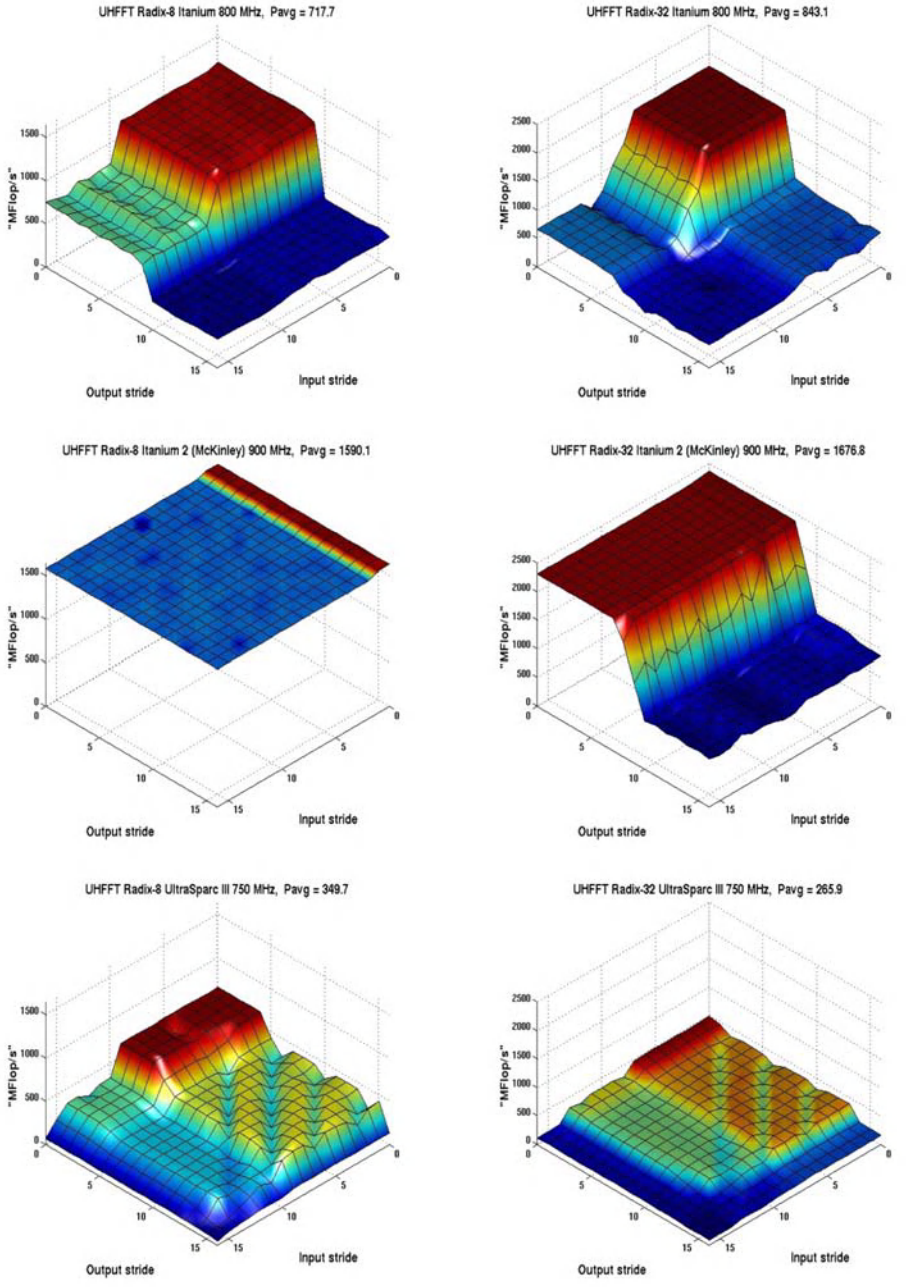


Fig. 3. Performance maps for codelets with sizes $n = 8$ and $n = 32$ on the 800 MHz Intel Itanium, 900 MHz Intel Itanium 2, and 750 MHz Sun UltraSparc-III.

in general, accessed in a strided fashion. For that reason, strided performance of codelets is key to understanding the overall performance of the library. The UHFFT library uses a performance model for the codelets to speed up the selection of the optimal plan for the specific transform and this performance model is based on the data shown in Figure 3.

Acknowledgment.
This work has been carried out with partial support from the DOE funded Los Alamos Computer Science Institute, LACSI, the GrADS projected funded under the Next Generation Software initiative with the National Science Foundation, and in part by the Alliance funded as part of the NSF PACI program.

References

[1] Whaley, R., Petitet, A., Dongarra, J. Automated Empirical Optimization of Software and the ATLAS Project. *Parallel Computing*, 27:3–25, 2001.
2] J. Bilmes and K. Asanović and J. Demmel and D. Lam and C.W. Chin. The PHiPAC WWW Home Page `http://www.icsi.berkeley.edu/~bilmes/phipac`
[3] Matteo Frigo and Steven G. Johnson. The Fastest Fourier Transform in the West. Technical Report MIT-LCS-TR-728, MIT, 1997.
[4] Charles Van Loan. *Computational frameworks for the fast Fourier transform.* Philadelphia:SIAM, 1992.
[5] Richard Tolimieri, Myoung An, and Chao Lu. *Algorithms for Discrete Fourier Transforms and Convolution.* Springer–Verlag, New York, 1 edition, 1989.
[6] P. Duhamel and M. Vetterli. Fast Fourier Transforms: A Tutorial Review and a State of the Art. *Signal Processing*, 19:259–299, 1990.
[7] Matteo Frigo. A Fast Fourier Transform Compiler. *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 169–180, 1999.
[8] Dragan Mirković, Rishad Mahasoom, and Lennart Johnsson. An Adaptive Software Library for Fast Fourier Transforms. *Proceedings of the 2000 International Conference on Supercomputing*, pages 215-224, 2000.
[9] Dragan Mirković and Lennart Johnsson. Automatic Performance Tuning in UHFFT Library. *Proceedings of the 2001 International Conference on Computational Science, Lecture Notes in Computer Science 2073*, Vol. 1, pages. 71-80, 2001.
[10] Dragan Mirković and Lennart Johnsson. Automatic Performance Tuning for Fast Fourier Transforms. *International Journal of High Performance Computing Applications*, to appear.
[11] The PAPI web page. `http://icl.cs.utk.edu/projects/papi/`.
[12] The Performance Counter Library web page. `www.fz-juelich.de/zam/PCL/`.
[13] Home Page of the SPIRAL Project. `http://www.ece.cmu.edu/~spiral`.

A Knowledge Discovery Methodology for Behavior Analysis of Large-Scale Applications on Parallel Architectures *

Elias N. Houstis[1,2], Vassilios S. Verykios[1,3], Ann C. Catlin[2], and John R. Rice[2]

[1] Dept. of Computer and Commun. Engr., Univ. of Thessaly, Volos, GREECE
[2] Dept. of Computer Sciences, Purdue University, West Lafayette, IN, USA
[3] Research and Academic Computer Technology Institute, Patras, GREECE

Abstract. The focus of this paper is the application and extension of the knowledge discovery in databases process [5] developed in PYTHIA recommender system, to analyze the behavior of a DOE ASCI application/hardware pairs in the context of POEMS project[4]. The POEMS project has built a library of models for modeling scalable architectures like those in the ASCI program. Moreover, it supports detail simulation of a variety of state-of-the-art processors and memory hierarchies and incorporates parallel evaluation of discrete-event simulation. The driver application used is SWEEP3D.

Keywords: Recommender System, Data Mining, Knowledge Discovery, Scientific Computing

1 Introduction

One of the basic research issues in designing recommendation services is understanding the fundamental processes by which knowledge about scientific problems and their solutions is created. Some of this knowledge will come directly from experts - scientists and engineers - in the field. Other knowledge can be mined from experimental data arising from simulations. Yet, further knowledge will be learned from the experience gained by the recommendation service itself as it extracts performance knowledge about software components when these are running and applied to various problems.

The focus of this paper is the adoption of the knowledge discovery in databases process [5]of the PYTHIA recommender system, to the knowledge acquisition process for an ASCI application, in the context of the POEMS project [4]. PYTHIA [7, 10, 8], a "scientific recommender system" whose users are scientists/engineers, is trying to locate software module(s) appropriate to their needs. Given a scientific problem instance and performance criteria constraints on its solution, PYTHIA recommends an algorithm, estimates its parameters, and identifies a location on the web where a software module implementing the algorithm

* This work was supported in parts by NSF grants, EIA-0103688, DMI-0122214, and EIA-0205663.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 739–748, 2003.
© Springer-Verlag Berlin Heidelberg 2003

can be obtained. The current implementations of PYTHIA address the domains of numerical quadrature, elliptic partial differential equations (PDEs), domain decomposition and linear algebra. PYTHIA's basis for recommendations is a rule-bank mined by spooling performance trace data for various algorithms on benchmark scientific problems and correlating variations in performance to specific characteristics of the problem input. In addition, PYTHIA interfaces with the GAMS software repository on the web, where implementations of the implemented algorithm can be downloaded. In this paper, we specifically concentrate on the data mining methodology utilized for the design of PYTHIA.

The POEMS project is building an initial library of models, at multiple levels of granularity, for modeling scalable architectures like those envisaged under the DOE ASCI program. The project supports evaluation of component behaviors through application of discrete-event simulation at multiple levels of detail. POEMS supports detailed simulation of a variety of state-of-the-art processors and memory hierarchies and incorporates parallel evaluation of discrete-event simulation. The project is also building a knowledge base from mining performance data that can be used to predict the performance properties of hardware/software as a function of their characteristics. POEMS development is being driven by modeling of the behavior of large-scale complex applications on parallel architectures. The first driver application is the SWEEP3D program that is being used to evaluate advanced parallel architectures at Los Alamos National Laboratories.

This paper gives a short introduction of the issues addressed in POEMS project and presents the design and implementation of the POEMS knowledge base (POEMSKB) in the context of SWEEP3D application and two computational environments associated with SP2 parallel machine and a cluster of Sun20 workstations. This paper is organized as follows. In Section 2 we describe the performance methodology we followed in POEMS and the case study used. In Section 3 we summarize the components of the POEMSKDD approach in terms of the components of the PYTHIA system and in Section 4 we present the results of the KDD process for POEMS. Conclusions are listed in Section 5.

2 POEMS Performance Methodology and Sweep3D Case Study

The major elements of the conceptual framework of POEMS are a general model of parallel computation and the use of associative objects as the representation basis for components. The models developed in POEMS span three different domains: *application*, *operating system*, and *hardware*. The application domain represents parallel computation as a dynamic task graph where nodes represent sequential computation units and edges represent dependencies. The operating system domain provides the models for process and memory management, interprocess communication, and parallel file systems. The hardware domain provides models for the processor and memory components, where the latter includes models from cache memory as well as shared memory hierarchies.

As a proof of concept, the following performance analysis tools currently are being integrated in POEMS: an automatic task graph generator [1], the LogGP [9] and LoPC [6] analytic models, the MPI-SIM simulator [2], and the SimpleScalar instruction level processor/memory architecture [3]. Each tool contains capabilities for modeling the application domain, software domain, runtime system domain, and hardware domain. Together these tools provide the capability to analyze the performance of current and future applications and architectures, for very large and complex configurations and with a fairly high degree of confidence.

2.1 POEMS Driver Application: SWEEP3D

We consider the problem of predicting the performance of an ASCI kernel application called SWEEP3D on large scale parallel architectures such as IBM SP/2 or the SGI Origin2000. The benchmark code SWEEP3D represents the heart of a real ASCI application. It solves a 1-group time-independent discrete ordinates (Sn) 3D Cartesian (XYZ) geometry neutron transport problem. The XYZ geometry is represented by an IJK logically rectangular grid of cells. The angular dependence is handled by discrete angles with a spherical harmonics treatment for the scattering source. The solution involves two steps: the streaming operator is solved by sweeps for each angle and the scattering operator is solved iteratively.

This version of SWEEP3D is based on blocks of IJK and angles with a stride-1 line-recursion in the I-direction as the innermost work unit. The stride-1 I-line recursion is good for microprocessors and the blocked domains provide parallelism opportunities.

2.2 POEMS Performance Database

The POEMS performance database is to provide for the storage, retrieval and analysis of performance data, actual or estimated, for an application like SWEEP3D. The data includes: (a) actual measurements of performance, (b) data derived from actual measurements, e.g., a speedup curve that fits actual data, (c) simulation data derived from the "execution" of a low level simulator of the hardware and software environment, (d) specification data (for unavailable components of hardware or software), and (e) data derived from an analytical model, e.g., LogGP.

Currently the performance database has been modeled based on the underlying schema of the PYTHIA design. Following we present an instance of the PERFDATA table because of its major importance in the POEMS context. The PERFDATA record is shown in Figure 1. Some comments are in order. The record includes an *id* (name) and the system that is used for generating the performance data (PELLPACK). The database that this record correspond to is also listed (sweep3d), along with the name of the COMPOSITE record that includes the experiment that generated the data. The `perfind_set` attribute states the name of the file to be used that contain information related with the

```
name             |   sweep3d 50x50x50 experiment SP2-6
system           |   pellpack
comp_db          |   sweep3d
composite_id     |   sweep3d 50x50x50 experiment
perfind_set      |   sweep3d-std-par-grd
solverseq        |   50x50x50 1x2 sweep3d mk=1, mmi=1
rundata          |   IBM SP2 with 18 compute nodes
featurenames     |   {"matrix symmetric","problem type"}
featurevals      |   {"no","FEM"}
nproc            |   2
CSwitch          |   {8715, 8638}
SysCalls         |   {72886, 72790}
PgFlts           |   {2224, 2215}
PktsRd           |   {294, 247}
PktsSt           |   {186, 173}
mean_UserStat    |   {98.82, 98.98}
mean_KernelStat  |   {0.69, 0.53}
mean_WaitStat    |   {0.11, 0.06}
mean_IdleStat    |   {0.38, 0.43}
std_UserStat     |   {6.92, 6.87}
std_KernelStat   |   {2.61, 2.59}
std_WaitStat     |   {2.54, 1.45}
std_IdleStat     |   {5.21, 5.76}
SENDing          |   {1.1922, 0.748}
RECVing          |   {86.891, 5.4036}
BCASTing         |   {0.0008, 0.1129}
BARRIER          |   {0.0005, 0.0722}
ALLREDUCINGing   |   {0.0005, 0.0722}
perfproc2names   |   {"BytesTo","SendsTo","BytesFrom","RecvsFrom"}
perfproc2vals    |   {{{"5760000","14400","5760000","5760000"}},
                 |    {{"5760000","14400","5760000","5760000"}}}
modnames         |   {"domain processor","decomposer","triple"}
timeslice        |   {"elapsed","communication"}
time             |   {{{"0","0"},{"0","0"},{"302.1300049","2.7499979"}},
                 |    {{"0","0"},{"0","0"},{"301.7099915","2.7699978"}}}
total            |   {"302.1300049","301.7099915"}
```

Fig. 1. PERFDATA record for a 2-processor configuration in SP2.

data collection process. The particular sequence from the COMPOSITE record that generated the data is also referenced along with the RUNDATA record. In some cases, there are features that are dependent neither upon the application nor upon the underlying machine. These features are mostly related with the specific run, and for this reason have been included in the PERFDATA record such as the ones included as values for the attributes `featurenames` and `featurevals`. As we said, this record correspond to a parallel run of the SP2

system. For this reason, the number of processors of the parallel machine that the data was collected from is also explicitly stated in the record. In addition to the above information, there are four groups of metrics whose values are kept in the PERFDATA record. The first group corresponds to information related to the processes running in each processor and other system and user specific information. The second group of measurements include information related to the number of bytes communicated between every pair of processors. Actually, the corresponding field is a three dimensional array that states the *id*s of the processors communicated along with the number of bytes exchanged in one run of the application. The third group of measurements include timing information related to the various modules of the PELLPACK system. The last group include information related to the total timing of the application in per processor basis. Although PELLPACK allows us to extract timing information for each module, it does not provide us with the capability to take measurements related to the performance behavior of the underlying system. For this reason, we have incorporated into PYTHIA, various profilers that provide us with trace data from different platforms.

3 Components of POEMSKDD

The implementation of the POEMSKDD follows closely the design of PYTHIA system. The system consists of all the layers included in the architectural diagram of PYTHIA system with the only difference that the statistical analysis module performs a slightly different operation than simply ranking of system modules.

For the POEMSKDD we have made use of a well known clustering tool that partitions the space of the performance data independently of its dimension. For the interpretation of the actual meaning of the identified clusters, we use the decision tree inducer from the pattern extraction module of PYTHIA. After the classes have been described, the decision tree inducer runs once again in order to build a classification model to be used for prediction purposes. At this time, the inducer takes as an input the parameters of the machine/application pair along with the label assigned to each pair by the clustering tool for building a model that predicts the performance class of a certain combination of application/machine parameters. Following, we give a detailed description of the clustering phase for the SWEEP3D raw performance data.

3.1 Clustering of SWEEP3D Performance Data

The information we have collected for the SWEEP3D application, includes application and machine feature data as well as performance data from running the application in an SP/2 parallel machine. In particular we make use of the grid size for the three dimensions, x, y and z, the parameter mk (K-planes for blocking), and mmi (the angles for blocking), the number of processors used for executing the application, as well as measurements for the elapsed and communication time. Thus, we have generated a feature vector for our study of size 8. We

have only used this small number of variables for demonstration purposes and also because this set adequately reflects the variation in the underlying domain for observing the interesting relationships.

The first step in discovering knowledge from the collected data is to consider what kind of task must be applied in this case study. The main problem is that in such an environment where we are looking for patterns without having any a-priori knowledge of the trends that exist in them, we need to start the data analysis by identifying subsets of data with similar characteristics or else clusters. This is what is usually called as clustering. Clustering, is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data. For partitioning the performance data space into clusters in this study we have made use of the AutoClass system, a well-known clustering tool developed by NASA Ames Research Center.

The classes we are after in this knowledge extraction engagement are related to the performance of the underlying parallel machine for the various application parameters. For this reason, the cluster generation process is given as input only the performance information that is kept in the feature vector mentioned before. This information include the elapsed and communication time spent for executing the application in the parallel machine. The model we have selected for both variables is a Multi Normal model which implements a likelihood term representing a set of real valued attributes, each with constant measurement error and without missing values, which are conditionally independent of other attributes given the class. This is an assumption that we make and is related to the specific nature of the performance data we use. A pictorial representation of classes in the 2D space of performance data is given in Figure 2. The clustering tool identified 6 classes. The visualization tool has re-mapped the identified classes like follows: 4 to 1, 0 to 2, 1 to 3, 2 to 4, 3 to 5, and 5 to 6.

The performance data along with the corresponding classes identified by AutoClass are provided to a classification tool for the description of the performance classes. We have once again used the `ID3` software and the decision tree produced is presented in Figure 3.

3.2 Feature-Based Cluster Description

After the identification and description of performance classes, we need to associate them with application/machine features, in order to be able to decide or predict the performance class of a certain pair of application and machine configuration based on their static characteristics without having to run the application. This is not only very efficient, but sometimes it is the only feasible way. For example, in this manner we can investigate how the number of processors relates to the performance of the system, and based upon the results, we might decide to change the configuration of the system.

For this step, we use four features. The first feature is the problem size, which is the product of the 3 grid sizes in the 3 dimensions, the second is the number of processors (nproc) used for running the application, and the last two the mk and mmi parameters that are related closely to the application. We consider two

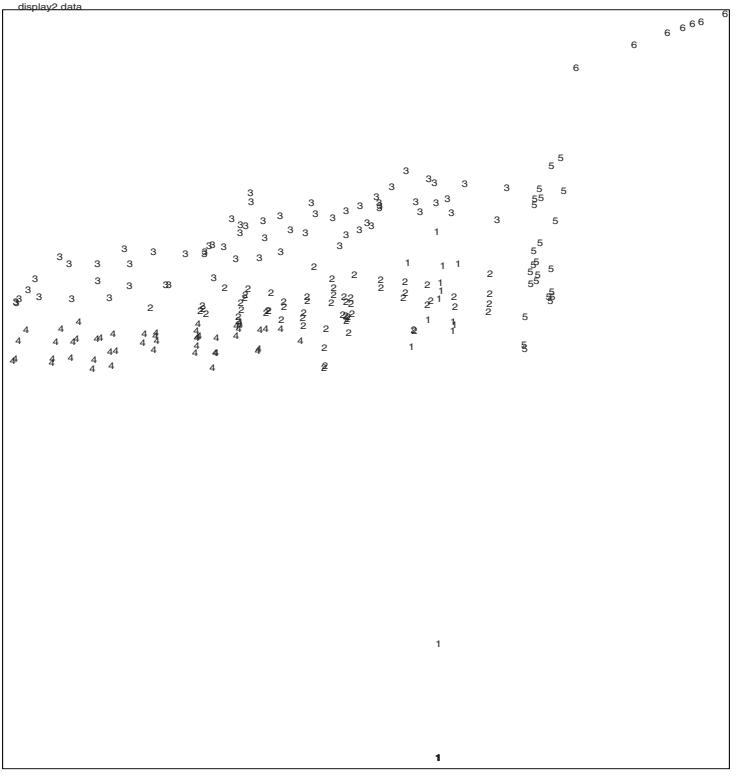


Fig. 2. Pictorial representation of the distribution of classes in the 2D space. The values of the `elapsed time` are depicted in the x axis and the values of the `communication time` are depicted in y axis.

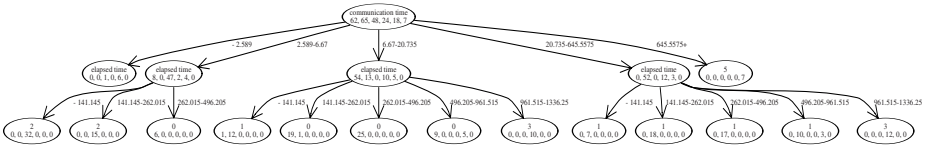


Fig. 3. Decision tree generated for the description of the performance classes identified by AutoClass.

of these variables, the number of processors and the problem size as continuous ones, while we consider the other two as categorical, because they only take values from a very small set of integers. The result of this classification task by using ID3 can be observed in Figure 4.

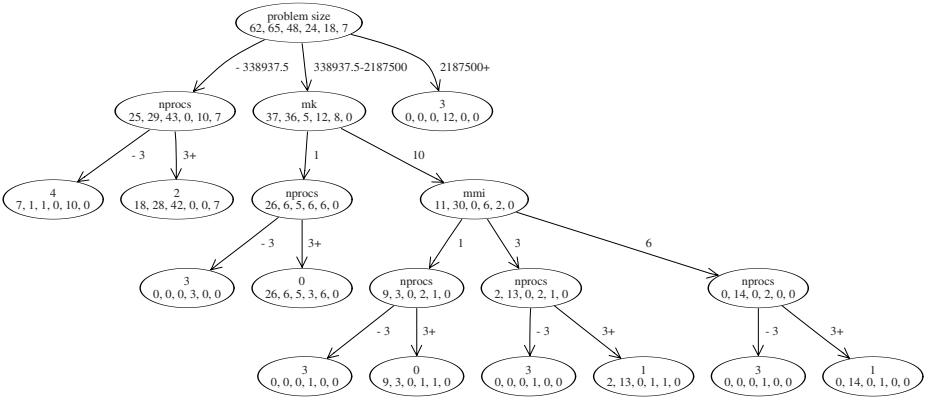


Fig. 4. Decision tree for identifying the performance classes based on static application and machine characteristics.

4 The Results of the KDD Process on SWEEP3D Performance Data

The decision tree corresponding to the description of the performance classes identified by AutoClass and shown in Figure 3 attains strong classification patterns, in the sense that the condition vector (ordered set of values shown inside a lead node) for the leaf nodes, contains cases/examples that belong to only one class. For example, we can state that the class 2 is very efficient with respect to both the communication and elapsed time.

The decision tree generated for predicting the performance classes based on static application and machine characteristics shown in Figure 4 does not deliver strong classification because more than one set of examples is covered by most of the leaves in the induced tree. This is an indication that the decision tree algorithm did not have enough features for explaining the data. This does not create any problem though for deriving rules that the recommender system can use for predicting the performance of a machine with some specific parameter values.

- A general observation that can be made from the data is that even for the minimum problem size of applications which were tried in these experiments, and the maximum number of processors, the application does not exhibit excessive communication volume, which means that the application scales really well for the problems sizes and number of processors considered in this case study.
- Another result which can be derived from the patterns in the decision tree (Figure 4) is that there is an ordering of the parameters used for explaining

the performance behavior. This ordering implies that the problem size has the highest discriminating power for separating the data into performance classes, the parameter mk has the second highest, the parameter mmi has the third highest and the number of processors has the lowest discriminating power.

- Classes 0 and 1 exhibit almost similar performance behavior with only difference that class 1 exhibits higher communication times than class 0. Both classes exhibit medium to high communication times and small to large elapsed times. This behavior can be attributed mainly to the large number of processors.
- Class 2 exhibits small communication and elapsed time. The large majority of the examples are characterized either by small problem size, and a large number of processors or medium problem size, or by medium problem size, $mk = 1$ and large number of processors.
- Class 3 exhibits medium to high communication time and a very high elapsed time. Half of the examples belonging to this class are characterized by large problem sizes. The other half is distributed equally between $mk = 1$ and $mk = 10$. For those having $mk = 10$, the examples are furthermore equally distributed among all the possible values of the parameter mmi. Notice that for all the previous cases the examples are equally distributed as well between the two categories for the number of processors.
- Class 4 exhibits very stable medium elapsed time but it spans a big range of communication times from small to medium-high ones. The examples falling under this class are characterized either by small problem size and small number of processors, or medium problem size and large number of processors.
- Class 5 exhibits high communication time. All the examples covered by class 5 are characterized by small problem size and large number of processors. This fact completely explains the high communication volume.

5 Conclusions

In this paper we have extended the KDD methodology of the PYTHIA system to address the selection of software/machine pairs and their parameters in the context of an environment for designing high performance application/machine pairs and predicting its performance. In this approach, the designer can utilize different performance models at the application, architecture, and OS levels. The goal of the KDD methodology in this paper is to identify various relationships of the system parameters involved with the application parameters. For example, the system can predict the optimum number of processors for a given machine out of many application partitionings or the best partitioning of the application with respect to various levels of grid sizes. PYTHIA has been extended to make these predictions. The PYTHIA database schema is adequate for accommodating the needs of various modeling requirements.

References

1. V. S. Adve and R. Sakellariou. Application Representations for Multi-Paradigm Performance Modeling. *International Journal of High Performance and Scientific Applications*, 1999. To appear.
2. R. Bagrodia and S. Prakash. Using Parallel Simulation to Evaluate MPI Programs. In *Proceedings of the 1998 Winter Simulation Conference WSC '98*, 1998.
3. Doug Burger and Todd M. Austin. The SimpleScalar Tool Set, Version 2.0. Technical Report Technical Report #1342, Computer Sciences Department, University of Wisconsin-Madison, June 1997.
4. E. Deelman, R. Bagrodia, A. Dube, J. Browne, A. Hoisie, Y. Luo, O. Lubeck, H. Wasserman, R. Oliver, R. Teller, D. Sundram-Stukel, M. Vernon, V. Adve, E. Houstis, and J. Rice. POEMS: End-to-end Performance Design of Large Parallel Adaptive Computational Systems. In *Proceedings of the 1st International Workshop on Software and Performance (WOSP) '98*, October 1998.
5. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press/MIT Press, 1996.
6. M. I. Frank, A. Agarwal, and M. K. Vernon. LoPC: Modeling Contention in Parallel Algorithms. In *Proceedings of the 6th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming (PPoPP '97)*, 1997. Las Vegas.
7. E. N. Houstis, S. Weerawarana, A. Joshi, and J. R. Rice. The PYTHIA Project. In S. K. Aityan, editor, *Neural, Parallel, and Scientific Computations*, pages 215–218. Dynamic Publishers., 1995.
8. Elias N. Houstis, Ann C. Catlin, John R. Rice, Vassilios S. Verykios, Naren Ramakrishnan, and Catherine E. Houstis. PYTHIA-II: A Knowledge/Database System for Managing Performance Data and Recommending Scientific Software. *ACM Transactions on Mathematical Software*, 26(2):227–253, 2000.
9. A. M. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating Long Messages into the LogP Model. In *Proceedings of the 7th Annual ACM Symposium On Parallel Algorithms and Architectures (SPAA '95)*, 1995. Santa Barbara, July.
10. S. Weerawarana, E. N. Houstis, J. R. Rice, A. Joshi, and C. E. Houstis. PYTHIA: A Knowledge Based System to Select Scientific Algorithms. *ACM Transactions on Mathematical Software*, 22(4):447–468, 1996.

Performance Modeling for Dynamic Algorithm Selection

Michael O. McCracken[1], Allan Snavely[2], and Allen Malony[3] *

[1] Department of Computer Science, University of California, San Diego
mike@cs.ucsd.edu
[2] San Diego Supercomputer Center
allans@sdsc.edu
[3] Department of Computer and Information Science, University of Oregon
malony@cs.uoregon.edu

Abstract. Adaptive algorithms are an important technique to achieve portable high performance. They choose among solution methods and optimizations according to expected performance on a particular machine. Grid environments make the adaptation problem harder, because the optimal decision may change across runs and even during runtime. Therefore, the performance model used by an adaptive algorithm must be able to change decisions without high overhead. In this paper, we present work that is modifying previous research into rapid performance modeling to support adaptive grid applications through sampling and high granularity modeling. We also outline preliminary results that show the ability to predict differences in performance among algorithms in the same program.

1 Introduction

Grid environments [1] present novel performance challenges, adding variability to many characteristics of high performance code. Heterogeneous platforms and varying network performance mean that the best algorithm for an application may change between runs of an application, and even during execution. Adaptive algorithms, developed to support portable performance in libraries, present an excellent opportunity to deal with these challenges by switching algorithms based on runtime information. To choose the optimal algorithm, a performance prediction must be made based on this information and the performance characteristics of the candidate algorithms. Because it is important to keep the combined overhead of measurement, modeling, prediction, and adaptation low, current time-consuming modeling techniques are not suitable for grid environments. We propose using a combination of ongoing research into rapid performance modeling and new development of a general adaptive algorithm framework to support exploration of portable performance on Grids.

We will initially focus our research on data-intensive applications running on Virtual Private Grids (VPGs), which combine the heterogeneity and physically

* We would like to acknowledge the help of Wayne Pfeiffer.[2]

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 749–758, 2003.
© Springer-Verlag Berlin Heidelberg 2003

distributed qualities of a public Grid, without the complication of distributed administration. Examples of important VPGs that this work is applicable to include GAMESS (General Atomic Molecular Electronic Structure Systems) Portal [2, 3], Encyclopedia of Life (EOL) [4], Biomedical Informatics Research Network (BIRN) [5], GriPhyn (Grid Physics Network) [6] projects, and Real-time observatories, applications, and data-management (ROADNet) [7].

2 Rapid Performance Modeling

Performance modeling is the science of constructing analytical and empirical models to predict and explain the performance of applications. Past techniques involve very detailed low-level predictions on a single architecture, as in [8–10], or developing custom models for single applications, for example [11]. Our goal is to develop general methods fast enough to update models at runtime, therefore such methods requiring expensive cycle-accurate simulation, slow evaluation, or human intervention are not suitable. Because it is sometimes possible to amortize the cost of more detailed analysis, we are investigating the tradeoff between analysis speed and prediction accuracy.

Current techniques for rapid performance modeling include convolution and Petri net methods. These approaches combine properties of abstraction and rapid computability. Only the most significant properties of an application are measured to facilitate building a model quickly. Our current approach uses convolution methods to model node-level performance, and Petri nets to model network performance. Current work using these methods has had success with HPC application modeling [12], so we propose to use a similar approach for VPGs.

2.1 Convolution Methods

It is now feasible to rapidly estimate the performance of an application across several candidate platforms, using techniques shown in [12–14]. We have developed a framework for automatic generation of performance models for whole HPC applications through convolution of application signatures with system profiles. Application signatures are summaries of the operations performed during execution. Ideally, signatures are machine independent. System profiles are measurements of the rates at which machines can carry out fundamental operations such as floating-point operations, memory accesses, message transfers, and disk accesses. Convolution methods are techniques for mapping signatures to profiles for predicting and understanding performance. The separation of signatures and profiles means that signatures need only be gathered once to support prediction on multiple machines. Convolutions are fast and allow exploration of varying accuracy by adding detail to application signatures.

2.2 Petri Net Methods

A Petri net is a tool for modeling concurrent systems. Details about Petri Nets are available in the references of [15], which describes Petri nets applied to Grid

performance prediction. In previous work with Petri nets using the DIMEMAS tool [16], we found that the performance of many data-intensive HPC applications is bottlenecked by their interaction with the memory hierarchy while their scalability is mostly a function of their interaction with the interconnect. Thus our application signatures focused on memory access patterns and communications patterns.

Our focus on rapid modeling of deep memory hierarchies and multi-tiered networks makes our previous modeling work most pertinent for modern HPC architectures. The fact that these same architectural factors are even more influential in Grid performance, along with the need to rapidly respond to changing network conditions and data sparsity, suggests that our methods will be useful to predict performance on Grids rapidly and cheaply.

3 Algorithmic Selection and Adaptation

Adaptive algorithms, also called poly-algorithms, encapsulate a number of algorithms for solution of the same numerical problem, along with a mechanism for selecting the best algorithm amongst the available alternatives [17–19]. The software mechanism responsible for making the determination of the best available choices at run-time is known as a switching function. The optimal choice of algorithm, can be determined at run time, typically using data such as measurements of hardware characteristics, system load, and input-data dependent properties of the application. Prototypes for adaptive dense matrix multiplication such as ATLAS [19], and FFTs [20, 21] show that adaptive algorithms can frequently do as well as or even better than hand-tuned vendor code.

Currently, adaptive algorithms and the switching functions employed by them run within standard HPC computing environments such as a tightly coupled cluster of SMP nodes [22] The next generation of adaptive algorithms must be able to run within distributed computational environments such as the TeraGrid [23], the NASA Information Power Grid (IPG) [24], or the Grid Physics Network (GriPhyN) [6], with heterogeneous compute resources and varied and fluctuating network bandwidths and latencies. Therefore, the ability to update algorithms choices automatically at runtime will become more important.

4 Applications of Performance Modeling in Adaptive Grid Software

Rapid performance modeling techniques can be used to enable real-time adaptive decisions in VPG environments. Such environments require rapid decisions about the best combination of software and hardware resources for the given application at a particular time. We are extending current rapid modeling techniques to grid applications, by identifying additional performance factors which must be measured to model grid applications. These include machine availability and heterogeneity, increased variance in network characteristics, and grid middleware service overhead.

We will continue to use convolution-based modeling to capture compute node-level characteristics of the VPGs such as expected CPU and memory subsystem performance. Petri net modeling will be used to model higher level characteristics such as varying network load, middleware service overheads, and machine availability. As our previous modeling work assumed a relatively fixed-state computing platform, our new research will explore how to make performance models capture changing conditions. Possible approaches we are evaluating include sampling of performance characteristics. We are currently extending the tracing technology to use the DynInst run-time code patching API [25] to allow sampling.

Present convolution methods gather signatures for an entire application. Using DynInst, we are extending the signatures to describe individual procedures and loops. This additional capability will allow us to use the convolution method to predict performance of candidate algorithms within an adaptive program.

5 Preliminary Results

Current investigations have already shown promising results in whole-application performance modeling with rapid evaluation [13]. To support modeling at a finer granularity to enable comparisons of candidate algorithms, we have extended the convolution method to the loop level. We are developing a tool similar to MetaSim [12] that gathers information at the loop level.

In order to validate that the convolution approach can work at the loop level, we show the results of predicting performance of a set of 8 synthetic loops which represent a range of memory usage and computational intensity. The loops were padded to avoid cache conflicts, and compiled using the IBM compiler with the following options: `xlf90 -O3 -qtune=pwr3 -bmaxdata:0x40000000`. Table 1 shows a summary of the loops studied. The experiments were performed on the SDSC Blue Horizon IBM SP, with 375 MHz Power3 processors.

Figure 1 shows the results of our convolution under development for use with loops. A simple linear fit using the L1 and L2 cache miss rates predicts the efficiency of a loop very well. The units are efficiency in terms of the limiting operation, which is either memory references or floating-point operations, depending on the loop. This set of loops will be expanded in ongoing work to include strided access, tiling optimizations, and then on to loops from application kernel benchmarks, to support building a robust convolution for modeling. These results were generated by hand and show that the dynamic instrumentation tool under development, which includes the fitted variables among those it measures, has the potential to support useful performance models.

6 Related Work

Past related work in the areas of performance modeling and adaptive algorithms are outlined in the following sections.

Table 1. Loops studied

Loop	Memory References per Iteration	FP operations per Iteration	Computational Intensity
fill			
$a(i) = s$	1	0	0
copy			
$a(i) = b(i)$	2	0	0
vadd			
$a(i) = b(i) + c(i)$	3	1	0.33
sum			
$s = s + a(i)$	1	1	1.0
daxpy			
$a(i) = a(i) + s * b(i)$	3	2	0.67
dot			
$s = s + a(i) * b(i)$	2	2	1.0
poly2	2	4	2.0
$a(i) = (c2 * b(i) + c1)$ $*b(i) + c0$			
poly5	2	10	5.0
$a(i) = ((((c5 * b(i) + c4)$ $*b(i) + c3) * b(i) + c2)$ $*b(i) + c1) * b(i) + c0$			

6.1 Performance Modeling

Gustafson and Todi first proposed the convolution method in [26]. Because cycle-accurate simulation is slow, there is a range of techniques that have been developed to improve its efficiency [10, 27–31]. These techniques are useful in domains such as architecture design but fall short of the rapid estimation capability needed for adaptable grid applications. To enable more rapid performance estimation, Saavedra [32–34] proposed to model applications as a collection of independent Abstract Fortran Machine tasks. For parallel system predictions, Mendes [35, 36] has proposed a cross platform approach. Traces are used to record the explicit communications among nodes and to build a directed graph based on the trace. Then sub-graph isomorphism is used to study trace stability and to transform the trace for different machine specifications. Simon [37] proposed to use a Concurrent Task Graph to model applications using the dependence relationship between nodes.

Scheduling using performance contracts is a current topic of grid research which requires models of expected performance. The GrADS Project [38] work on performance contracts measures the intrinsic application signature, monitors system processing and communication rates, and uses fuzzy logic to determine whether the application processing and communication speeds for each task are within the bounds defined by the user-specified performance contracts. However, algorithms for adjusting the application and/or system resource configuration

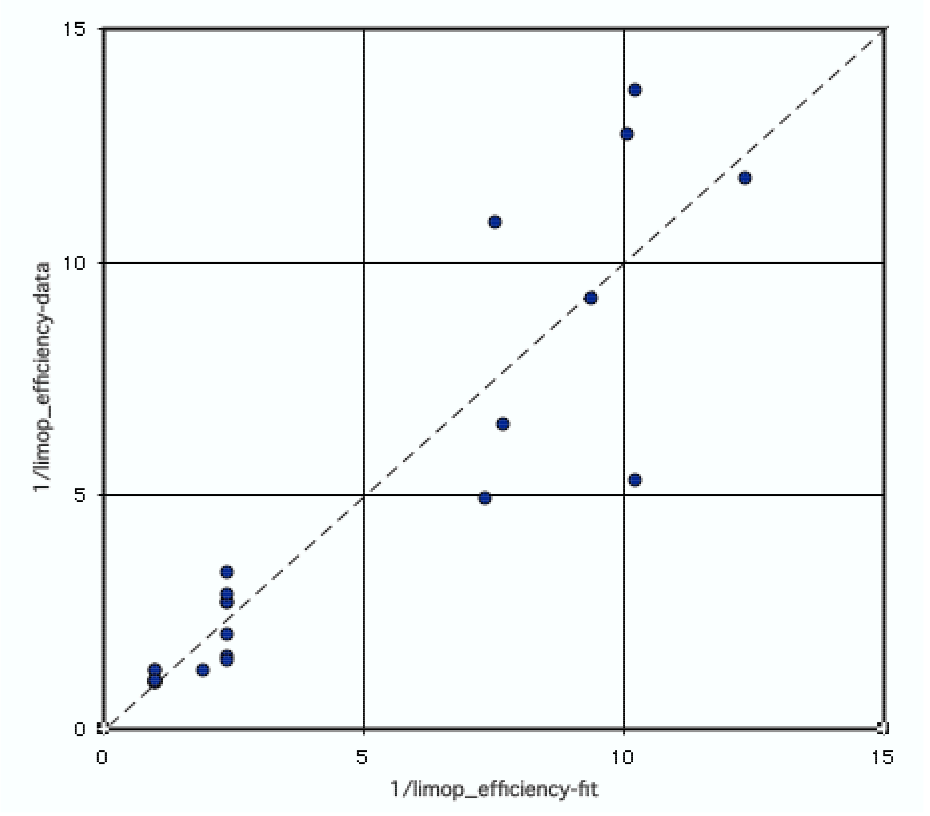


Fig. 1. Efficiency vs. predicted efficiency for loops in Table 1.

when the performance contract is violated are not yet addressed in those works. Furthermore, the performance models developed so far for GrADS are ad hoc and application-specific, for example the model developed for ScaLAPACK in [39]. Heymann et al. [40] measure task execution times and worker node efficiencies in each iteration of an executing Condor Master-Worker (MW) application. Results from previously measured synthetic MW applications provide a table of the estimated number of worker nodes to allocate, as a fraction of the number of parallel tasks, to achieve 80% efficiency with no more than a 10% increase in iteration execution time, based on the relative processing times of the largest 20% of the tasks. This algorithm assumes homogeneous processing nodes, that each task performs (approximately) the same work in successive iterations, homogeneous processing nodes, and that the application is synchronous (i.e., the results from one iteration are completed before a new iteration executes) while our target VPGs are heterogeneous and many of our applications are asynchronous. Hollingsworth and Keleher [41] propose an approach in the Harmony system, in which the application developer specifies the processing time and communication

time, or provides a model to predict these values at runtime, for each possible multidimensional configuration of the application. The Harmony system then dynamically allocates resources to each executing application to achieve a particular system objective (rather than application objective), such as maximizing throughput. By contrast our work is applications-centric.

The Delphi project [42] uses a language-directed whole-system approach to performance prediction and analysis. Compilers, instrumentation, custom libraries, and analysis tools all work together to support performance modeling. In contrast, our work does not require the same extensive system support. We focus on developing tools that are as lightweight as possible, specifically requiring minimal change in the application workflow. This focus will allow us to build performance models of existing applications and enhance decisions made by their adaptive libraries transparently.

6.2 Adaptive Algorithms

Some of the well-known numerical software packages with adaptive algorithms are the Linear Solver package (LINSOL) [43], ATLAS [19], HPL [44], Lapack for Clusters (LFC) [22], FFTW [20], and UHFFT [21]. The primary issue is how some of these packages use parameters for switching functions. In LINSOL, adaptive algorithms are chosen based on different numerical characteristics and their effect on convergence. All of these packages modify computation patterns in order to deliver high performance. Other aspects on which algorithms may adapt are data storage and communication patterns. All of these packages differ on criteria for switching functions based on functionality and goals specific to their application. In an alternative approach, Brewer made an extensive treatment of adaptive algorithms to achieve portable high performance sensitive to machine and instance issues [45]. The switching functions employed by Brewer are based upon statistically generated data from a database of empirical performance data. This approach, though highly accurate, is expensive, as shown by Sussman [46].

7 Conclusions and Future Work

Our targeted data-intensive Grid applications make substantial use of libraries, some of which already have adaptive features. Switching functions for Grid applications will need additional capabilities beyond those currently available on HPC platforms to deal with heterogeneity in compute resources and varying network conditions. Future work will investigate generalizing adaptive algorithms' switching functions to use output from general purpose performance models. These new switching functions will need access to a wider range of data, including past performance history, available machine characteristics, and configuration information about the Grid environment, including topology and network conditions. Custom development of performance models for each application is already time consuming, and this additional data will make it more so. Therefore, adaptive algorithms will benefit from separate development of automatic

performance models. Switching functions will then easily employ runtime performance estimates for making algorithm and machine selections, using the present work on rapid performance modeling.

References

1. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
2. Baldridge, K.K., Greenberg, J.P., Elbert, S.T., Mock, S., Papadopoulos, P.: QMView and GAMESS: Integration into the world wide computational grid. In: Proceedings of Supercomputing. (2002)
3. Schmidt, M., Baldridge, K.K., Boatz, J.A., Elbert, S., Gordon, M., Jenson, J.H., Koeski, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M., Montgomery, J.A.: The general atomic and molecular electronic structure system. Journal of Computational Chemistry **14** (1993) 1347 – 1363
4. EOL: Encyclopedia of life (2003) http://eol.sdsc.edu/.
5. BIRN: Biomedical informatics research network (2003) http://www.nbirn.net/.
6. GriPhyN: Grid physics network (2003) http://www.griphyn.org/.
7. ROADNET: Real-time observatories, applications, and data management network (2003) http://roadnet.ucsd.edu/.
8. Gibson, J., Kunz, R., Ofelt, D., Horowitz, M., Hennessy, J., Heinrich, M.: FLASH vs. (simulated) FLASH: Closing the simulation loop. In: Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). (2000) 49–58
9. Chatterjee, S., Parker, E., Hanlon, P.J., Lebeck, A.R.: Exact analysis of the cache behavior of nested loops. In: Proceedings of the International Symposium on Programming Language Design and Implementation (PLDI), ACM, ACM (2001)
10. Weikle, D., Skadron, K., McKee, S., Wulf, W.: Caches as filters: A unifying model for memory hierarchy analysis. Technical Report 16, University of Virginia Department of Computer Science (2000)
11. Kerbyson, D.J., Alme, H.J., Hoisie, A., Petrini, F., Wasserman, H.J., Gittings, M.: Predictive performance and scalability modeling of a large-scale application. In: Supercomputing, Denver (2001)
12. Snavely, A., Wolter, N., Carrington, L.: Modeling application performance by convolving machine signatures with application profiles. In: IEEE Workshop on Workload Characterization. (2002)
13. Snavely, A., Carrington, L., Wolter, N., Labarta, J., Badia, R., Purkayastha, A.: A framework for performance modeling and prediction. In: Supercomputing. (2002)
14. Carrington, L., Wolter, N., Snavely, A.: A framework for application performance prediction to enable scalability understanding. In: Scaling to New Heights Workshop, Pittsburgh (2002)
15. Anglano, C.: Predicting parallel application performance on non-dedicated cluster platforms. In: Proceedings of the 12th ACM International Conference on Supercomputing (ICS), Melbourne, Australia (1998)
16. Dimemas: Internet (2003) http://www.cepba.upc.es/dimemas.
17. Im, E., Yelick, K.: Optimizing sparse matrix-vector multiplication on smps. In: Ninth SIAM Conference on Parallel Processing for Scientific Computing, SIAM (1999)

18. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: PETSc 2.0 users manual. Technical Report ANL-95/11 - Revision 2.0.29, Argonne National Laboratory (2000)
19. Whaley, R.C., Petitet, A., Dongarra, J.J.: Automated empirical optimizations of software and the ATLAS project. Parallel Computing **27** (2001) 3–35 http://math-atlas.sourceforge.net/.
20. Frigo, M., Johnson, S.: FFTW: An adaptive software architecture for the FFT. In: Proceedings of ICASSP. Volume 3. (1998) 1381 – 1384
21. Mirkovic, D., Johnsson, S.: Automatic performance tuning in the UHFFT library. In: International Conference on Computational Science. (2001)
22. Roche, K., Dongarra, J.J.: Deploying parallel numerical library routines to cluster computing in a self adapting fashion. http://icl.cs.utk.edu/news_pub/submissions/dyn_pnumlibs.pdf (2002)
23. Teragrid: The national science foundation's teragrid project (2003) http://teragrid.org/.
24. Johnston, W.E., Gannon, D., Nitzberg, B.: Grids as production computing environments: The engineering aspects of NASA's information power grid. In: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing. (1999)
25. Buck, B., Hollingsworth, J.K.: An API for runtime code patching. The International Journal of High Performance Computing Applications **14** (2000) 317–329
26. Gustafson, J.L., Todi, R.: Conventional benchmarks as a sample of the performance spectrum. The Journal of Supercomputing **13** (1999) 321–342
27. Sherwood, T., Perelman, E., Hamerly, G., Calder, B.: Automatically characterizing large scale program behavior. In: International Conference on Architectural Support for Programming Languages and Operating Systems. (2002)
28. Haskins, J., Skadron, J.: Minimal subset evaluation: Rapid warm-up for simulated hardware state. In: Proceedings of International Conference on Computer Design. (2001)
29. Lafage, T., Seznec, A.: Choosing representative slices of program execution for microarchitecture simulations: A preliminary application to the data stream. In: Proceedings of the Third IEEE Annual Workshop on Workload Characterization. (2000)
30. Conte, T., Hirsch, M., Menezes, K.: Reducing state loss for effective trace sampling of superscalar processors. In: Proceedings of IEEE International Conference on Computer Design, IEEE Computer Society (1996) 468–477
31. Nussbaum, S., Smith, J.E.: Modeling superscalar processors via statistical simulation. In: 10th International Conference on Parallel Architectures and Compilation Techniques (PACT'01), Barcelona (2001)
32. Saavedra, R.H., Smith, A.J.: Analysis of benchmark characteristics and benchmark performance prediction. ACM Transactions on Computer Systems **14** (1996) 344–384
33. Saavedra, R.H., Smith, A.J.: Measuring cache and TLB performance and their effect on benchmark runtimes. IEEE Transactions on Computers **44** (1995) 1223–1235
34. Saavedra, R.H., Smith, A.J.: Performance characterization of optimizing compilers. Software Engineering **21** (1995) 615–628
35. Mendes, C.L., Reed, D.A.: Integrated compilation and scalability analysis for parallel systems. In: IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT). (1998) 385–392

36. Mendes, C., Reed, D.: Performance stability and prediction. In: Proceedings of the IEEE/USP International Workshop on High Performance Computing, Sao Paulo (1994)
37. Simon, J., Wierum, J.M.: Accurate performance prediction for massively parallel systems and its applications. In: Euro-Par 96. Volume 1124 of LNCS., France, Lion, Springer (1996)
38. Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Crummey, J., Reed, D., Torczon, L., Wolski., R.: The GrADS project: Software support for high-level grid application development. International Journal of High Performance Computing Applications **15** (2001) 327 – 344
39. Petitet, A., S.Blackford, J.Dongarra, B.Ellis, G.Fagg, K.Roche, S.Vadhiyar: Numerical libraries and the grid: The GrADS experiment with ScaLAPACK. International Journal of High Performance Computing Applications **15** (2001) 359 – 374
40. Heymann, E., Senar, M.A., Luque, E., Livny, M.: Evaluation of an adaptive scheduling strategy for master-worker applications on clusters of workstations. In: HiPC. (2000) 310–319
41. Hollingsworth, J.K., Keleher, P.J.: Prediction and adaptation in active harmony. Cluster Computing **2** (1999) 195–205
42. Delphi: (2003) http://www-pablo.cs.uiuc.edu/Project/Delphi/DelphiOverview.htm.
43. Weiss, R., Haefner, H., Schoenauer, W.: LINSOL (LINear SOLver) - description and user's guide for the parallelized version (1995) http://www.uni-karlsruhe.de/ linsol/html/documentation.html.
44. Dongarra, J., Luszczek, P., Petiet, A.: The LINPACK benchmark: Past, present and future (2001) http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf.
45. Brewer, E.: High-level optimization via automated statistical modeling. In: Proceedings of PPOPP. (1995)
46. Sussman, A.: Model-driven mapping onto distributed memory parallel computers. In: Proceedings of the 1992 ACM/IEEE conference on Supercomputing, IEEE Computer Society Press (1992) 818–829

Self-Adapting Numerical Software and Automatic Tuning of Heuristics*

Jack Dongarra and Victor Eijkhout

Innovative Computing Laboratory, University of Tennessee, Knoxville TN 37996, USA

Abstract. Self-Adapting Numerical Software (SANS) systems aim to bridge the knowledge gap that exists between the expertise of domain scientists, and the know-how that is needed to fulfill efficiently their computational demands. This know-how extends to algorith choice, computational grid utilization, and use of properly optimized kernels. A SANS system is a piece of meta software that mediates between the application program and the computational platform so that application scientists – with disparate levels of knowledge of algorithmic and programmatic complexities of the underlying numerical software – can easily realize numerical solvers and efficiently solve their problem.
The main component of a SANS system is an Intelligent Agent that automates method selection based on data, algorithm and system attributes. The IA uses heuristics to make its decisions. In this paper we explain how the heuristics of the IA can be tuned over time by redundant testing and using the nature of many applications.

1 Introduction

In numerous technologically important areas, such as aerodynamics (vehicle design), electrodynamics (semiconductor device design), magnetohydrodynamics (fusion energy device design), and porous media (petroleum recovery), production runs on expensive, high-end systems last for hours or days, and a major portion of the execution time is usually spent inside of numerical routines, such as for the solution of large-scale nonlinear and linear systems that derive from discretized systems of nonlinear partial differential equations. These numerical parts of the code can contain a large number of tuning parameters, the choice of which greatly influences the efficiency of the total code, or can even make the difference between obtaining a solution and obtaining none.

Such numerical concerns, however, are *artifactual* from the perspective of the scientific and engineering users, who are usually more concerned with modeling and discretization issues. The classic response to numerics was to encode the requisite mathematical, algorithmic and programming expertise into libraries that could be easily reused by a broad spectrum of domain scientists. However, in high-performance computing this solution is no longer sufficient. There is typically more than one algorithm for the stated purpose, and since several levels of

* This work was funded in part by the Los Alamos Computer Science Institute through the subcontract # R71700J-29200099 from Rice University.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 759–767, 2003.
© Springer-Verlag Berlin Heidelberg 2003

algorithms are needed in a large-scale application; the different algorithms can have interlocking parameter settings, and the availability of parallel computing platforms influences algorithmic decisions. Since the difference in performance between an optimal choice of algorithm and hardware, and a less than optimal one, can span orders of magnitude, it is unfortunate that selecting the right solution strategy requires specialized knowledge of both numerical analysis and of computing platform characteristics. Our SANS system aims to assist the application user to navigate this maze of computational possibilities.

This discrepancy in expertises can of course be bridged with manpower based solution: a large enough project can afford to hire a post-doc from computational science, or have a post-doc of its own cross-train in computational science. We believe, however, that it is possible have meta-software – software operating on software – to deliver as good a solution, without the wasted manpower. *Self-adapting Numerical Software (*SANS*)* systems have several levels on which they automate the computational choices for the application scientist.

1.1 Components of a SANS System

We will now describe the various aspects of a SANS system; we will go into details in the following sections. A SANS system comprises the following:

- An *Intelligent Agent* that includes an *automated data analyzer* to uncover necessary information about logical and numerical structure of the user's data, *a data model* for expressing this information as structured metadata, and a *self-adapting decision engine* that can combine this problem metadata with other information (e.g. about past performance of the system) in order to choose the best library and algorithmic strategy for solving the current problem at hand;
- A *history database* that not only records all the information that the intelligent component creates or acquires, but also all the data (e.g., algorithm, hardware, or performance related) that each interaction with a numerical routine produces;
- A *system component* that provides the interface to the available computational resources (whether on a desktop, in a cluster or on a Grid), combining the decision of the intelligent component with both historical information and its own knowledge of available resources in order to schedule the given problem for execution;
- A *metadata vocabulary* that expresses properties of the user data and of performance profiles, and that will be used to build the performance history database. By considering this as *behavioural metadata*, we are led to *intelligent software components* as an extension of the CCA [1,9] framework. The metadata associated with user input makes it possible for the user to pass various degrees of information about the problem to be solved. In the cases where the user passes little information, the Intelligent Agent uses heuristics to uncover as much of this information as is possible.

– One or more *prototype libraries*, for instance for sparse matrix computations, that accept information about the structure of the user's data in order to optimize for execution on different hardware platforms.

One of the more interesting aspects of a SANS system is that it will gradually increase its intelligence. To this end, the IA needs to tune its heuristics, and build new ones over time. In this article we will outline various approaches that can be taken to this end.

1.2 Further Outline of This Paper

We start off with a brief inventory of earlier research into adaptive systems in section 2. Section 3 further describes the metadata that is used internally and externally to store information about problems and algorithms. In section 4 we will go into further details on the Intelligent Agent, in particular describing the heuristic building process. Finally, in section 5 we briefly describe the system components of a SANS system.

2 Related Work

We note the following examples of earlier research into numerical poly-algorithms [20] and adaptive numerical software, focusing mainly on how our proposed research differs from, or often goes beyond, earlier approaches.

– Brewer [6] and Sussman [22] find an *a priori* model, accurately predicting runtime of their algorithms, for which they tune the parameters by measuring specific runs. By contrast, in our application no actual prediction of runtime is possible; we can only weight options against each other as more or less likely to solve the problem faster. Also, they have only a very finite number of algorithms to choose from, whereas our search space is for all practical purposes infinite.
– ATLAS [24] and Sparsity [16] optimize the dense and sparse BLAS [17], respectively. ATLAS relies on a one-time expensive installation, after which its use at runtime engenders no overhead. Sparsity has a runtime component that needs to be amortized, for instance over multiple systems with identical structure. These packages, and in particular ATLAS, are to a large extent independent of the nature of the user data, so no runtime decisions are needed.
– The LINSOL package of Weiss et al. [23] includes a poly-algorithm that picks one among a small number of iterative solvers through backtracking. No analysis of the linear system is performed prior to the iteration process; all decision making is done during the iteration and based on tracking the error norm. Also, any backtracking is only through the space of iterative methods; no ordering of preconditioners is attempted.

- LSA [12,18] is a project for componentizing linear solver software into a problem solving environment. Its stated goal is a visually programmed testbed, rather than an adaptive, intelligent, solver, although there is mention of integrating Case-Based Reasoning for some intelligent assistance.
- Houstis et al. [13] consider the problem of building a 'recommender system' out of a database of performance results. As a case study of their PYTHIA II system they consider solving PDEs with PELLPACK. However, their choices in linear system solvers are extremely limited.
- Some compilers already make use of trace data about previous executions in making decisions about compile-time optimization for code generation. Our approach extends this to both algorithmic choices and to runtime optimizations.

3 Metadata and User Interaction

The operations typically performed by traditional libraries are on data that has been abstracted from the original problem. For instance, one may solve a linear system in order to solve a system of ODEs. However, the fact that the linear system comes from ODEs is lost once the library receives the matrix and right hand side. By introducing metadata, we gain the facility of annotating problem data with information that is typically lost, but which may inform a decision making process. A SANS system will have the facility for the user to pass such metadata explicitly in order to guide the intelligent library. However, one needs to design heuristics for uncovering such lost data, taking the burden completely off the user.

3.1 Types of Metadata

The metadata passed by the user can not only be of varying levels of detail and sophistication, it can also lie on various points of a scale between purely numerical specification on the one extreme, and user application terms on the other. The former corresponds to the traditional parameter-passing approach of numerical libraries: users who are well-versed in numerics can express guidelines regarding the method to be used. However, most users are not knowledgeable about numerics; they can at most be expected to have expert knowledge of their application area. By building in a – heuristic – translation from application domain concepts to numerical concepts we allow the user to annotate the data in problem-native terms, while still assisting the SANS system in decision making.

3.2 Usage Modes of a SANS System

SANS systems can be employed in various ways, depending for instance on the level of expertise of the application user, and on the way the system is called from the application code.

- For a non-expert user, a SANS system acts like an expert system, fully taking the burden of finding the best solver off the user's hands. In this scenario, the user knows little or nothing about the problem – or is perhaps unable to formulate and pass on such information – and leaves it up to the intelligent software to analyze structural and numerical properties of the problem data.
- Users willing and able to supply information about their problem data can benefit from a SANS system in two ways. Firstly, decisions that are made heuristically by the system in expert mode can now be put on firmer ground by the system interrogating the user or the user passing on the information in the call. Secondly, users themselves can search for appropriate solution methods by using the system in 'testbed' mode.
- Finally, expert users, who know by what method they want to solve their problem, can benefit from a SANS system in that it offers a simplified and unified interface to the underlying libraries. Even then, the system offer advantages over the straightforward use of existing libraries in that it can supply primitives that are optimized for the available hardware, and indeed, choose the best available hardware.

4 The SaNS Agent

4.1 The Intelligent Component

The Intelligent Component of a SANS system is the software that accepts the user data and performs a numerical and structural analysis on it to determine what feasible algorithms and data structures for the user problem are. We allow the users to annotate their problem data with 'metadata' (section 3), but in the most general case the Intelligent Component will do this by means of automated analysis (section 4.2). Moreover, any rules used in analyzing the user data and determining solution strategies are subject to tuning (section 4.3) based on performance data (section 4.4) gained from solving the problems. Below we present each of these aspects of the SANS agent in turn.

4.2 Automated Analysis of Problem Data

Users making a request of a SANS system pass to it both data and an operation to be performed on the data. The data can be stored in any of a number of formats, and the intended operation can be expressed in a very global sense ('solve this linear system') or with more detail ('solve this system by an iterative method, using an additive Schwarz preconditioner'). The fewer such details the user specifies, the more the SANS will have to determine the appropriate algorithm, computational kernels, and computing platform. This determination can be done with user guidance, or fully automated. Thus, a major component of a SANS system is an intelligence component that performs various tests to determine the nature of the input data, and makes choices accordingly.

Some of these tests are simple and give an unambiguous answer ('is this matrix symmetric'), others are simple but have an answer that involves a tuning

parameter ('is this matrix sparse'); still others are not simple at all but may involve considerable computation ('is this matrix positive definite'). For the tests with an answer on a continuous scale, the appropriateness of certain algorithms as a function of the tested value can only be preprogrammed to a limited extent. Here the self-adaptivity of the system comes into play: the intelligence component will consult the history database of previous runs in judging the match between algorithms and test values, and after the problem has been solved, data reflecting this run will be added to the database.

4.3 Self-Tuning Rules for Software Adaptation

The Intelligent Component can be characterized as self-tuning in the following sense: The automated analysis of problem data concerns both questions that can be settled quickly and decisively, and ones that can not be settled decisively, or only at prohibitive cost. For the latter category we will use heuristic algorithms. Such algorithms typically involve a weighing of options, that is, parameters that need to be tuned over time by the experience gained from problem runs. Since we record performance data in the history database (section 4.4) of the SANS Agent, we have a mechanism to provide feedback for the adaptation of the analysis rules used by the Intelligent Component, thus leading to a gradual increase in its intelligence.

4.4 History Database

We can gather statistics during the actual problem solving process on a number of levels. There are tools for gathering hardware statistics, such as PAPI [7]. The vocabulary of statistics is also relatively well established in this case.

For the algorithmic level there is no standardization of this vocabulary. In the case of systems solving the obvious measure is the time to solution. However, this time is obviously influenced by hardware considerations (did the problem fit in memory or was page swapping a substantial part of the solution time) so these may need to be recorded too. In iterative methods there is the number of iterations to record, but more detail is obtained if we store the full convergence history. In fact, even more informative is storing the convergence history of the individual Ritz values. This of course requires a solver package (such as Petsc [2]) that allows such sophisticated iteration monitors to be installed.

4.5 Heuristic Building

We may view heuristics as a mapping from the space of problem properties to the space of possible algorithms and algorithm parameters. For each input, this mapping can yield either a specific output, or several outputs with a plausibility ranking; if one method fails, one can fall back one the next ranked.

Heuristics can be based on a single problem parameter, where reaching a certain value triggers a switch over from method to another, or a related variant of

it. Such a heuristic can be discovered by performing statistical analysis, correlating a range of the parameter against a range of performance results. For this to be possible, preferably we need performance results on a sequence of problems where only the parameter under investigation varies. There are several ways in which uch an analysis is possible.

- We can, given enough time, for instance during a setup phase of the system, run exhaustive tests, where a collection of test matrices is subjected to every available method.
- In a production system with only modest time criticality we can occasionally let a system be solved by two methods that only differ in one variable. Note that, given the right software setup, such redundant solving need not come at a 100% overhead cost. For instance, it was shown in [3] how many iterative methods have the same structure, so in running two methods side-by-side their communication stages can be combined, largely eliminating the communication cost of the second method. Even on a single processor, combining two methods can be efficient due to cache effects when a matrix that is operating on two vectors need not be reloaded for the second vector.
- In time-dependent problems and nonlinear solvers linear system occur that are of a gradually evolving nature. This also gives data from which statistical correlations can be harvested.

Heuristics based on multiple problem parameters are not so intuitively straightforward to construct, but the statistical techniques sketched above will still hold. However, since the method parameters usually vary discretely rather than continuously, the performance data stored will likewise be coarse grained. Finding correlations in a multi-dimensional space then requires more confirmation before we trust a thusly tuned heuristic.

5 System Components

5.1 Scheduler

The System Component of the SANS agent manages the different available computation resources (hardware and software), which in today's environment can range from a single workstation, to a cluster, to a Computational Grid. This means that after the intelligent component has analyzed the user's data regarding its structural and numerical properties the system component will take the user data, the metadata generated by the intelligent component, and the recommendations regarding algorithms it has made, and based on its knowledge of available resources farm the problem out to a chosen computational server and a software library implemented on that server. Eventually the results are returned to the user. Empirical data is also extracted from the run and inserted into the database; see section 4.4.

However, this process is not a one-way street. The intelligent component and system component can actually engage in a dialogue as they weigh preferred algorithms against, for instance, network conditions that would make the available implementation of the preferred algorithm less computationally feasible.

Part of the System Component is scheduling operations and querying network resources. Software for this part of a SANS system already exists, in the Netsolve [8], GrADS [4,19], and LFC [21] packages.

5.2 Libraries

Automation of the process of architecture-dependent tuning of numerical kernels can replace the current hand-tuning process with a semiautomated search procedure. Current limited prototypes for dense matrix-multiplication (ATLAS [24] and PHIPAC [5]) sparse matrix-vector-multiplication (Sparsity [15, 14], and FFTs (FFTW [11,10]) show that we can frequently do as well as or even better than hand-tuned vendor code on the kernels attempted.

Current projects use a hand-written *search-directed code generator* (SDCG) to produce many different C implementations of, say, matrix-multiplication, which are all run on each architecture, and the fastest one selected. Simple performance models are used to limit the search space of implementations to generate and time. Since C is generated very machine specific optimizations like instruction selection can be left to the compiler. This approach can be extended to a much wider range of computational kernels by using compiler technology to automate the production of these SDCGs.

References

1. R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. C. McInnes, S. Parker, and B. Smolinski. Toward a common component architecture for high-performance scientific computing. In *Proceedings of High Performance Distributed Computing*, pages 115–124, 1999.
2. Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.
3. Richard Barrett, Michael Berry, Jack Dongarra, Victor Eijkhout, and Charles Romine. Algorithmic bombardment for the iterative solution of linear systems: a poly-iterative approach. *J. Comp. Appl. Math.*, 74:91–109, 1996.
4. F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski. The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Applications and Supercomputing*, 15(4):327–344, Winter 2001.
5. J. Bilmes, K. Asanović, C.W. Chin, and J. Demmel. Optimizing matrix multiply using PHiPAC: a portable, high-performance, ANSI C coding methodology. In *Proceedings of the International Conference on Supercomputing*, Vienna, Austria, July 1997. ACM SIGARC. see `http://www.icsi.berkeley.edu/~bilmes/phipac`.
6. E.A. Brewer. High-level optimization via automated statistical modeling. In *Proceedings of Principles and Practice of Parallel Programming*, pages 80–91, 1995.
7. S. Browne, J Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *International Journal of High Performance Computing Applications*, 14:189–204, Fall 2000.

8. H Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 1997.
9. Common Component Architecture Forum. see `www.cca-forum.org`.
10. Matteo Frigo. A fast fourier transform compiler. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, Atlanta, Georgia*, May 1999.
11. Matteo Frigo and Stephen Johnson. Fftw: An adaptive software architecture for the fft. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Seattle, Washington*, May 1998.
12. D. Gannon, R. Bramley, T. Stuckey, J. Villacis, J. Balasubramanian, E. Akman, F. Breg, S. Diwan, and M. Govindaraju. Component architectures for distributed scientific problem solving. *IEEE CS&E Magazine on Languages for Computational Science and Engineering*. to appear.
13. E.N. Houstis, V.S. Verykios, A.C. Catlin, N. Ramakrishnan, and J.R. Rice. PYTHIA II: A knowledge/database system for testing and recommending scientific software, 2000. to appear.
14. Eun-Jin Im. *Automatic Optimization of Sparse Matrix - Vector Multiplication*. PhD thesis, University of California at Berkeley, May 2000. To Appear.
15. Eun-Jin Im and Katherine Yelick. Optimizing sparse matrix vector multiplication on SMPs. In *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, TX, March 1999.
16. Eun-Jin Im and Katherine Yelick. Optimizing sparse matrix computations for register reuse in SPARSITY. In Vassil N. Alexandrov, Jack J. Dongarra, Benjoe A. Juliano, René S. Renner, and C.J. Kenneth Tan, editors, *Computational Science – ICCS 2001. International Conference, San Francisco, CA, USA, May 2001, Proceedings, Part I.*, pages 127–136. Springer Verlag, 2001.
17. C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic linear algebra subprograms for FORTRAN usage. *Transactions on Mathematical Software*, 5:308–323, 1979.
18. http://www.extreme.indiana.edu/pseware/LSA/LSAhome.html.
19. A. Petitet, S. Blackford, J. Dongarra, B. Ellis, G. Fagg, K. Roche, and S. Vadhiyar. Numerical libraries and the grid: The GrADS experiments with scalapack. Technical Report ut-cs-01-460, University of Tennessee, Computer Science Department, 2001.
20. J.R. Rice. On the construction of poly-algorithms for automatic numerical analysis. In *Interactive Systems for Experimental Applied Mathematics. M. Klerer and J.Reinfelds*, pages 301–313. Academic Press, 1968.
21. Kenneth J. Roche and Jack J. Dongarra. Deploying parallel numerical library routines to cluster computing in a self adapting fashion, 2002. Submitted.
22. A. Sussman. Model-driven mapping onto distributed memory parallel computers. In *Supercomputing '92, Minneapolis, MN*, pages 818–829. IEEE Computer Society Press, 1992.
23. Rüdiger Weiss, Hartmut Höfner, and Willi Schönauer. LINSOL (LINear SOLver) – description and user's guide for the parallelized version. Technical Report 61-95, University of Karlsruhe, 1995.
24. R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, January 2001.

Computer Simulation of Multi-dimensional Bone Deformity Correction and Lengthening by Unilateral External Fixator

Yoon Hyuk Kim

School of Advanced Technology, Kyung Hee University, 1 Sochon-ni,
Kihung-eop,Yongin-shi, Kyongki-do, 449-701, Korea
yoonhkim@khu.ac.kr

Abstract. An Analytic and graphic model of a unilateral external fixator and bone system is presented to simulate and visualize the pre-operative planning of three-dimensional adjustments of the fixator joints in osteotomy. An analytic model was developed as a rigid body linkage system, and kinematic chain loop equations were utilized in the adjustment analysis to obtain the necessary rotations and translations at each joint of the fixator to correct bone deformity at the fracture site. Corrections of various deformities, combining various rotations and translations, were simulated to validate the models developed. The presented models achieve the goal of using computer-aided pre-operative planning and intra-operative execution of an osteotomy to correct malalignment and joint dysfunction.

1 Introduction

Knee osteotomy has been widely recognized as a treatment method for unicompartmental osteoarthritis of the knee [1–5]. This method has been used more frequently in relatively young and active patients with early degenerative changes in the joint [3]. The underlying principles for osteotomy are sound, but one of the causes of undesirable outcome of the treatment relates to a lack of knowledge based pre-operative planning and inadequate execution of the procedure as planned [6–8].

In the pre-operative planning stage for osteotomy, one crucial factor in obtaining good clinical results is the accurate determination of femoral-tibial alignment in order to redistribute joint contact pressure [6, 9–11]. Achieving the desired corrective angle is also difficult and unpredictable in preventing deformity recurrence. To accomplish accurate osteotomy, an examination of three-dimensional deformity, initial osteotomy plane orientation, and subsequent angular and translational adjustment, taking into account bone segment fixation stability must be carefully planned and executed.

External fixation has been widely used in bone segment stabilization following osteotomy [12–14]. External fixation has a distinct advantage in its ability to directly adjust the bone deformity intra-operatively in one setting or post-operatively adjust bone segments through gradual distraction at the osteotomy site. Furthermore, it al-

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 771–778, 2003.
© Springer-Verlag Berlin Heidelberg 2003

lows continuous monitoring of bone and joint alignment to avoid residual deformity in rotation and translation. To the author's knowledge, however, there has been no study conducted to evaluate the existing fixator's ability to achieve deformity correction based on certain application configurations and also determine the precise fixator joint adjustments in order to execute the correction plan accurately.

The objective of this study was to simulate a three-dimensional knee osteotomy of different bone and joint deformities using a unilateral external fixator to accomplish the required alignment correction. In order to determine the necessary fixator joint adjustments, the bone segments, the osteotomy site and the external fixator were modeled as a multi-link kinematic mechanism. The associated kinematic analysis under given bone deformities and fixator application configurations provided the degree of translation and rotation to be performed in each fixator joint in order to precisely correct a bone deformity. A computer graphic model of both the fixator and tibia were developed to visualize and validate the analysis results and to evaluate the osteotomy execution plan under external immobilization and adjustment. The presented technology assures the precise execution of the desired deformity correction.

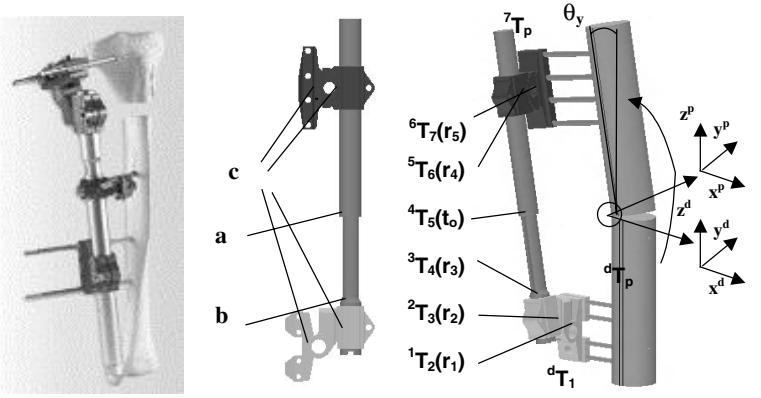
2 Materials and Methods

2.1 Mathematical and Computer Graphic Model

A unilateral external fixator for deformity correction (DynaExtor®, Bkmeditech, Korea) was used in this analysis. The fixator model is composed of eight pins inserted into the bones, a central telescoping rod, two pin clamps, two rotational rod clamps, and two connecting bodies which have two revolute rotations, respectively (Fig. 1). The analytical representation of the motion of each link of the fixator and the bone segments can be expressed as a kinematic linkage system interconnected by five revolute and one prismatic joints, thus having 6 degrees-of-freedom (DOF). The central rod of the fixator has a range of 5 cm of compression/distraction, and the rotational clamps have unlimited rotation along the long axis of its body. Both the distal connecting body and the proximal connecting body reach their full range of varus/valgus angulation at 45° of flexion/extension. In the model, all joints of the fixator begin in their neutral positions, with the exception of the lower revolute joint, which is rotated -10° in y-direction. Other geometric dimensions were measured and defined to facilitate the kinematic analysis (Fig. 1).

The bone was modeled as a cylinder, and the osteotomy cutting-plane was defined at diaphyseal area of the bone. The simple frontal plane angular correction was accomplished by rotating the distal bone segment with respect to the AP axis through the medial edge of the proximal bone segment in the frontal plane based on the specified wedge angle. The proximal and distal bone segments were modeled as two rigid bodies, with the distal bone segment fixed to the origin of the global coordinate system. Other three-dimensional osteotomy corrections including translation were defined as the input data specified by three rigid body Eulerian angle rotations and three translations, which were to be achieved by adjusting the fixator joints. The computer graphic

model of the bone-fixator system was developed using the software SolidWorks™ (Solidworks, Concord, MA, USA) in order to visualize the computed results of joint rotation and translation required to achieve the desired osteotomy correction.



(a) (b) (c)

Fig. 1. The real, (a), and the graphic, (b), models of the DynaExtor® external fixator (Bkmeditech, Korea): "**a**" – the central telescoping rod, "**b**" – the rotational joints, "**c**" – the revolute joints. The geometric parameters used in this paper were referred from the company catalog. (c) The tibia-fixator system for 3-D deformity correction with the transformation matrices, $^{i}\mathbf{T}_{i+1}$, and all the joint variables indicated on the DynaExtor fixator. The original varus deformity, θ_y, was defined as -10° of y-rotation

2.2 Inverse Kinematics Analysis

In order to define deformity parameters between the two bone segments, a 4 x 4 homogeneous transformation matrix was utilized to express the kinematic loop equations of the fixator-tibia system [15]. The rotational sequences at the fracture site follows the x-y'-z" Euler-angle system [15]. The global coordinate system was fixed to the distal segment of the tibia (Fig. 1). All local joint coordinate systems were defined identically to the global coordinate system. In the mathematical analysis, $^{d}\mathbf{T}_p$ is the transformation matrix from the proximal tibial segment to the distal segment, was expressed by the matrix or loop equation shown below:

$$^{d}\mathbf{T}_p = {}^{d}\mathbf{T}_1 \cdot {}^{1}\mathbf{T}_2 \cdot {}^{2}\mathbf{T}_3 \cdot {}^{3}\mathbf{T}_4 \cdot {}^{4}\mathbf{T}_5 \cdot {}^{5}\mathbf{T}_6 \cdot {}^{6}\mathbf{T}_7 \cdot {}^{7}\mathbf{T}_p. \quad (1)$$

$^{d}\mathbf{T}_1$ and $^{7}\mathbf{T}_p$ represent the rigid body translations of the local coordinate systems between the bone segments and the pin clamps (Fig. 2). The matrices $^{1}\mathbf{T}_2$, $^{2}\mathbf{T}_3$, $^{5}\mathbf{T}_6$ and $^{6}\mathbf{T}_7$ are pure rotations at the revolute joints. $^{3}\mathbf{T}_4$ represents the axial rotation at the lower rotational clamp, and $^{4}\mathbf{T}_5$ represents translation between the prismatic joints at

the central telescoping rod. After substituting the unknown fracture (${}^{d}\mathbf{T}_{p}$) and fixator geometric parameters into the transformation matrices, the six unknown joint variables, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, $\mathbf{r}_4$, $\mathbf{r}_5$ and $\mathbf{t}_o$, can be determined by solving the matrix Eq. (1), where, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, $\mathbf{r}_4$ and $\mathbf{r}_5$ are the revolute variables, and $\mathbf{t}_o$ is the translational variables, respectively [15] (Fig. 1). The resulting systems of non-linear equations were solved using the nonlinear least square optimization method (MATLAB™, Mathworks, MA, USA) [15]. In the optimization process, the limit of range in each joint was set as a constraint equation.

2.3 Simulation and Graphic Validation

To validate the presented analysis method, a correction of 10° varus deformity was simulated using graphic animation. In addition, anterior (-y-axis) translation, medial (-x-axis) translation, bone lengthening (z-translation), external rotation (z-rotation), and flexion (x-rotation) of the proximal segment in reference to the distal segment were combined with the varus deformity correction in order to fully illustrate the 3-D fixator adjustability and the analysis method used to achieve accurate alignment correction during treatment. Based on analysis results and graphic animation, the deformity correction planning process can be clearly visualized and evaluated.

3 Results

In order to demonstrate the adjustability of the fixator and the wide range of osteotomy options in 3-D space, both simple and complex malalignment cases were considered. Computer graphic models of the fixator and the bone segments were used to illustrate their pre-adjustment and corrected configurations.

In the case of a 10° varus deformity, the solution of the fixator joint variables for a closed wedge osteotomy site was obtained for the DynaExtor® fixator, as ($\mathbf{r}_1 = 0^{o}$, $\mathbf{r}_2 = 5.8^{o}$, $\mathbf{r}_3 = 0^{o}$, $\mathbf{r}_4 = 0^{o}$, $\mathbf{r}_5 = 4.2^{o}$, $\mathbf{t}_o = 10.8$mm) (Table 1). By rotating the revolute joints associated with $\mathbf{r}_2$ and $\mathbf{r}_5$ along the y-axis and translating $\mathbf{t}_o$, the fixator closes the wedge gap perfectly. The upper and lower bounds imposed on the rotational and prismatic fixator joints based on the fixator design constraints improve the solution convergence. According to the analysis results, the graphic simulation showed a perfect reduction of the wedge gap achieved by the above changes in the fixator joint variables (Fig. 2b).

In the case of a 10° varus deformity with 5 mm medial translation of the proximal segment in reference to distal osteotomy surface, the resulting solutions of the fixator joint variables for the osteotomy site were: $\mathbf{r}_1 = 0^{o}$, $\mathbf{r}_2 = 4.2^{o}$, $\mathbf{r}_3 = 0^{o}$, $\mathbf{r}_4 = 0^{o}$, $\mathbf{r}_5 = 5.8^{o}$, $\mathbf{t}_o = 11.5$mm (Table 1). When knee joint laxity needs to be increased, additional medial translation can be incorporated into the osteotomy in order to balance the knee ligaments. In this case, the same two revolute joints, which were involved in pure osteotomy reduction, now functioned to perform combined rotation and translation (Fig. 2c).

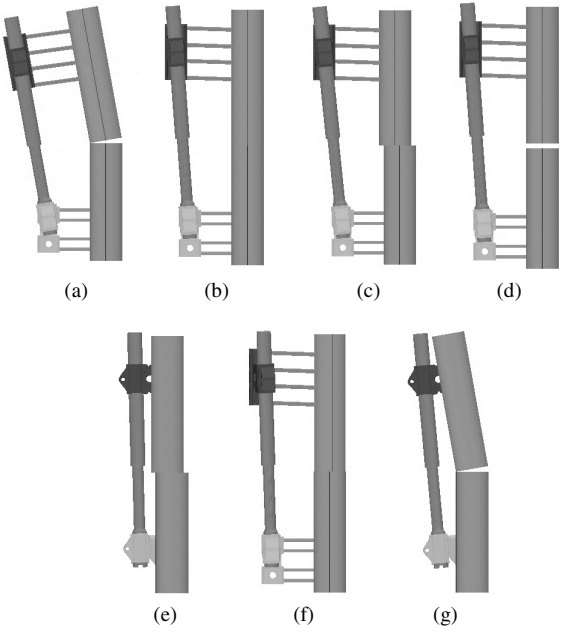


Fig. 2. (a) The bone-fixator configuration before the correction of deformity. (b)-(g) shows the bone-fixator configuration after the correction for pure correction; (b): correction with 5 mm medial translation, (c): correction with bone lengthening, (d): correction with 5 mm anterior translation, (e): correction with 10° external rotation, (f): and correction with 10° flexion, respectively. The results showed that different joint combinations contributed to perform the correction processes for different deformities and planning using the presented techniques would be useful before the intra-operation to obtain the accurate execution results

In the case of a 10° varus deformity with 5mm bone lengthening, the solutions of the fixator joint variables at the osteotomy site were: $\mathbf{r}_1 = 0^o$, $\mathbf{r}_2 = 5.9^o$, $\mathbf{r}_3 = 0^o$, $\mathbf{r}_4 = 0^o$, $\mathbf{r}_5 = 4.1^o$, $\mathbf{t}_o = 16.7$mm (Table 1). In this case, the result added 5 mm of the distal prismatic joint, $\mathbf{t}_o$, to the case of pure 10° of varus deformity correction. The graphic model shows the 5 mm of gap at the osteotomy site (Fig. 2d).

In the case of a 10° varus deformity with 5mm anterior translation at the osteotomy site, the solutions of the fixator joint variables at the osteotomy site were: $\mathbf{r}_1 = 1.5^o$, $\mathbf{r}_2 = 5.8^o$, $\mathbf{r}_3 = 0.1^o$, $\mathbf{r}_4 = -1.5^o$, $\mathbf{r}_5 = 4.7^o$, $\mathbf{t}_o = 11.1$mm (Table 1). In order to obtain the anterior translation of the bone segments, the two revolute joints $\mathbf{r}_1$ and $\mathbf{r}_4$ were rotated in the direction of X-axis, with the same magnitudes but opposite signs. Fig. 2e shows the lateral view (x-axis) of the fixator-bone system to validate the anterior translation of the system.

The case of a 10° varus deformity with 10° external rotation at the osteotomy site required all fixator joints to move significantly in order to perform the correction: $\mathbf{r}_1 = 3.7^o$, $\mathbf{r}_2 = 8^o$, $\mathbf{r}_3 = -9.6^o$, $\mathbf{r}_4 = -3.4^o$, $\mathbf{r}_5 = 2.6^o$, $\mathbf{t}_o = 11.4$mm, (Table 1). This result clearly

illustrates the capability of the fixator and the ability of the analysis software to generate 3-D movement at Knee osteotomy site. The Fig. 2f shows perfect adjustment of the bone segments as well as the axial rotations between the proximal and the distal osteotomy planes.

In the combined rotational deformity case with 10° varus and 10° flexion, the solution of the fixator joint variables osteotomy site were determined to be $\mathbf{r}_1 = 5.1^{\circ}$, $\mathbf{r}_2 = 5.8^{\circ}$, $\mathbf{r}_3 = -0.4^{\circ}$, $\mathbf{r}_4 = 4.9^{\circ}$, $\mathbf{r}_5 = 4.1^{\circ}$, $\mathbf{t}_o = 10.8$mm (Table 1). To obtain the flexion effect between the bone segments, the two rotational joints in the medial-lateral direction, $\mathbf{r}_1$ and $\mathbf{r}_4$, together rotated about 10°. In the graphic model, the rotations are clearly demonstrated between the two segments in the lateral view (Fig. 2g). With the use of an appropriate planning program that can define the correction required in all planes, the presented analysis technique can determine the precise adjustment required of each joint to achieve the specified osteotomy.

Table 1. Solutions of the DynaExtor® external fixator joint under 10° varus deformity correction combined various translations and rotations based on closed wedge osteotomy

Deformity Parameters	Fixator Joint Variables					
	$\mathbf{r}_1$ (°)	$\mathbf{r}_2$ (°)	$\mathbf{r}_3$ (°)	$\mathbf{r}_4$ (°)	$\mathbf{r}_5$ (°)	$\mathbf{t}_o$ (mm)
10° varus deformity	0.0	5.8	0.0	0.0	4.2	10.8
10° varus + 5mm medial translation	0.0	4.2	0.0	0.0	5.8	11.5
10° varus + 5mm bone lengthening	0.0	5.9	0.0	0.0	4.1	16.7
10° varus + 5mm anterior translation	1.5	5.8	0.1	-1.5	4.7	11.1
10° varus + 10° external rotation	3.7	8.0	-9.6	-3.4	2.6	11.4
10° varus + 10° flexion	5.1	5.8	-0.4	4.9	4.1	10.8

4 Discussion

Unicompartmental osteoarthritis (OA) of the knee is a common disorder due to the weight bearing, high-mobility and stability required of the joint [1–4]. Altered limb alignment in a knee with unicompartmental arthritis causes the load to shift towards the affected compartment, thus causing further joint deformity and degenerative changes. Therefore, joint alignment and loading redistribution are considered to be the key biomechanical justifications for knee osteotomy.

Despite the many advantages of knee osteotomy, it has not been favored as the treatment of choice even when clinical indications are favorable for such treatment because of several possible complications. Improper pre-operative planning and poor execution of the surgical plan have been identified as causes of deformity and pain recurrence [7]. In order to obtain good results, computer-assisted 3-D preoperative planning and execution of knee osteotomy is necessary [6, 9–12]. If an external fixator is used as the adjustment and stabilization device in a knee osteotomy, gradual and precise correction of the residual deformity at the osteotomy site can be accomplished by rotating and translating the fixator joints gradually on an adjustment plan.

When flexional/extensional deformities are combined with internal/external rotational malalignments, additional factors must be considered during preoperative planning, execution, and postoperative residual deformity readjustment. During osteotomy, there is a possibility of the tilting of the osteotomy line in the anterior/posterior direction due to the difficulty of saw blade angle control [8]. To maintain sagittal plane orientation of the tibial plateau it is important to avoid residual flexion/extension deformity of the tibia, since joint articular surface motion is improved when the normal posterior tilt of the tibial plateau is recovered. Furthermore, an intended internal/external rotation of the foot can be generated when the axis of the osteotomy closing is not perpendicular to the frontal plane. In addition to the intra-operative technical difficulties of knee osteotomy, rotational deformities from cerebral palsy or a congenital disease in which 3-D osteotomy is needed have been reported [16]. The current analysis software and graphic models can facilitate correction of all deformities at any level of the long bone by providing accurate and reliable corrections through fixator joints or robotic arms both in one-step correction and internal fixation or in gradual correction through callus distraction under external fixation.

The results of current analysis and simulation show that knee osteotomy can be achieved by adjusting a combination of translations and rotations of the fixator joints. Furthermore, the current analysis and simulation techniques can be used to determine the adjustment capability of the current design of any fixator, evaluate its performance during surgical treatment, and review its clinical application in individual patients. These kinds of tools can also provide the guidelines for a desirable biomechanical environment at fracture sites that enhances fracture union and bone maturation after lengthening. By integrating its use with image-guided and robotics-assisted tools, a fully controlled, 3-D, computer-aided osteotomy surgery and rehabilitation procedure can be established.

5 Summary

This paper presents a kinematic analysis and a graphic simulation tool for osteotomy under external fixation for complex 3-D bone deformities. Using the developed analysis tool, the required amount of the rotations and translations of the fixator joints were determined to reduce the given deformities in osteotomy with combined 3-D rotations and translations at the osteotomy site. In addition, the graphic model of the bone-fixator system developed in this study demonstrates its capabilities in validating analy-

sis results and simulating the pre-operative planning of osteotomy. Furthermore, these tools can be used to evaluate the existing planning methods of osteotomy, enhancing their performance and facilitating changes in the design configuration for the external fixator.

Acknowledgement. This study was supported in part by Res. Inst. of Med. Instr. & Reh. Eng., Kosef, Korea.

References

1. Coventry, M.B.: Upper tibial osteotomy for osteoarthritis. J. Bone Joint Surg. **67-A** (1987) 1136–1140
2. Coventry, M.B.: Proximal tibial varus osteotomy for osteoarthritis of the lateral compartment of the Knee. J. Bone Joint Surg. **69-A** (1987) 32–38
3. Fu, F.H., Harner, C.D., Vince, K.G.: The Knee Surgery Vol 2. Williams & Wilkins (1994)
4. Grelsamer, R.P.: Current concepts review: Unicompartmental osteoarthrosis of the Knee. J. Bone Joint Surg. **77-A** (1995) 278–287
5. Marti, R.K., Verhagen, R.A.W., Kerkhoffs, G, Moojrn, T.M.: Proximal tibial varus osteotomy. J. Bone Joint Surg. **83-A** (2001) 164–170
6. Hsu, R.W.W., Himeno, S., Coventry, M.B., Chao, E.Y.S.: Normal axial alignment of the lower extremities and load-bearing distribution at the Knee. Clin. Orthop. Related Res. **255** (1990) 215–227
7. Jackson, M., Sarangi, P.P., Newman, J.H.: Revision total Knee arthroplasty. J. Arthroplasty **9** (1994) 539–542
8. Ellis, R.E., Tso, C.Y., Rudan, J.F., Harrison, M.M.: A surgical planning and guidance system for high tibial osteotomy. Comput. Aided Surg. **4** (1999) 264–274
9. Paley, D., Tetsworth, K.: Mechanical axis deviation of the lower limbs; Preoperative planning of uniapical deformities of the tibia and femur. Clin. Orthop. Related Res. **280** (1992) 48–64
10. Chao, E.Y.S., Sim, F.H.: Computer-aided preoperative planning in Knee osteotomy. The Iowa Orthop. J. **15** (1995) 4–18
11. Chao, E.Y.S., Neluheni, E.V.D., Hsu, R.W.W., Paley, D.: Biomechanics of malalignment. Orthop. Clin. North America **25** (1994) 379–386
12. Paley, D., Chaudray, M., Pirone, A.M., Lentz, P., Kautz, D.: Treatment of malunions and mal-nonunions of the femur and tibia by detailed preoperative planning and the Ilizarov techniques. Orthop. Clin. North America **21** (1990) 667–91
13. Choi, I.H., Chung, C.Y., Cho ,T.J., Park, S.S.: Correction of genu recurvatum by the Ilizarov method. J. Bone Joint Surg. **81-B** (1999) 769–774
14. Larsen, S.T., Magyar, G., Onsten, I., Ryd, L., Lindstrand, A.: Upper tibial valgus osteotomy using a dynamic external fixator. J. Bone Joint Surg. **80-B** (1998) 295–297
15. Kim, Y.H., Inoue, N., Chao, E.Y.S.: Kinematic simulation of fracture reduction and bone deformity correction under unilateral external fixation. J. Biomech. (2002) 1047–1058
17. Healy, W.L., Anglen, J.O., Wasilewski, S.A., Krackow, K.A. Distal femoral varus osteotomy. J. Bone Joint Surg. **70-A** (1988) 102–109

Problems and Prospects for Quantum Computational Speed-up

E.V. Krishnamurthy

Computer Sciences Laboratory,
Australian National University, Canberra, ACT 0200, Australia.
abk@discus.anu.edu.au

Abstract. This paper studies the problems involved in the speed-up of the classical computational algorithms using the quantum computational paradigm. In particular, we relate the primitive recursive function approach used in computability theory with the harmonic oscillator basis used in quantum physics. Also, we raise some basic issues concerning quantum computational paradigm: these include failures in programmability and scalability, limitation on the size of the decoherence – free space available and lack of methods for proving quantum programs correct. In computer science, time is discrete and has a well-founded structure. But in physics, time is a real number, continuous and is infinitely divisible; also time can have a fractal dimension. As a result, the time complexity measures for conventional and quantum computation are incomparable. Proving properties of programs and termination rest heavily on the well-founded properties, and the transfinite induction principle. Hence transfinite induction is not applicable to reason about quantum programs.

1 Introduction

Quantum theory of computation is an important area of fundamental study linking the physical quantum theory with the logical theory of computability. From a pragmatic point of view, we are not even sure whether this study will lead us to the design of a quantum computer in the near future or almost never! Although, quantum version of boolean logic circuits [4, 6, 8, 9, 23] have been shown to be feasible, the machine hierarchy (namely, finite state machines, push-down stack machine and Turing machine) and the corresponding Chomskian grammatical and linguistic hierarchy [11,13], their connection to algorithms and related data structures have not been established in quantum computational logic, leaving a wide gap in understanding. Yet, some problems have been shown to be amenable for quantum speed-up [4,8,10,23]. These algorithms have been ad hoc discoveries using the principle of quantum superposition (Schrodinger cat states) and are probabilistic. Based on these, there have been several claims that quantum speed-up can break the computational intractability, as well as, the Turing barrier of noncomputability. But so far there has been no success to prove these claims due to the difficulty in bridging a physical theory with the logical computability theory. In fact, extending the recursion theory over the integers to real and complex numbers to include approximate operations that are typically composition of functions and solutions of differential equations [3] do not

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 779–788, 2003.
© Springer-Verlag Berlin Heidelberg 2003

result in newer complexity classes and so the standard computational complexity measures remain intact. Yet another obstacle is set by Physics, due to decoherence. Decoherence prevents coupling with the environment thereby making the quantum speed up ineffective [2, 10, 23]. Thus this is a challenging area for innovative experimental and theoretical research seeking answers to many questions.

The purpose of this paper is to explore the reasons as to why the quantum paradigm is not so easy to extend to all of the classical computational algorithms. Firstly, our arguments based on quantum field theory (second quantization) leads us to the result that quantum speed-up can be effective only for the primitive recursive functions (or the unravelled "for loop" or "count loop" programs), which correspond to the harmonic oscillator functions, and not for total or partial recursive functions ("general while loop" programs).This places a severe restriction on the type of algorithms that can be mapped to gain quantum-speed-up. Secondly, the problem of designing quantum algorithms through superposition is analogous to designing algorithms based on the "unrealistic Concurrent-Write- Exclusive Read PRAM" model [14]. That is we need a quantum measurement that can extract all of the 2^m values of the component functions from their superposition pattern. The only way to extract information from the pattern is through the recognition of joint properties among the values of the function f(x), for example, the periodicity, using quantum probabilistic means. However, extracting a definite answer from this pattern is highly problem domain dependent and is not easy although, Hadamard or Fourier transform and other filtering or projection techniques are useful for particular problems.

While superposition principle provides space complexity reduction for time-independent parallel computation, there remains yet another possibility called "*Zeno squeezing*", in which an infinite amount of computation can be performed in a finite time through a time scale change, since time is a real number in Quantum Physics (See Section 6.5). This aspect remains to be explored in quantum speed-up.

In this paper we consider the following issues:

1.To study why quantum algorithms are hard to design and what are the requirements for realising such algorithms?

2.What are the current open problems and what are the limitations in our knowledge?

The aforesaid studies lead to the following results:

(i) Errors in encoding and difficulty in Mapping: Quantum computation work with a continuum of states and use unitary transformation framework over complex numbers rather than conventional boolean gates. In this sense they are analog;hence inexact and are subject to operational errors. Therefore, realising an arbitrary unitary operator with arbitrary precision is not easy. For example, one of the most common quantum gates rotates a quantum bit by an angle; this can result in error causing operational errors to propagate [24].

Quantum paradigm requires that the problems are converted into a periodic superposition of functions. This approach is different from the mapping of processes to architectures, in the sense, it is a structural transformation of a problem from a given domain to a non-dissipative quantum dynamical system. Also the absence of suitable supporting data structures-such as- stack and other data structural mechanisms required in the design of algorithms are not available.This makes programming conceptually difficult.

(ii) Failure of Programmability: Quantum nonlocality [10, 17, 21] creates non-anticipated interactions resulting in the failure of a total program.So the software

engineering requirement for modular construction of a whole program from its parts, or the principle of denotational semantics is violated. Thus programming a quantum computer through a recursive set-up is not possible.
(iii) Failure of Composition: Conventional programs are built as a composition of primitive recursive functions over the integers. The quantum programs need to realise such functions using a composition of the different real or complex functions for each subprograms. This would result in both operational errors and dissipation leading to accelerated decoherence. Hence the divide and conquer strategy (or related strategies used in algorithmic engineering) in which we decompose a problem into subproblems and composing the results fails [18].
(iv) Failure of Scalabilitty: Scalability is a metric that tells us how the running time to solve a problem scales or varies with its size. Assume that we use the divide and conquer strategy. If we partition the problem into N pieces and each such partition is solved by N different processors simultaneously, and the algorithm is linearly scalable, we expect that the total problem is solvable in the same time as solving a single partition plus some overhead for combining the N solutions. Since each such partition is a different coherent state, their phase coherence is to be preserved before combining the results in order to avoid an accelerated decoherence. The preservation of coherence is difficult if the different partitions are not identical as in nontrivial computations.Hence partitioning a problem and solving is ruled out in quantum methodology.Hence scalability fails and we need bigger and bigger machines to solve larger and larger problems.
(v) Quantum Decoherence in while loop: The reason why we cannot use "while loop" for testing is obvious. Each such test in a loop destroys the coherence. Hence "while loop" programs are forbidden quantum mechanically. However, the unravelled "for loop" computations that are primitive recursive are realisable as coherent harmonic oscillator states. But they cannot be composed due to accelerated decoherence. Some attempts are under way to prevent accelerated decoherence by choosing *decoherence-free space* so that phase coherence between such states remains intact.
(*vi*) *Proving Quantum Programs Correct*: The structure of time, namely, whether it is continuous or discrete, plays an important role in reasoning, verification and proving the correctness of programs. In computer science, time has a discrete well-founded structure, while in physics, it is well accepted that the structure of time is a real number and is continuous and is infinitely divisible. In fact time can have a fractal dimension too. This means the transfinite induction principle is not applicable to reason about quantum programs. As a result, the time complexity for conventional and quantum computation are incomparable.
(vii) Curse of dimensionality: We have shown elsewhere [18] that quantum speed-up cannot obviate the curse of dimensionality encountered in many numerical problems due to quantum entropy arsing from operational and decoherence errors.
(viii) Zeno-squeezing: Is there a physical possibility of Zeno squeezing of time through a time-scale change for quantum speed-up? (Section 6.5)

2 Quantum Algorithmic Paradigm

Quantum algorithmic paradigm is the realisation of a physical quantum system whose dynamical evolution takes the system from one of a set of input states to one of a set

of output states. If we assume that a quantum computer consists of a set of quantum components and a quantum register, the quantum computation can be viewed as a process which consists in preparing the input state S(in) and evolving it to an output state S (out). The output state is a positive map of the input state through a linear operator transformation that results in a superposition of all m input values into a single state resulting in a pattern formation (Concurrent Write without any constraint - unknown to computer science theory). The quantum measurement is then supposed to extract (or Exclusive Read) of all the 2^m values from their joint properties, e.g. periodicity. It is assumed that such a process preserves quantum coherence in the register since it is the quantum coherence that provides the required quantum speed-up. Thus quantum decoherence should not take place before the reading.

3 Quantum Evolution, Dissipation, and Coherence

In [15, 16] we studied the relationship between the notion of computability and complexity in computer science, and the notion of integrability in mathematics and physics to understand the complexity aspects of quantum computing**.** In [15], using the Feynman path integral (FPI) approach we showed that a quantum computer behaves like an analog of an Alternating Turing machine (ATM) exhibiting and-or parallelism by computing the probability amplitude for the reachability (transitive closure) between a given initial and the final algorithmic state. We also stated the two important requirements for the success of quantum evolutionary computation, namely:

1. The evolutionary structure of the exact algorithmic solution of a given mathematical problem is mappable to the "exact unitary" evolutionary structure of a quantum dynamical system leading to the exact spectrum, and these two evolutionary structures have isomorphic histories between the initial, intermediate and final states of the computation. Any approximation would incur degenerative operational errors.
2. The initial, intermediate and final states of the quantum dynamical system are coherent states and the evolution of the hamiltonian preserves coherence. In other words, decoherence does not take place during the progress of computation, and before the final result is read by an observer. This requirement heavily hinges upon the theory of quantum measurements.

The requirements 1 and 2 demand that the evolution is nondissipative,Braun [2] and is also coherence preserving. The requirements, namely, coherence preserving evolution, as well as nondissipative can be achieved if the quantum dynamical system belongs to two important group structures, SU(2) and SU(1,1), [1, 16]. Thus to have a coherence preserving, unitarily evolving, exact quantum computation it is sufficient that the class of computational problems have models isomorphic to SU(2) or SU(1,1) group theoretical structures. This class contains instances of periodic problems mappable to harmonic oscillator, e.g., Shor's factorization, and instances of problems mappable to the permutation groups e.g., sorting, Nielsen and Chuang [23]. Note that for the SU(2), SU(1,1) unitary processes (dealing with photon number measurement) in a quantum optical system, Ban [1] shows that the entropy decrease of the physical system is equal to the amount of information extracted from the measurement outcomes. That is there is no loss of information and the photon number is exactly obtained. This class of problems corresponds to synchronous coherently evolving non-

interacting parallel streams of computations whose hamiltonians are amenable for easy composition and analysis using quantum optical devices and molecular spin-resonance devices [23]. However, even for such computations the time scale necessary turns out to be not practical. This is because a million computations are to be carried out and the final results read-off in a thermal time scale time less than 10^{-11} second, at a very cold temperature as low as 1 degree Kelvin, Landauer [21]. Thus the unitary quantum dynamical evolution is restricted to solve only problems that can be mapped on to non-dissipative hamiltonian systems that can evolve without decoherence.

4 Primitive Recursion and Harmonic Oscillator Functions

Two approaches for modelling computability are [11,13,14]:
(i) Turing machine and (ii) Recursive function.

In the last decade the Turing machine model, which is an operational model has been widely used as the basis for modelling the Quantum mechanical computers, see Deutsch [4], Hey and Allen [9] to prove the universality of quantum computers. We may call the Deutsch approach as the first quantisation approach since it is concerned with the wave aspects of particles (Wavicles) rather than the particle aspect of waves. Although, the Turing machine approach is equivalent to recursive function approach, the latter is more suitable than the former for analysis [17]. The QFT approach is called the *second quantisation scheme* in Quantum physics.

The quanta of the radiation field corresponds to the non-negative integers and the harmonic oscillator spectra correspond to the recursive computation- with the projection, creation and annihilation operators respectively playing the same role as the Selection, Successor and Predecessor in computability theory. Accordingly this approach [17] relates the classical computational models and the quantum field models more directly than the Turing machine approach used earlier [4,9].

If we identify the successor function used in recursive function theory with the bosonic creation (or construction or raising) operator, and predecessor function with the bosonic annihilation (destruction or lowering) operator, then we can generate all the primitive recursive functions (with the quanta as the basic unit of computation) consisting of the function algebra, consisting of some basic functions and closed under certain operations:
{ZERO, SUCCESSOR, PREDECESSOR, PROJECTION, COMPOSITION, PRIMITIVE RECURSION, BOUNDED MINIMALIZATION} [11,13,17] .
Thus, *the harmonic oscillator plays the same role in the quantum physics, as the primitive recursive function generator does in the theory of computation.*

The use of harmonic oscillator as the basis, permits us to only compute primitive recursive functions, but not a total recursive function (such as the Ackerman function) or partial recursive function (a general "while loop") [11,13,17].

The precise counting of bosons or phase determination [17,18] provide mutually exclusive approaches for designing three types of primitive recursive exact digital Quantum algorithms, see [23]. In quantum computers, a string of bits is stored of a system equivalent to m 2-level systems (quantum bits or q-bits). The general state is then a superposition of states u(i) where i runs through 0 to $2^m - 1$ values of a classical bit string. If the devices are not error-prone and there is no loss of energy, then such a

superposition is identifiable with a primitive recursive computable function,if the coefficients in the linear combination are exact numbers or exact phases, since the number-phase relation forbids the exact determination of both [17,18].We can therefore create a coherent superposition of only phases or amplitudes respectively creating Schrodinger phase cats or amplitude cats, Braun [2]. To avoid operational errors, we need to use exact phases as rational numbers using the prime residue or p-adic rationals [7,12]. No speed-up methods are known for fast quantum computer arithmetic.

4.1 Decoherence-Free Sub-Space (DFS)

Decoherence is a key ingredient in the transition from quantum to classical mechanics. It has been shown [2, 21,24] by that in a macroscopic system the decoherence time scale or the time required for loss of coherence is many orders of magnitude less than the classical time scale over which probabilities evolve. However, some exceptions seem possible and one can look for decoherence free subspaces (DFS). Braun [2] shows that if the dimension of the Hilbert space is 2^N, and superrdiance is present, the dimension of the DFS scales as: $[2^{N+1/2} / \sqrt{\pi N}]$, for large N. That is a fraction $\sqrt{2}/\pi N$ of the Hilbert space can be decoherence free. This means the input and output domain should remain within the DFS. Obviously, Ackermann function would outgrow any given DFS super-exponentially.

5 Programmability and Scalability

In programming theory, we define each language construct in terms of certain mathematical entities (numbers, truth values, functions, operators) that model their meaning. The total meaning of a program is then reconstructed as a composition of the meanings of the individual basic constructs; or “a whole “is described in terms of its “parts”. If the program is recursively defined, its meaning is the input-output function corresponding to the least or minimal fixed point of a transformation associated with the program [13, 22]. This ensures that the program has a well-defined functional relationship between the input and output states and the program halts. This assumption also holds true in classical mechanics where the real states are always local and do not contain statistical correlations, among the states of their local subsystems. However, such an assumption breaks down in quantum systems. The quantum system can exhibit both kinematic and measurement-induced nonlocality. The many particle hamiltonian expression one can work out using recursive function like approach may not obey the basic assumption used in denotational semantics, namely, a composite system is composed of parts and the state of whole is definable in terms of the states of its parts. There are quantum correlation between different subsystems due to the superposition principle. In fact, the whole is more than its parts, even if we assume there is no interaction among the subsystems. Therefore, we cannot partition a quantum system into its parts as is done in computer science and synthesize the system using its parts; this is called non-separability. Hence the structural programmability, and the scalability of the system is lost.

6 Time, Proof, and Complexity of Quantum Programs

Reasoning about programs (verification, proving correctness and termination) is an essential requirement in program design. Such schemes are usually based on the transfinite induction principle that hinges on the theory of nonnegative integers [19, 20, 22]. Also while reasoning about the complexity of computation, as well as, time dependent and real time programs, time is represented as a discrete set of points mappable to the positive integers or to a set of intervals and temporal logic is used [19]. Such a notion of time satisfies the well-founded relationship to be described in the next subsection. Proving properties of programs and termination rest heavily on the well-founded properties, and chain reasoning method.

Despite numerous attempts to revise this concept over many years, it is well accepted in Physics that time is a real number, continuous and is infinitely divisible. (In fact, time can have a fractal dimension too). Many of the physical theories rest on this fundamental assumption. The notion of time complexity in computation depends on whether the time is continuous or discrete, since this notion distinguishes the continuous time dynamical evolution (as in analog computation) from the discrete digital computation. This is because time appears as an intrinsic parameter in the process of quantum (analog) evolution. For example, under suitable conditions, it may be possible to use an algebraic transformation or rescaling of the time parameter of a differential equation to rearrange it so that that the processes which take infinite time can be executed in a finite time in analog machines. However, analog computing is not effective in the sense of Church-Turing thesis [3,4,11,13]. Thus the notion of time complexity is incomparable between the analog and the digital set- up. As an example, consider the mapping: t := (t-1)/t. This would transform the time parameter from the range [1, ∞] to [0,1]. This is called Zeno contraction or squeezing (Section 6.5). Such types of contractions, dilatations or scale changes abound in Physics. Thus the computability logic and physics widely differ in defining the structure of time.

6.1 Well-Founded Sets

A binary relation $<$ is *well-founded* over a class of objects, if it satisfies the no-decreasing condition; that is, there is no infinite sequence of objects decreasing with respect to that relation. In other words, there are no sequences {x(0), x(1), x(2), x(3),....} of objects such that: $x(0) > x(1) > x(2) > x(3) > \ldots$ where $>$ is the inverse binary relation of $<$. For a detailed study of well founded sets and relations, and related temporal logic, see Manna and Waldinger [22], Kroger [19]. Formally a well-founded relation (WFR) is defined over a set of objects (obj) thus: There are no infinite sequences {t (0),t(1),t(2),...} of objects t(u) such that: for all integers u object t(u) and for all integers u: $t(u) > t(u+1)$.

A set in which the elements are related through a well-founded relation is called a *well-founded set.*

6.2 Properties of Well-Founded Relation (WFR)

Irreflexive: That is there are no objects such that $x < x$.
Asymmetric: For all object x,y: [If $x < y$ then not ($y < x$)] must be valid.

Not necessarily transitive: While WFR may be transitive, it is not necessary. **Minimal Element:** If A is any nonempty subset of a wellfounded set, there is an element in A such that $x < y$ for all y in A. In other words, every nonempty subset of a wellfounded set contains a minimal element.

In computing logic time is quantized and is a well-founded set satisfying the properties – irreflexivity and asymmetry. Hence, we have a strong sense that time starts with a minimal element, the experienced time can be measured by units, and it has an arrow due to the asymmetry property. In Physics (both classical and quantum), time is not a well founded set. Time is a computable real number which is infinitely divisible [25,26]. Also calculus (continuity, differentiability and integrability) of real numbers plays a fundamental role in the physics of dynamical systems. Time arrow has not yet been established and time is symmetric [26].

6.3 Transfinite Induction Principle

An arbitrary well- founded relation over a set of objects (nonnegarive integers, tuples, trees) gives rise to the "Transfinite Induction Principle" stated below: If for some property P under every $z \varepsilon Z$, P(z) (i.e. z has the property P) x can be proved under the assumption that p(z') holds for every $z' < z$ then P(z) holds for all $z \varepsilon Z$. This principle plays a very crucial in the deductive foundations of computer programming to prove correctness and termination of programs not only over the integers, but also over other data structures -such as tuples and trees.

6.4 Consequences of Infinite Divisibility of Time

1. Since time is real and not a well-founded set, the transfinite induction principle (that permits inductive inference) fails. Therefore, quantum programs are not provable for correctness and termination using the classical generalised induction principle for nonnegative integers, tuples or trees.

2. Since time is not discrete, it has to be represented as computable real. That is there exists a computable sequence of rationals which converge effectively to a given value t. Thus we need to compute "bit by bit" the value of time that converges effectively within a radius of convergence.

3. Since time is infinitely divisible, the complexity of quantum computing becomes incomparable to the classical complexity results.

4. The recursive function over integers consists of the function algebra, consisting of some basic functions and closed under certain operations [3,11,13,17] : {ZERO, SUCCESSOR, PREDECESSOR, PROJECTION, COMPOSITION, PRIMITIVE RECURSION, BOUNDED MINIMALIZATION}. This can be generalised [3] to the primitive real class with:{ZERO, SUCCESSOR, PREDECESSOR, PROJECTION, COMPOSITION, INTEGRATION, ZEROFIND-ING on REAL} permitting us to define periodic functions like Sin, π, e. Also similar to the case of primitive recursive functions over integers that are not closed under minimalization, we also have the corresponding situation arising for integration. An important difference in real recursion is the replacement of primitive recursion by integration. Thus integrability becomes an important issue in dealing with complexity aspects over reals and complex numbers [40–43].

5. To reason about quantum programs, we need to develop the theory of recursive functions over the reals and an associated theory of programmability that can bridge with the theories of computational complexity, dynamical systems (classical and quantum), and numerical analysis.

6. Since time is real, we can convert an infinite time computation to a finite time by using Zeno squeezing oracle to be described in the next subsection. In fact time-line can be cut through to produce fractal time. This can result in statistically self-similar evolution [5,25,27,28,29]. In such a case, space and time should be appropriately defined for evaluating measures of complexity.

6.5 Zeno Squeezing Oracle

Turing defines an oracle [28] thus: An oracle is an agent or a "blackbox", which upon being consulted, supplies the true(correct) answers about mathematical or algorithmic or physical entities. Oracles though unrealizable in any universal computer, are useful to achieve the limit of an infinite computation. To conceive of such an oracle two time scales are introduced as below:

(i) an intrinsic time scale of the process of computation which tends to infinity and

(ii) an extrinsic time scale that remains finite for an external observer.

By using two such time scales, it is possible to produce within a finite proper time an output that would otherwise take an infinite time. This is called Zeno squeezing, since the infinite intrinsic time is squeezed to a finite extrinsic time limit. For example consider a time scale t (i) belonging to the set of integers:

$t(0) = 0; t(1)=1,....,t(k)= k,,.....$

Consider a scale T with $T(0)= t(0)= 0$; $T(i > 0) = [k^{t(i)} -1] /(k-1)$ with $0 < k< 1$.

In the limit when t (i) tends to ∞, the proper time $T(\infty) = 1/(1-k)$ remains finite.

A common feature of the Zeno oracle is that the valuation assigned to the intrinsic time t decreases with its increasing value, reaching a finite extrinsic time limit when the intrinsic time is infinity. Thus time flow is non-uniform with a higher valuation for closer events and a lower valuation for events into a far future, much like the visual valuation of the size of objects as spatial distance varies from near to far. This is analogous to the non-Archimedian valuation used in p-adic system [7]. Perhaps, quantum systems may offer the possibility of devising oracles to squeeze time**.** This can result in statistically self- similar evolution locally at every point of time and scaling invariance analogous to Brownian motion and fractional Brownian motion.

7 Concluding Remarks

We explained some important reasons why quantum algorithms are difficult to design and implement. In particular, we explained the difficulty in seeking an appropriate structural transformation from a given problem domain to the non-dissipative quantum domain,failure of programmability and scalability. Since time in Physics is a real number quantum programs are not provable using transfinite induction principle.

References

1. Ban, M: Information and entropy in Quantum measurement processes, International J Theoretical physics, **37** (1998) 2491–2537
2. Braun,D: Dissipative Quantum chaos and Decoherence,Springer, New York (2001)
3. Campagnolo, M.L.: The Complexity of real recursive functions,in Lecture Notes in Computer Science, Vol.2509: Springer Verlag , New York (2002) 1–14.
4. Deutsch,D.: Quantum Theory, the Church Turing principle and the universal quantum computer, Proc. Roy. soc. London, **A 400** (1985) 97–117.
5. Feder,J: Fractals, Plenum Press, New York,(1988)
6. Feynman,R: Quantum Mechanical Computers, Foundations of Physics,**16** (1986)507–531.
7. Gregory ,R.T and Krishnamurthy, E.V: Methods and Applications of Error-free Computation, Springer Verlag, New York (1984)
8. Gruska ,J: Quantum Computing, McGraw Hill, London (1999)
9. Hey,A.J.C., and. Allen, R.W: Feynman Lectures on Computation, Addison Wesley, New York, (1996)
10. Kitaev, A., Yu, Shen, A.H., and Vyalyi, M.N: Classical and Quantum Computation, American Mathematical Society, New York (2002)
11. Krishnamurthy, E.V.: Introductory Theory of Computer Science, Springer Verlag, New York, (1984)
12. Krishnamurthy, E.V.: Error-free Polynomial Matrix Computations, Springer Verlag, New York, (1985)
13. Krishnamurthy, E.V.: Parallel Processing, Addison Wesley, Reading, Mass(1990)
14. Krishnamurthy E.V.: Complexity Issues in parallel and distributed Computing, Chapter 4, in Parallel and Distributed Computing Handbook, Ed. A. Zomaya, McGraw Hill, New York, (1996) 89–146
15. Krishnamurthy, E.V.: Computational power of quantum machines, quantum grammars, and feasible computation, International J. Modern Physics. **C 9** (1998) 213–241.
16. Krishnamurthy, E.V: Integrability, Entropy and quantum computation, International J. Modern Physics. **C10**(1999) 1205–1228.
17. Krishnamurthy, E.V and Krishnamurthy, V:, Quantum Field theory and Computational Paradigms, International J Modern Physics. **C12** (2001),1179–1205.
18. Krishnamurthy, E.V: Mapping, Programmability and Scalability of problems for quantum speed-up, International J. Modern Physics, in press.
19. Kroger, F: Temporal Logic of Programs, Springer Verlag, New York (1987)
20. Lamport, L: Time, Clocks, and the ordering of events in a distributed system, Comm. ACM,.**21** (1978) 558–565.
21. Landauer, R: Is Quantum mechanics useful? Phil.Trans.Roy.Soc., Lond, **A.353** (1995) 367–376
22. Manna, Z., and Waldinger,R.:The deductive foundations of Computer programming, Addison Wesley, Reading, (1993)
23. Nielsen, M.A., and Chuang, I.L.: Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, (2000)
24. Niwa, J., Matsumoto. K.,and Imai, H.: General purpose Parallel simulator for Quantum computing, Lecture Notes in Computer Science, Vol.2509,Springer Verlag, New York, (2002) 230–251.
25. Nottale, L.: Fractal Space-Time and Microphysics, World Scientific, Singapore (1993)
26. Park, D.: Time in Quantum mechanics, in Fundamental questions in quantum mechanics, Ed. L.M. Roth and A. Inomata, Gordon and Breach Science Publishers, London (1986)
27. Peitgen, H., Jurgens, H., and Saupe, D.: Fractals for the class room, Part 1, Springer Verlag, New York, (1991)
28. Svozil, K.: Randomness & Undecidability in Physics,World Scientific, Singapore(1993)
29. Wolfram, S.: A New kind of Science, Wolfram Media Inc., Champaign, Ill,(2002).

A Configuration Algorithm for Provisioning of Cell-Based MPLS Network

Daniel Won-Kyu Hong[1], Choong Seon Hong[2], and Dong-Sik Yun[1]

[1] Operations Support System Lab., KT
463-1 Jeonmin-Dong Yuseong-Gu, Daejeon 305-811 Korea
{wkhong,dsyun}@kt.co.kr
[2] School of Electronics and Information, Kyung Hee University
1 Seocheon-Ri Kiheung, Yongin, Kyungki-Do 449-701 Korea
cshong@khu.ac.kr

Abstract. This paper proposes a configuration algorithm of multipoint-to-point connection for facilitating VC merge concept in an MPLS over ATM network. This configuration solves two major problems: determining the most reasonable candidate ATM switch served as an ATM-LSR under the complex ATM network topology; and providing the most reasonable multipoint-to-point connections with regards to network resource utilization. This algorithm is therefore subsequently composed of two procedures: ordering and backtracking. The ordering algorithm is for determining the orders of link and node taking the routing metrics of available bandwidth delay and hop count into account. The backtracking algorithm is to determine the optimal multipoint-to-point route based on the order information allocated by ordering algorithm. We evaluate the proposed algorithms under the operational network environment of KT.

1 Introduction

The Internet Engineering Task Force community recently established a working group on multi-protocol label switching (MPLS) aimed at standardizing the label-swapping paradigm that integrate layer 3 routing with layer 2 switching [1]. At the outset, MPLS is designed to be scalable and capable of being deployed in the core network with a large number of nodes. VC merging is an attractive feature that allows MPLS to scale to a very large network. Although layer 2 switching in MPLS can be applied in principle, the use of asynchronous transfer mode (ATM) switches in the backbone network is considered the most attractive solution as ATM hardware switches have been extensively studied and are widely available in various architectures [4,5,6,8]. This paper defines layer 2 switching that uses the ATM technology and refer to the associated label switched router (LSR) as an ATM-LSR. In this case, each IP packet may be segmented to multiple 53-byte cells before being switched. Traditionally, ATM adaptation layer 5 (AAL5) has been used as the encapsulation method in data communications since it is simple, efficient, and possesses a powerful error-detection mechanism.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 789–798, 2003.
© Springer-Verlag Berlin Heidelberg 2003

For the ATM-LSR to forward incoming cells to the correct output, the IP-route information needs to be mapped to the ATM labels that are kept in the virtual path identifier (VPI) or/and virtual channel identifier (VCI) fields. The relevant route information that is stored semi-permanently in the IP-routing table contains the tuple {ingress, next-hop router}.

Several methods for mapping the route information to ATM label exist. In the simplest method, each ingress-ingress pair is mapped to a unique VCI at a switch. The non-VC merge scheme, as this method is called, allows receivers to easily reassemble cells into respective packets as the VCI can be used to distinguish between senders. However, for n ingresses and ingresses, each AT-LSR is potentially required to manage $O(n^2)$ VC labels for full-meshed connectivity. This method is not scaleable for large networks. On the other hand, in the VP merge scheme the VP labels of cells intended for the same ingress are translated to the same outgoing VP value, thereby reducing VP consumption downstream. The VCI identifies the sender for each VP so that the receiver can reconstruct packets, even though cells from different packets are allowed to interleave. Although it can remarkably reduce the number of label entries compared with the non-VC merge scheme, VP merge is limited because the VP space is limited to only 4096 entries at the network-to-network interface (NNI). The VC merge scheme does not have the space-constraint problem as in VP merge. However, it cannot distinguish the sender at the output port of ATM-LSR. Therefore, cells belonging to different packets for the same ingress cannot interleave with each other or the receiver will not be able to reassemble the packets. Currently there have been several attempts to solve the problem of indistinguishable senders of VC merge and to propose the ATM-LSR architecture to accommodate VC merge scheme [4,5,6,7,8]. Using the results of these studies [4,5,6,7,8], scalable MPLS network using VC merge scheme in MPLS over ATM network can be configured. However, our proposal gives less emphasis to deployment of MPLS over ATM network. In newly-deployed ATM-LSR under the pre-provisioned ATM network in which some network resources are already allocated to ATM service provision, the most reasonable merge point where the existing ATM switch should be replaced with ATM-LSR to provide efficient MPLS service using VC merge concept must be determined.

This paper proposes an innovative multipoint-to-point algorithm that is able to identify the most reasonable VC merge point under the complex ATM network. It is composed of two subsequent algorithms: ordering and backtracking. The ordering algorithm trims the unreasonable nodes or links and assigns the orders of nodes and links with routing metrics of hop count and available bandwidth. The performance of ordering is unrelated to the number of connecting points in multipoint-to-point connection, as will be demonstrated by performance comparable to the original topology. The backtracking algorithm finds the most reasonable merge point based on the order information of nodes and links allocated by the ordering algorithm. Backtracking performs the connection admission control (CAC) and resource reservation. In addition, some LSRs can be added to the ATM-LSR according to the network expansion. This paper also

proposes the addition of LSRs mechanisms, also using ordering and backtracking algorithms, to the existing multipoint-to-point connection to add new edge LSRs to PoPs. The proposed multipoint-to-point algorithm finds the optimal merge point under the complex ATM network composed of 400 ATM switches and 1,078 links connecting per node. In addition, the performance of ordering and backtracking algorithms under the above network topology is measured.

2 Determination of an Optimal Merge Point

MPLS networks can use traditional ATM equipment to migrate and introduce MPLS to an existing ATM network. ATM network providers may prefer to keep an existing ATM infrastructure. On the other hand, building a new MPLS infrastructure allows customers to access both the MPLS network and services and along with the old ATM network, even from a single access link. There are several considerations in choosing ATM LSRs, such as type of trunks, number of trunks, number of connections, whether or not VC merge is required, and requirements for redundancy and reliability. This paper focuses only on the support of VC merge facility while maintaining traditional ATM services. Fig. 1 shows a simplified network configuration for introducing MPLS service to an existing ATM network. There are three ingress label switched routers (LSR1, LSR2, and LSR3) and one egress LSR (LSR4).

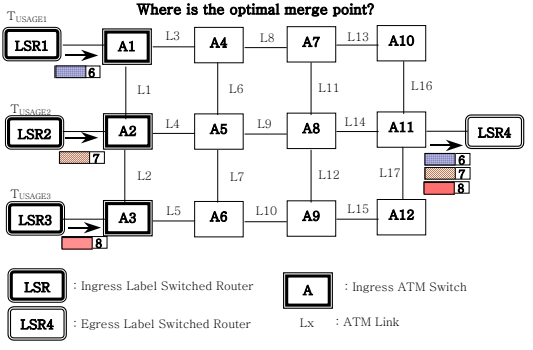


Fig. 1. Determination of optimal VC merge point

In order to accommodate MPLS service using the VC merge scheme, the most efficient location should be determined. There are three intermediate ingress ATM switches and one intermediate ingress ATM switch. In such an environment, a novel multipoint-to-point algorithm is needed to find the most reasonable merge point, particularly with regards to the maximization of network resource utilization and end-to-end transit delay. Determining the optimal merge point is vital in introducing MPLS service to a traditional ATM network as the ATM

switch identified as the optimal merge point should be replaced with the ATM-LSR having MPLS control functions. To find the most reasonable merge point under the complex ATM network topology, taking into account network resource utilization and end-to-end delay, we propose an innovative multipoint-to-point algorithm composed of ordering and backtracking algorithms.

2.1 Ordering Algorithm

The ordering algorithm is for determining the orders of link and node taking the routing metrics of available bandwidth delay and hop count into account. Network topology is represented by Graph, $G(V, E)$, where Vertex (V)denotes a set of nodes and Edge (E) denotes a set of links among nodes. Vertex contains three kinds of information: order (O), visit flag $(VISIT)$, and availability (AVA); while edge maintains three kinds of information: order (O), available bandwidth (B_{AVA}) and availability (AVA). Availability represents the possibility of cell transfer capability of each vertex or edge, which depends mostly on fault status or performance degradation of nodes or links.

```
1. G(V,E), where V : Vertex,   E : Edge
2. V = {O,VISIT},  E = { O,B_AVA}   //   O : Order , B_AVA : Available Bandwidth
3. B_REQ : Requested Bandwidth
4. E_ADJ : Adjacent Vertex of E
5. for all E(O) ← ∝;                    // ∝ : Infinite
6. V_VISIT(O,VISIT) ← (0,YES);          // V_VISIT : Source Vertex
7. Procedure Ordering(V_VISIT, B_REQ) {
8.        for all E ∈ V_VISIT {
9.              E(O) ← V_VISIT(O)+ 1;
10.             V_NEIGH ← E_ADJ;          // Adjacent Vertex of E
11.             if(V_VISIT( VISIT) == NO && V_NEIGH(O) < E(O) && E(B_AVA) ≥ B_REQ) {
12.                   V_NEIGH(O, VISIT) ← (E(O), YES);
13.                   if(V_NEIGH(O) != ∝)
14.                         Ordering(V_NEIGH, B_REQ); // Recursive Call
15.             } else {
16.                   E(O) ← ∝;
17.                   V_NEIGH(O) ← ∝;
18.             }
19.       } // end of for
20.} // end of Ordering
```

Fig. 2. Ordering algorithm

Fig. 2 illustrates the ordering algorithm. Fig. 3 shows the ordering procedure for determination of orders of nodes and links from ingress nodes of A and B to egress node E with the requested bandwidth of 7Mbps. In the ordering procedure, the ingress and egress vertexes are reversed. In Fig. 3, there are two ingress vertexes of A and B and one egress vertex E. Traversing the network topology to determine the orders of vertexes and the orders of edges, we define two terminologies of the visiting vertex (V_{VISIT}) and the neighbor vertex (V_{NEIGH}). The Vvisit is an active vertex that allocates the proper orders of all neighbor edges (V_{NEIGH}) connected to it and sets its visit flag to YES. On the other hand, V_{NEIGH} is a passive vertex in that its order is determined by the order of edge

connected to it and its visit flag is not changed to YES until it can be V_{VISIT}. For example, when we start to traverse network topology from ingress vertex E, it can be V_{VISIT}, and the vertexes of C, and D connected to the edges (L1 and L3) of vertex E can be $V_{NEIGH}s$ in Fig. 3.

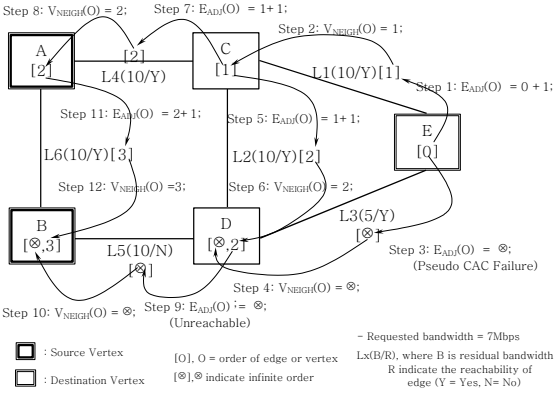


Fig. 3. Ordering procedure

The ordering algorithm trims the unusable network resource from whole network topology in terms of lack of resource and abnormalities and determines the appropriate orders (O) of links and nodes. In order to assign proper order to each edges or vertexes, initially, we assign infinite (∞) to all orders of vertexes and edges and assign NO to visit flags of all vertexes, as show in line 5 of Fig. 2 ($for\ all\ E(O) \leftarrow \infty$;). We assign zero to the order of egress vertex E that can serve as the first V_{VISIT}, as shown in line 6 of Fig. 2 ($V_{VISIT}(O, VISIT) \leftarrow (0, YES)$;). From here on, we traverse the network topology with the ordering algorithm of Fig. 2 until the visit flag of all vertexes is set to YES and the order of all edges and vertexes according to the hop count, availability and available bandwidth is assigned with the following rules:

The determination of the next V_{VISIT} is done in line with the depth first search manner. One important scheme can determine the efficient branch point for multipoint-to-point route. It is normal that one of the V_{NEIGH} can serve as the next V_{VISIT} in terms of the depth-first search scheme. We hereby define one exception where the V_{NEIGH} cannot serve as the next V_{VISIT} if its order is infinite (∞). For example, the next V_{VISIT} vertex of E is not D but C because the order of edge L3 connected to vertex E is infinite (∞) in the case of ordering algorithm of Fig. 3.

In the process of traversing the network topology, the order is assigned according to the simple rule that the order of edges connected to the V_{VISIT} is assigned by the order of the V_{VISIT} plus one ($E(O) \leftarrow V_{VISIT}(O) + 1$). If the order of any edge is larger than the order of V_{VISIT} plus one, it is changed to

the order of V_{VISIT} plus one. If the order of any edge is less than or equal to the order of V_{VISIT} plus one, it will not be changed. For example, the V_{VISIT} is E, the links connected to the vertex E are L1 and L3. The requested bandwidth (B_{REQ}) is 7Mbps. The order of the edges connected to the vertex E is replaced with one (the order of vertex E plus one) because the order of the first V_{VISIT} E is zero. If any link can accommodate B_{REQ} and its order is not infinite, its order will be changed to one such as L3. However, if any link cannot accommodate B_{REQ}, its order is changed to infinite (∞), having nothing to do with the order of V_{VISIT}. If the order of any link such as L1 is greater than or equal to the order of the V_{VISIT} plus one, it will not be changed. Fig. 3 shows the sample network topology and its results of order assignment in accordance with the ordering algorithm. There are three ingress vertexes of A1 and A2 and one egress vertex A11 with the requested bandwidth of 20Mbps.

2.2 Backtracking Algorithm

Backtracking aims to determine the optimal multipoint-to-point route based on the order information allocated in the process of order assignment. In multipoint-to-point route selection, the ingress and regress vertexes are not reversed when compared with the ordering procedure. In addition, the CAC on the selected edges is done in multipoint-to-point route selection procedure. Fig. 4 shows the multipoint-to-point route selection procedure based on the order assignment result of Fig. 3.

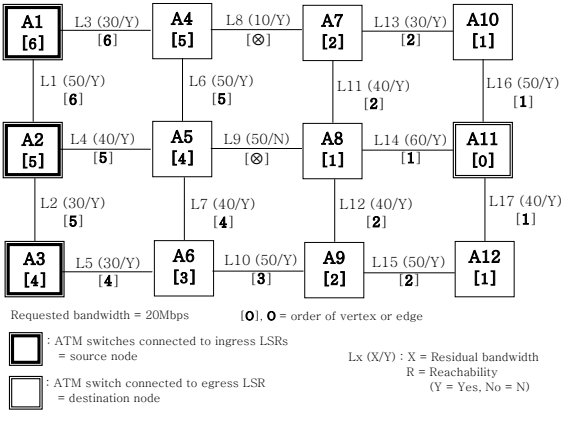


Fig. 4. Results of order assignment

There are three ingress vertexes of A1, A2 and A3 and one egress vertex A11 from the perspective of backtracking procedure, as shown in Fig. 5.

Backtracking algorithm is as follows:

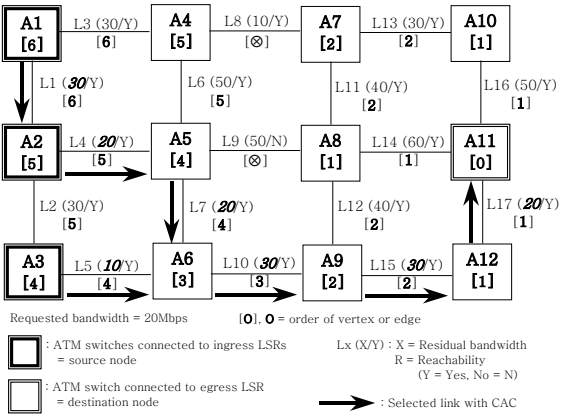


Fig. 5. Backtacking algorithm and procedure

- When it traverses the network topology from the ingress to egress vertexes, it selects the edge having the least order. For example, ingress vertex A3 selects the edge L5 as an optimal edge between two edges of L2 and L5 because the order of L2 is less than that of L5.
- If there are two or more edges with the same order, the edge having the largest residual bandwidth is selected. For example, ingress vertex A1 selects L1 as an optimal edge between the two edge of L1 and L3 because the residual bandwidth (30Mpbs) of L1 is larger than that (10Mps) of L3 after CAC.

Thus, the CAC on the selected edges is carried out in multipoint-to-point backtracking. The optimal route between ingress vertex A1 and egress vertex A11 can be {A1-L1-A2-L4-A5-L7-A6-L10-A9-L15-A12-L17-A11}. Furthermore, The optimal route between A2 and A11 can be {A2-L4-A5-L7-A6-L10-A9-L15-A12-L17-A11}. In addition, the optimal route between A3 and A11 can be {A3-L5-A6-L10-A9-L15-A12-L17-A11}.

2.3 Determination of Optimal VC-Merge Point

With the ordering and backtracking algorithms, we can determine the optimal VC-merge point under the complex network topology. The VC-merge point implies that an ATM switch identified as an optimal VC-merge point should be replace with the ATM-LSR having MPLS VC-merge capability. As a result of backtracking, there are duplicated routes among multipoint-to-point routes. In the case of Fig. 6, the duplicated route among three ingress vertexes of A1, A2 and A3 and egress vertex A11 can be {A6-L10-A9-L15-A12-L17-A11}. Hence, the first duplicated vertex can be the optimal VC-merge point, as shown in Fig. 6. Therefore, we can replace the ATM switch corresponding to A6 with the ATM-LSR having VC-merge capability. Of course, there is another candidate

VC-merge point of A2. However, it can only merge MPLS traffics from A1 and A2. We need to identify the optimal VC-merge point where all MPLS traffics from A1, A2 and A3 is simultaneously merged. It can be the first duplicate vertex among the found three routes in terms of multipoint-to-point route selection, as shown in Fig. 6. The ingress vertexes of A1, A2 and A3 are connected to ingress ATM-LSRs and egress vertex A11 is connected to LSR in IP-based MPLS network. In order to configure ATM based MPLS network, we establish ATM virtual path connection corresponding to an Label Switched Path (LSP) between ingress ATM-LSR and egress ATM-LSR having VC-merge capability.

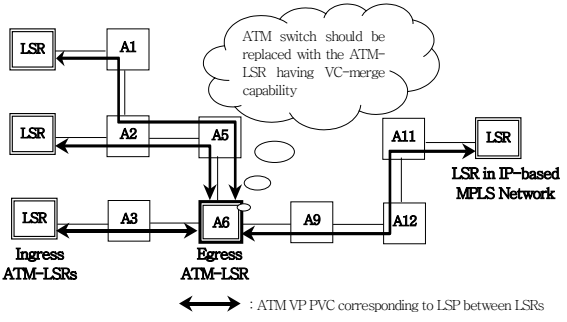


Fig. 6. Optimal VC-merge point

3 Performance Analysis

We implement the multipoint-to-point routing algorithm as a part of network management system (NMS) in accordance with the functional layering concepts of TMN, as applied to the High Speed Information Network (HSIN) of KT. We evaluate the proposed multipoint-to-point algorithm under the KT's HSIN composed of 14 core nodes, 38 edge nodes and 239 access nodes. All the core nodes are connected in nearly full mesh with STM-4, while the edge and access nodes are connected to the core nodes in star topology with STM-1. There are 291 nodes and 908 links, comprising a significantly large network. We measured the performances of ordering algorithm, backtracking algorithm and adding branch algorithm using the SUN E65000 equipped with 8CPUs and 2G main memory. We did not use any simulation tool to measure performance because of some differences between performance measured using any simulation tool and performance measured using the real environment. As a national network carrier, we need real performance measured using a real service provider to guarantee customer's satisfaction. Fig. 7 presents the performance of multipoint-to-point route selection up to one hundred ingresses. There are two major performance metrics: ordering performance and backtracking performance. In ten ingresses, it

took 2.13 seconds to find the optimal multipoint-to-point route. In the worst case among fifty-one ingresses, it took 4.48 seconds to find the multipoint-to-point route. The ordering performance is constant under the same network topology regardless of the number of ingresses. However, backtracking performance including CAC and residual bandwidth manipulation gradually deteriorates until the number of ingresses reaches fifty-one. However, the backtracking performance gradually improves in proportion to the number of ingresses when it is over fifty-one, as shown in Fig. 7.

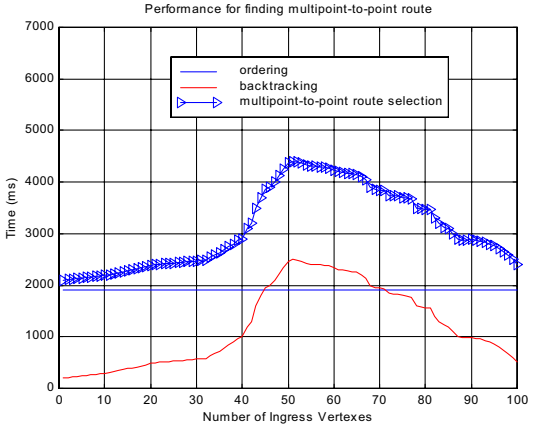


Fig. 7. Performance of multipoint-to-point routing algorithm

We analyze performance under the constraint of one hundred ingresses as the maximum number for purposes of validating scalability. The ordering algorithm manifested a constant performance of 1.90 seconds under the same network topology. However, the backtracking algorithm resulted in higher performance in reverse proportion to the number of ingresses when it is larger than fifty-one. It is mainly because the duplication ratio of intermediate edges and vertexes are increased in proportion to the number of ingress. Our performance evaluation revealed that the proposed ordering and backtracking algorithms for finding the optimal merge point for utilizing ATM VC-merge concept under the MPLS over ATM network environment yielded good scalability.

4 Concluding Remarks

This paper proposed the multipoint-to-point routing algorithm to find the optimal merge point for utilizing ATM VC-merge concept under the MPLS over ATM network. It could identify the ATM switch that should be replaced with ATM-LSR having VC-merge capability. It is composed of ordering algorithm for

determining the orders of link and node taking the routing metrics of available bandwidth delay and hop count into account and backtracking algorithm for selecting optimal multipoint-to-point routes based on the order information of vertexes and edges. In addition, this paper proposed the algorithm for adding extra branches to the existing multipoint-to-point route. This paper showed that the proposed ordering and backtracking algorithms could be applied to large-scale ATM network by means of performance analysis of the multipoint-to-point route selection. It could confirm the multipoint-to-point service provision within 2.13 seconds when the ingress is below ten, with residual bandwidth manipulation of CAC, and optimal route selection included. The ordering algorithm manifested a constant performance of 1.90 seconds under the same network topology. However, the backtracking algorithm resulted in higher performance in reverse proportion to the number of ingress when it is larger than fifty-one. It is mainly because the duplication ratio of intermediate edges and vertexes are increased in proportion to the number of ingress. Our performance evaluation revealed that the proposed ordering algorithm and backtracking algorithms for providing optimal multipoint-to-point service yielded good scalability. It is the most prominent strength of the proposed adding branch algorithm. With the proposed multipoint-to-point algorithm, we can find the most optimal merge point under the complex ATM network for utilizing the ATM VC merge concept under the MPLS over ATM network.

References

1. Rosen, E., Viswanathan, A. and R. Callon, "Multi-Protocol Label Switching Architecture", RFC 3031, January 2001.
2. B. Davie, J. Lawrence, K. McCloghrie, et al, "MPLS using LDP and ATM VC Switching," RFC 3035, January 2001.
3. http://www.cisco.com/warp/public/121/mpls_vcmerge.html
4. Hungkei Keith Chow and Alberto Leon-Garcia, "VC-Merge Capable Scheduler Design," IEEE Proceedings of ATM Workshop, pp. 153–160, 1999.
5. R. Venkateswaran, C.S. Taghvendra, X.Chen and V.P. Kumar, "Support for Multiway Communications in ATM Networks," IEEE Proceedings of ATM Workshop, pp. 339–348, 1998.
6. Peiganf Zhou and Oliver W.W. Yang, "Reducing Buffer Requirement for VC_Merge Capable ATM Switches, " Proceedings of IEEE Global Telecommunications Conference, pp. 44–48, Globecom'99. 1999.
7. Indra Widjaja and Anwar I. Elwalid, "Performance Issues in VC-Merge Capable Switches for IP over ATM Networks," IEEE Journal on Selected Areas in Communications, VOL 17, No. 6., pp. 372–380, June 1999.
8. D.W.Lee, T.W. Lee, Y.C.Kim, D.J.Choi and Mike M.O. Lee, "Implementation of a VC-Merge Capable Crossbar Switch on MPLS over ATM," IEEE International Conference on ATM and High Speed Intelligent Internet Symposium, ICATM 23001, pp. 66–69, 2001.

Exact and Approximate Algorithms for Topological Design of Wide Area Networks with Non-simultaneous Single Commodity Flows

Andrzej Kasprzak

Wroclaw University of Technology, Chair of Systems and Computer Networks
50-370 Wroclaw, Poland
kasprzak@zssk.pwr.wroc.pl

Abstract. The problem is concerned with designing of wide area computer networks. This problem is very important from practical point of view because the network, when designed efficiently, can and does become the lifeblood of a business organization. The problem consists in selection of the non-simultaneous single commodity flow routes, channel capacities and the wide area network topology in order to maximize the value of the min-max flow subject to the budget constraint. The problem is NP-complete. Then, the branch and bound method is used to construct the exact algorithm. Several properties of the problem are formulated and proved. Computational results are reported.

1 Introduction

Data transmission in a wide area network (WAN) environment involves the design of a communication network. Several different formulations of the design problem can be found in the literature; generally, they correspond to different choices of criterion, of design variables and of constraints [1], [2], [3]. Usually delay, cost, throughput and network reliability are applied as criteria. For the purpose of this paper the throughput of the WAN represented by non-simultaneous single commodity flow requirements is selected as criterion. The design variables are following: topology, channel capacities and routing (i.e. flows). The techniques used for the solution of topological design problem depend on the nature of cost-capacity functions. The problem is considered for linear, concave and discrete cost-capacity functions [3]. Since the channel capacities can be chosen from discrete sequence defined by ITU-T (International Telecommunications Union – Telecommunications Sector) recommendations, then only discrete cost-capacity function is important from practical point of view.

Consider the network and maximal flows between each pair of its nodes. Of all the maximal flows in the network the one the value of which is minimal is called the min-max flow. The non-simultaneous single commodity flow requirements are understood here as the value of the min-max flow in the network.

In the paper, we consider the new problem which is concerned with selection of the capacities of channels and their geographical layout (i.e. network topology) so as to satisfy the investment budget constraint and maximize the min-max flow value. The

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 799–808, 2003.
© Springer-Verlag Berlin Heidelberg 2003

budget constraint limits the number of channels that may be included in the optimal network topology as well as the values of capacities of these channels chosen from the set of discrete values of capacities defined by ITU-T recommendations. The discrete cost-capacity function is considered. Also we assume that there are only half-duplex channels in the WAN. The considered problem can be formulated as follows.

Given:	node location, possible channels locations, channel capacity options and costs, budget of the WAN
Maximize:	value of min-max flow
over:	topology, channel capacities, routing (i.e. min-max flow)
subject to:	flow constraints, channel capacity constraints, budget constraint.

First, several cases of the network synthesis problem with non-simultaneous single commodity flow requirements were studied by Gomory and Hu [2], [4]. They study minimum cost network design problems with the linear cost-capacity function and without budget constraint. The formulation of the considered problem differs from the formulation of Gomory and Hu problem because: we assume the discrete costs-capacity function (Gomory and Hu assume only linear cost-capacity function), we take into account the budget constraint (Gomory and Hu problem is solved without the budget constraint) and we use other optimization criterion (value of min-max flow) than criteria used in [2], [4]. The main difference between the problem [2], [4] and the problem considered here is such that the problem [2], [4] belongs to the class of continuous problems and our problem belongs to the class of discrete problems. The formulation of the problem considered here is similar to formulations of topological design problems of WAN presented for example in the papers [1], [3].

2 Problem Formulation

Let p' be the number of channels which must be installed in the WAN and let m be the number of possible channels which may be used to build the WAN. The capacity of the i-th channel which must be installed is denoted by c_0^i. For each potential i-th channel there is the set $ZC^i = \left\{c_1^i, c_2^i, ..., c_{r(i)-1}^i\right\}$ of alternative capacities from among which exactly one must be chosen if the i-th channel was chosen to build the network. Let d_k^i be the cost of leasing capacity c_k^i. Let the network $S = \langle G; c, d \rangle$ be the model of the WAN. The network S is defined by a undirected multigraph $G = \langle N, L \rangle$, by a capacity function c and by a discrete cost function d; L denotes the set of arcs.

$N=\{1,..., n\}$ - the set of nodes; N corresponds to the set of nodes of the WAN,

$A=\{w^{m+1},..., w^p\}$ - the set of fixed undirected arcs; each arc from this set corresponds to the channel which must be installed in the WAN; $p=p'+m$.

$\overline{W}^i = \left\{w_1^i, ..., w_{r(i)-1}^i\right\}$ - the set of alternative undirected arcs; at most one arc from this set may be used to construct the network S. This set corresponds to the i-th channel which may be used to build the WAN. The number of all $\overline{W}^i$ is equal to m.

The functions c and d are defined as follows:

$c\left(w_j^i\right)=c_j^i$ and $d\left(w_j^i\right)=d_j^i$ for every $w_j^i \in \overline{W}^i$, i=1,..., m,

$c\left(w^i\right)=c_0^i$ and $d\left(w^i\right)=0$ for every $w^i \in A$,

where $c\left(w_j^i\right)$ and $d\left(w_j^i\right)$ are the capacity and the cost of the arc w_j^i, respectively.

All arcs from the set $\overline{W}^i$ connect the same pair of nodes. Each arc w_j^i corresponds to capacity c_j^i from the set ZC^i. If the arc w_j^i is chosen to construct the network S, then it means that the i-th channel with capacity c_j^i is used to build the WAN.

Let's call the arcs $w_{r(i)}^i$ for i=1,..., m the empty arcs; then $c\left(w_{r(i)}^i\right)=0$ and $d\left(w_{r(i)}^i\right)=0$. Then the set $W^i=\overline{W}^i \cup \left\{w_{r(i)}^i\right\}$ is the set of alternative arcs from among which exactly one arc must be used to construct the network S. If the arc $w_{r(i)}^i$ is chosen then the i-th channel is not used to build the WAN. Let the set W be the sum of all sets W^i. Thus, $L=W\cup A$. Let $X_r \subset W$ be the set of arcs such that X_r includes exactly one arc from each set W^i. X_r is called a selection. Let R be the set of all such selections. Each selection X_r generates the undirected graph $G_r=\langle N, A\cup X_r\rangle$ and the associated network $S_r=\langle G_r; c, d\rangle$. Therefore, the network S_r defines unique WAN because the selection X_r determines simultaneously potential channels which are used to build the WAN and the values of capacities for these channels.

Let $v_r(i,j)$ be the maximal flow value from source i to sink j in the network S_r. $v_r(i,j)$ is obtained solving the capacited maximal flow problem [2]. To find the maximal flows between all pairs of nodes of the network the Gomory and Hu algorithm [2] may be applied. It is easy to see that the flow between each pair of nodes in the network S_r corresponds to the packets flow in the WAN which topology and channel capacities are given by X_r.

Let $V(X_r)$ be the value of min-max flow in the network S_r, then

$$V(X_r)=\min_{i,j\in N} v_r(i,j). \tag{1}$$

Let DB be the budget of the WAN and let $d(X_r)$ be the building cost of S_r. Then, the considered topological design problem can be formulated as follows.

$$\max_{X_r} V(X_r) \tag{2}$$

subject to

$$X_r \in R \tag{3}$$

$$d(X_r)=\sum_{w\in X_r} d(w) \le DB\,. \tag{4}$$

3 Calculation Scheme

Assuming that $r(i)=2$ and $c\left(w_1^i\right)=1$ for $i=1,\ldots, m$, $A=\varnothing$, the problem (2-4) is resolved itself into the "two-connectivity" problem. Since the two-connectivity problem is NP-complete [5] then the problem (2-4) is also NP-complete as more general. Then, the branch and bound method can be used to construct the exact algorithm for solving the considered problem. The detailed description of the calculation scheme of the branch and bound method may be found in the literature.

Starting with the initial selection $X_1 \in R$ we generate a sequence of selections. The arc $w \in W$ is called normal if $w \in X_1$. The arc which does not belong to X_1 is called reverse. A replacement of any arc by other arc from W^i is called complementing.

The generation of a new selection involves the choice of a certain normal arc from X_r for complementing; it is called the branching rules. For each selection X_r we perform a testing operation (i.e. upper bound) to check the possibility of the generation of X_s with the greater value of min-max flow than already found.

If the new selection X_s is generated from X_r by complementing the normal arc by the reverse arc, then we constantly fix this reverse arc. It means that this reverse arc cannot be complemented by any other in every possible successor of X_s. So, for each selection X_r we constantly fix a set $F_r \subset X_r$. The arcs which do not belong to F_r are called free in X_r. If we want backtrack from X_1 then the algorithm terminates.

4 Testing Operation

The basic task of this operation is the computation of upper bound UB_r of value of min-max flow for every possible successor generated from X_r. Let $\overline{C}_r$ be the upper bound of the sum of capacities of all arcs in S_r and in all its successors – it may be calculated using the algorithm similar to one presented in [3].

For each network S_s we have $v_s(x,y) \le c_s(x,N)$ for every $x, y \in N, x \ne y$, where $c_s(x,N)$ is the sum of capacities of all arcs connected to node x in S_s. Summing over all nodes $x, y \in N, x \ne y$, we obtain

$$v_s(N,N) = \sum_{x=1}^{n} \sum_{\substack{y=1 \\ y \ne x}}^{n} v_s(x,y) \le (n-1) \sum_{x=1}^{n} c_s(x,N) = (n-1) c_s(N,N) \tag{5}$$

where $c_s(N,N)$ is double sum of capacities of all arcs in S_s. Since $V(X_s) \le v_s(x,y)$ for every $x, y \in N, x \ne y$, then after summing over all nodes x, y we obtain

$$n(n-1)V(X_s) \le v_s(N,N). \tag{6}$$

It follows from (5) and (6) that $V(X_s) \le c_s(N,N)/n$. Since $\overline{C}_r$ is the upper bound of sum of all arc capacities in S_r and its successor S_s then $c_s(N,N) \le 2\overline{C}_r$. Then the value of min-max flow $V(X_s)$ in every successor X_s of the X_r is bounded and

$$V(X_s) \le 2\overline{C}_r / n . \tag{7}$$

Then the right hand side of the inequality (7) is the upper bound of min-max flow value in every successor of X_r. The following corollary follows from (6) and (7).

Corollary 1. Let X_s be the successor of $X_r \in R$. The value $V(X_s)$ is maximal iff the maximal flows between every pair of nodes in S_s are equal to $2\overline{C}_r / n$.

Let $v_r^u(x,y)$ be the maximal flow in the network $S_r^u = \langle G_r^u; c, d \rangle$, where $G_r^u = \langle N, A \cup F_r \rangle$. The network S_r^u contains only arcs from the sets A and F_r which are not changed in every successor of S_r. Then, for the maximal flows $v_s(x,y)$ in every successor S_s of the network S_r the following condition is satisfied

$$v_s(x,y) \ge v_r^u(x,y) \text{ for every } x, y \in N, x \ne y . \tag{8}$$

Let us assume that there is $v_r^u(k,l)$ such that $v_r^u(k,l) > 2\overline{C}_r / n$ in S_r. It follows from the above assumption and from (8) that for each successor S_s of the network S_r the condition $v_s(k,l) = v_s(l,k) > 2\overline{C}_r / n$ is satisfied.

Then, it follows from the corollary 1 that there also exist maximal flows less than $2\overline{C}_r / n$ in the network S_s. In such case, the value $V(X_s)$ may be bounded using condition (6) only for $x, y \in N$ such that $x \ne y$ and $(x,y) \ne (k,l)$. Then, taking into account the conditions (5), (8) and that $c_s(N,N) \le 2\overline{C}_r$ we obtain

$$V(X_s) \le \frac{2(n-1)\overline{C}_r - \left(v_r^u(k,l) + v_r^u(l,k)\right)}{n(n-1)-2} . \tag{9}$$

Next, if there exists $v_r^u(a,b)$, $(a,b) \ne (k,l)$ greater than the right hand side of (9) then the above procedure should be done again. Based on (9) the following algorithm for calculation the upper bound UB_r may be formulated.

Step 1. Compute $\overline{C}_r$. Let $B = \{(x,y): x, y \in N, x \ne y\}$, $e = 2\overline{C}_r / n$, a=0 and b=0. Calculate all $v_r^u(x,y)$ in S_r^u.

Step 2. If there are the pairs (x,y) such that $v_r^u(x,y) > e$ in the set B then perform $a = a + v_r^u(x,y)$, $b = b+1$, $B = B - \{(x,y)\}$ for each pair (x,y) and go to step 3. Otherwise go to step 4.

Step 3. Calculate $e = (2(n-1)\overline{C}_r - a)/(n(n-1) - b)$ and go to step 2.

Step 4. Perform $UB_r = e$ and stop.

5 Branching Rules

The purpose of branching rules is to find the normal arc from X_r for complementing and generating a successor of X_r with the possibly greatest value of the min-max flow.

The arc $w_j^i = \langle x, y \rangle$ is adjacent to the cut $(X, \overline{X})$ in the network S_r if there exists $x \in X$ and $y \in \overline{X}$. The minimum capacity cut which is connected with the min-max flow is called the min-max cut. It follows from the *max-flow − min-cut theorem* [2] that if we want increase the maximal flow in the network then the minimum capacity cut must increase. Then, it is easy to observe that the following corollary is true.

Corollary 2. Let $X_r \in R$. If there exists a network S_s (successor of S_r) such that $V(X_s) > V(X_r)$ then the selection X_s contains at least one reverse arc $w_j^i \in W - X_r$ of the normal arc $w_k^i \in X_r$ adjacent to the min-max cut in S_r.

Let $E_r = X_r - F_r$ be the set of all normal arcs which may be complemented by reverse arcs in X_r, and let M_r be the set of all reverse arcs of normal arcs from the set E_r which are adjacent to min-max cut in S_r. Also let B be the subset of the set N, and let $z(x,B)$ be the number of node a such that $a \in B$ and $v_r(x,a) = \min_{b \in B} v_r(x,b)$.

Theorem 1. Let $X_r \in R$ and let $(Z, \overline{Z})$ be the min-max cut in S_r. If S_s is obtained from S_r by complementing the normal arc w_k^i by the reverse arc $w_j^i = \langle x, y \rangle \in M_r$ then $V(X_s) \geq V(X_r) + \Delta_{kj}^{ir}$, where $\Delta_{kj}^{ir} = \min\left(\Delta_{kj}^{ir,1}, c(w_j^i) - c(w_k^i), \Delta_{kj}^{ir,2}\right)$, $\Delta_{kj}^{ir,1} = v_r(z(x,Z),x) - V(X_r)$ and $\Delta_{kj}^{ir,2} = v_r(z(y,\overline{Z}),y) - V(X_r)$.

Proof. Let $e \in Z$, $t \in \overline{Z}$ and let $(Q, \overline{Q})$ be the minimal capacity cut separating nodes x and e in S_r (Fig. 1). Then $v_r(e,t) = V(X_r)$. After complementing w_k^i by w_j^i we can located "additional flow" $\Delta v_r(e,t)$ from e to t on paths from e to t which contain the arc w_j^i because it is the one way to increase the maximal flow $v_s(e,t)$ in S_s. Then, after complementing the value $v_s(e,t) - v_r(e,t)$ will be greater or equal to minimum of the three values: the value of additional flow from e to x, the difference between capacities of arcs w_j^i and w_k^i and the value of additional flow from y to t. Thus,

$$v_s(e,t) - v_r(e,t) \geq \min\left(\Delta v_r(e,x), c(w_j^i) - c(w_k^i), \Delta v_r(y,t)\right). \tag{10}$$

Assume that $\overline{Z} \subset \overline{Q}$. Otherwise we obtain the same results. Let $U = Z \cap Q$ and $D = Z \cap \overline{Q}$ (Fig. 1). It is easy to observe that $c(Z,\overline{Z}) = c(U,\overline{Z}) + c(D,\overline{Z})$. The additional flow from the node e to the node x is $\Delta v_r(e,x) = c(U,D) - c(U,\overline{Z})$. Because $(U,\overline{Z}) \subset (Q,\overline{Q})$ and $(D,\overline{Z}) \not\subset (Q,\overline{Q})$ then $c(U,\overline{Z}) \leq c(D,\overline{Z})$. Hence,

$$\Delta v_r(e,x) = \left(c(U,D)+c(U,\overline{Z})\right)-\left(c(U,\overline{Z})+c(D,\overline{Z})\right)+c(D,\overline{Z})-c(U,\overline{Z}) = \\ = v_r(e,x)-v_r(e,t)+c(D,\overline{Z})-c(U,\overline{Z}) \geq v_r(e,x)-v_r(e,t). \tag{11}$$

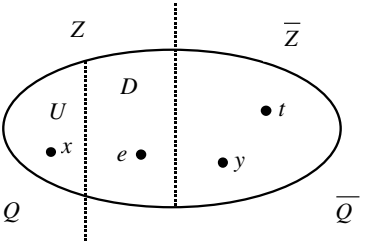


Fig. 1. The location of nodes and sets in the set of all nodes *N*

Taking into account that $v_r(e,t)=V(X_r)$ we obtain $\Delta v_r(e,x) \geq v_r(e,x)-V(X_r)$. The minimum value of the right hand side of the last inequality is for $e=z(x,Z)$, then

$$\Delta v_r(e,x) \geq v_r(z(x,Z),x)-V(X_r)=\Delta_{kj}^{ir,1}. \tag{12}$$

Analogically we obtain that

$$\Delta v_r(y,t) \geq v_r(z(y,\overline{Z}),y)-V(X_r)=\Delta_{kj}^{ir,2}. \tag{13}$$

Substituting (12) and (13) into (10) and taking into account that $v_r(e,t)=V(X_r)$ we obtain that $v_s(e,t)-V(X_r) \geq \min\left(\Delta_{kj}^{ir,1}, c(w_j^i)-c(w_k^i), \Delta_{kj}^{ir,2}\right)$. Then

$$v_s(e,t) \geq V(X_r)+\Delta_{kj}^{ir}. \tag{14}$$

To find the upper bound of $V(X_s)$ we should bounded the value of maximal flow between any pair of nodes a and b in S_s. There are three possible cases:

A. $a \in Z, b \in \overline{Z}$. We obtain, using (14) that $v_s(a,b) \geq V(X_r)+\Delta_{kj}^{ir}$,

B. $a,b \in Z$. Taking into account the triangle inequality [2] we obtain $v_s(a,b) \geq \min(v_s(a,t), v_s(b,t))$. Since values $v_s(a,t)$ and $v_s(b,t)$ are greater than right hand side of (14) then $v_s(a,b) \geq V(X_r)+\Delta_{kj}^{ir}$,

C. $a,b \in \overline{Z}$. Analogically to case B: $v_s(a,b) \geq \min(v_s(e,a), v_s(e,b)) \geq V(X_r)+\Delta_{kj}^{ir}$.

By definition the value of min-max flow is not greater than the value of the maximal flow between each pair of nodes, then $V(X_s) \geq V(X_r)+\Delta_{kj}^{ir}$. ■

The choice criterion for complementing is as follows: $\delta_{kj}^{ir} = \left|V(X_r)+\Delta_{kj}^{ir}-UB_r\right|$.

6 Approximate Algorithm

The exact algorithm involves to find the initial selection X_1 such that $X_1 \in R$ and for which the condition (4) is satisfied. Moreover, the initial selection X_1 should be the near-optimal solution of the problem (2-4). To find the feasible initial selection X_1 as above the following approximate algorithm is proposed.

Step 1. Assign maximum available capacities arcs to X_1 i.e. perform i=1 and $X_1^{(i)} = \{w_1^1, w_1^2, ..., w_1^m\}$.

Step 2. Compute $V(X_1^{(i)})$. If $d(X_1^{(i)}) \le DB$ then the feasible initial selection is found. Perform $X_1 = X_1^{(i)}$ and the algorithm terminates. Otherwise go to step 3.

Step 3. Choose the arcs $w_k^l = \langle x, y \rangle \in X_1^{(i)}$ and $w_{k+1}^l \in W^l - X_1^{(i)}$ for which the value $v_i(x,y) - c(w_k^l) + c(w_{k+1}^l) - V(X_1^{(i)})$ is maximal; $v_i(x,y)$ denotes the maximal flow from the node x to the node y in $S_1^{(i)} = \langle G_1^{(i)}; c, d \rangle$ where $G_1^{(i)} = \langle N, A \cup X_1^{(i)} \rangle$. Perform i=i+1 and $X_1^{(i)} = (X_1^{(i-1)} - \{w_k^l\}) \cup \{w_{k+1}^l\}$. Go to step 2.

Using the presented approximate algorithm we obtain the approximate (near optimal) solutions in a short computation time and using the presented exact algorithm we obtain the optimal solutions in a long computation time.

7 Computational Results

Extensive numerical experiments were performed with presented algorithms for many different network topologies. The experiments were conducted with two main purposes in mind: first, to examine the impact of various parameters on solutions (i.e. min-max flow) and second, to test the computational efficiency of the algorithms.

The typical dependence of optimal min-max flow value on budget is presented in the Fig. 2. It follows from results obtained for many considered networks that the optimal min-max flow value in terms of budget is the function belonging to following class:

$$V = \begin{cases} 0 & \text{for } 0 \le DB < D_1 \\ a \cdot DB + b & \text{for } D_1 \le DB \le D_2 \\ V_{\max} & \text{for } D_2 < DB \le DB_{\max} \end{cases} \tag{15}$$

where a, b, D_1 and D_2 are constant coefficients, $V_{\max}$ and $DB_{\max}$ are maximal feasible min-max flow value and maximal budget in the network respectively.

It is easy to see that $D_1 = -b/a$ and $D_2 = (V_{\max} - b)/a$. Then, to find the function (15) for some network it is enough to known only the values of a and b. To compute a and b the identification problem must be solved e.g. using the Last Squares Method.

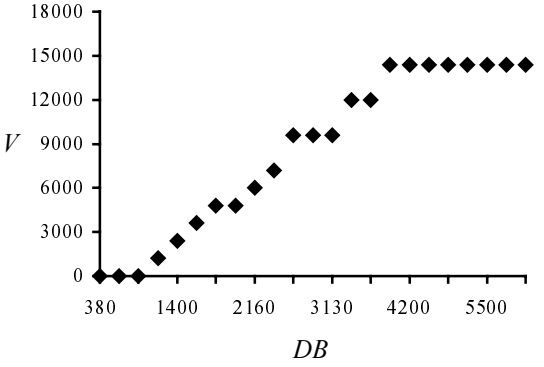


Fig. 2. The dependence of optimal min-max flow value V on budget DB

The quality of the approximate algorithm was examined. Let the distance between approximate and optimal solutions be denoted by $k = \left(\left|V(X_1) - V\right| / V\right) \cdot 100\%$. The value k shows how the results obtained using the approximate algorithm are worse than the optimal solutions. Let

$$K[a,b] = \frac{\text{number of solutions for which } k \in [a,b]}{\text{number of all solutions}} \cdot 100\%$$

denotes number of solutions obtained from approximate algorithm (in percentage) which are greater than optimal solutions more than $a\%$ and less than $b\%$. In the Fig. 3 the distance [0%] means that the approximate solutions obtained from the approximate algorithm are equal to the optimal solutions obtained from the exact algorithm, i. e. for such solutions the distance $k = 0$.

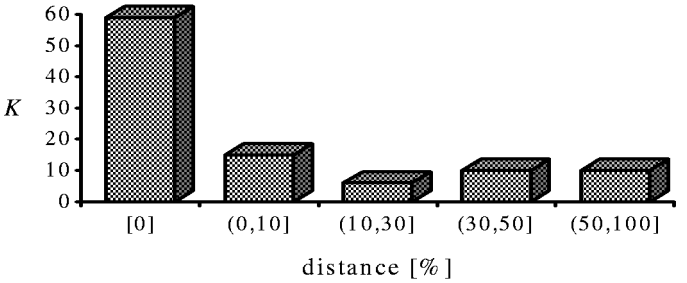


Fig. 3. The distance between the approximate and optimal solutions

Moreover, the numerical experiments have demonstrated that there exists dependence between numbers of iterations P of the exact algorithm and the

normalized budget $NDB = (DB / DB_{\max}) \cdot 100\%$ which was introduced to compare the results obtained for different network topologies. It follows from numerical experiments (Fig. 4) that the exact algorithm is especially effective from computation point of view for $NDB \in [0,20] \cup [40,100]$.

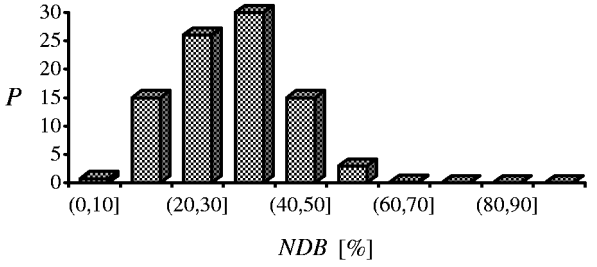


Fig. 4. The dependence of number of iteration *P* on normalized budget *NDB*

8 Conclusion

In the paper the exact and approximate algorithms are presented. It follows from computational experiments that about 60% approximate solutions are equal to optimal solutions (Fig. 3). We are of the opinion that the very good performance of the approximate algorithm obviates, in most practical cases, the application of the exact algorithm, that is for more time-consuming. Moreover, the proposed algorithms may be also applied for solving the network design problem maximizing the number of different routes between each pair of nodes subject to budget constraint. In this case, each set W^i contains only the empty arc and the arc the capacity of which is equal to one.

References

1. Amiri A.: A System for the Design of Packet-switched Communication Networks with Economic Tradeoffs. Computer Communications 21 (1998) 1670–1680
2. Ford L. R., Fulkerson D. R.: Flows in Networks. Princeton University Press, Princeton, New Jersey (1962)
3. Kasprzak A.: Topological Design of the Wide Area Networks. Wroclaw University of Technology Press, Wroclaw (2001)
4. Minoux M.: Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. Networks 19 (1989) 313–360
5. Plesnik J., The Complexity of Design a Network with Minimum Diameter. Networks 11 (1981) 77–85

Modeling and Simulation of Distributed Security Models

Hee Suk Seo[1], Tae Ho Cho[1], and Sung Do Chi[2]

[1]School of Information & Communications Engineering, Modeling & Simulation Lab, Sungkyunkwan University, Suwon, 440-746, South Korea.
{histone , taecho}@ece.skku.ac.kr

[2]Department of Computer Engineering, Hangkong University, Koyang-si, Hwajeon-dong 2001-1, Seoul, 411-791, South Korea.
sdchi@mail.hangkong.ac.kr

Abstract. It is quite necessary that an organization's information network should be equipped with a proper security system based on its scale and importance. One of the effective methods is to use the simulation model for deciding which security policy and mechanism is appropriate for the complex network. The need arises for systems to coordinate with one another, to manage diverse attacks across networks. The coordination issue is the essential problem since it is beyond the scope of any one IDS (Intrusion Detection System) to deal with the intrusions. This paper shows a modeling and simulation of network security in which the multi-security agents coordinate by sharing attacker's information for the effective detection of the intrusion.

1 Introduction

Networks are developed to make computers more accessible to the outside world. Making computers more accessible to the outside is a mixed blessing. Network security is a problem that has gotten larger with the growth of the Internet [1].

For the simulation, ID agent, SVDB (Simulation based Vulnerability Database) and Firewall models are constructed based on the DEVS (Discrete EVent system Specification) formalism [2]. Since evaluating the performance of a security system directly in real world requires heavy costs and efforts, an effective alternative solution is using the simulation model. In concrete terms, using the model we can build various simulation situations, perform iterative runs, and decide which security configuration is effective in meeting the change of network environment.

As intrusions become more sophisticated, it is beyond the scope of any one IDS to deal with them [3]. Thus we placed multiple IDSes in the network where the information helpful for detecting the intrusions is shared among these agents to cope effectively with attackers. Each agent coordinates through the BBA (Blackboard architecture) [4] for detecting intrusions. If an agent detects intrusions, it transfers attacker's information to Firewall [5] model. Using this mechanism attacker's packets detected by IDS can be prevented from damaging the network.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 809–818, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Overview of the Security Model

Security models capture policies for confidentiality (Bell-LaPadula) and for integrity (Biba). Some models apply to environments where policies are static (Bell-LaPadula), others consider dynamic changes of access rights (Chinese Wall). This section explains a few security models.

2.1 The Bell-LaPadula Model

Security models are an important concept in the design and analysis of secure systems. They capture the security policy that should be enforced in the system. The Bell-LaPadula model (BLP) is the most famous of the security models. It was developed by Bell and LaPadula. BLP is a state machine model capturing the confidentiality aspects of access control. Access permissions are defined both through an access control matrix and through security levels. Security policies prevent information flowing downwards form a high security level to a low security level. These policies are commonly referred to as multi-level security (MLS). BLP only considers the information flow that occurs when a subject observes or alters an object. BLP defines security as the property of states. Multi-level security policies allow a subject to read an object only if the subject's security level dominates the object's classification [6].

2.2 The Biba Model

The Biba model addresses integrity in terms of access by subjects to objects using a state machine model very similar to that of BLP. Integrity levels form the basis of expressing integrity policies that refer to the corruption of clean high level entities by dirty low level entities. In the integrity level, information may only flow downwards. Unlike BLP, there is no single high-level integrity policy. Instead, you find a variety of approaches [7].

3 The Structure of Target Network and Simulation Model

3.1 ID Agent

ID agent have four components: PCL, Audit, Alarm, AGENT. PCL (Packet Classify Library) model receives network's packets that are generated by the intrusion generator model and classifies them according to its usage. Then it filters the classified packets to reduce processing time as the following process. For example, for the mailbomb case, TA (Task Allocator) of PCL model receives packets from the generator model and then it transmits the packets to one of the three different types of models. These are MTONE, MTTWO, MTTHREE. If the packets that are sent to MTTWO are of TCP protocol and port number is 25 then PCL model transfers the packets to an agent model for further processing. Otherwise, MTTWO ignores it since it is useless.

AGENT model has a rule-based ES (Expert System) which plays a core component role in detecting the intrusion. It also transforms the packets, that are delivered by PCL model, into facts to be used by ES. ES inferences according to the facts thus generated. If a new attack is to be added to ID model later on, the administrator classifies the attack based on packet's type and adds a proper subcomponent model to PCL model and its corresponding rules to the AGENT model.

Audit model is composed of Auditlog and Buffer model. Auditlog model plays a following role. Intrusion detection systems often provide an audit log of their own, particularly in on-the-fly systems. Audited information is generally considered security critical and deemed suitable for storage in a protected log. Auditlog model stores the audit log of the intrusion detection system like these. Next Buffer model is explained. Generally intrusion detection system processes many information, it requires the storage for these process. This is considered buffering (or cache storage) and requires that fast hardware and software be available to keep up with target system capabilities and performance.

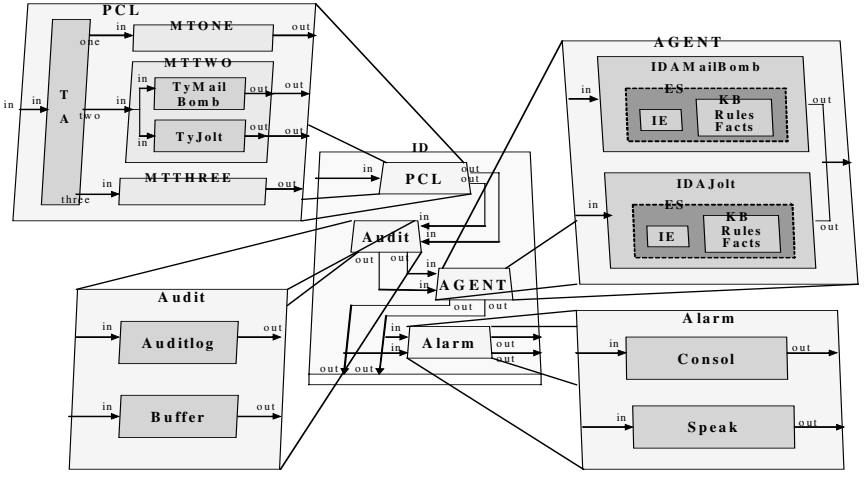


Fig. 1. Structure of intrusion detection model

Once intrusions or suspicious actions have been detected via the processing capability of the intrusion detection system, the response action most often found in available systems involves alarms. These alarms can range from simple textual messages to more involved procedures such as the sending of email to specified users, the initiation of phone calls or faxes to specified locations, or the execution of arbitrary programs in a local or remote computing environment. Alarm model has two sub-models: Consol and Speak model. Consol model send a textual messages to the monitor and Speak model rings out a alarm signal.

3.2 Vulnerability Scanner

Errors in computer systems and programs are called bugs [8]. Vulnerability scanner helps the system manager to find bugs that enable users to violate the network security policies. Vulnerability scanner is used to prepare the intrusion. We construct the SVDB for analyzing the vulnerabilities. Fig. 2 shows the structure and function of SVDB interface. SVDB has vulnerabilities of systems and policy information that can be used by security agents. Security agents-ID agents, vulnerability scanner and firewall- query the SVDB by DB interface then SVDB responses them.

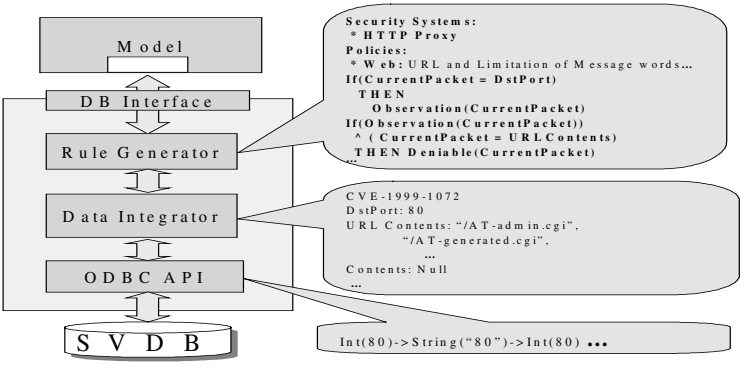


Fig. 2. Structure and function of SVDB interface

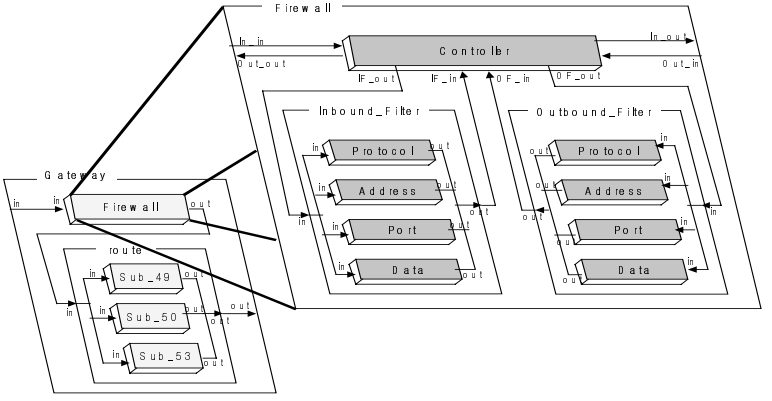


Fig. 3. Structure of firewall model

3.3 Firewall

The firewall is a way to restrict access between the Internet and the internal network [5]. Fig. 3 shows a composition of Firewall model exploited in this simulation. Firewall model is composed of a controller model, a Inbound_Filter model and a Outbound_Filter model. Inbound_Filter model includes Protocol model, Address model,

Port model and Data model. Controller model receives the packets from the internal network or from the Internet, and transfers the packets to Inbound_Filter model or Outbound_Filter model. Each Filter model processes the packets according to the security strategies and sends the packets to Controller model. Controller model sends the packets that are delivered by Inbound_Filter model or Outbound_Filter model to the network. For example, for the mailbomb case, Inbound_Filter model receives the packets from Controller model and examines the IP (Internet Protocol) address. If the packet contains the blacklist, it is ignored. If a e-mail contains harmful information, Data model of Inbound_Filter model filters the packet.

4 The Coordination among the Multi-security Agents

4.1 Communication among Agents of BB

The BBA, within the field of the distributed artificial intelligence, provides an approach to solving the coordination problem among distributed agents. BB (Blackboard) that is one of the components in the BBA is usually partitioned into several levels of abstraction appropriate for the problem at hand. The hierarchy in BB is set according to Joseph Barrus & Neil C. Rowe [9]. They proposed Danger values to be divided into five different levels. These five BB levels are Minimal, Cautionary, Noticeable, Serious and Catastrophic. We classify BB levels into each five levels for a host attack and a network attack based on these divisions. Each agent communicates by two types of messages. One is the control messages, the other is the data messages. The control messages are used to communicate between agents and controller and the data messages are required to send data between agents and blackboard.

In this paper, the BB levels for the host attack case are presented. The host attack is defined as single host out of the network hosts is attacked. In this case the attacked host inserts the intrusion related information to the Host-Attack of the BB. Each agent must request the permission, sends a BB_update_request message, to the controller in order to manage consistency and contention problems. The controller sends a BB_update_permit message to the agent when the agent can write some information to BB. The agent which receives this message writes (BB_update_action) the intrusion related information to BB. After updating is done, the agent sends the BB_update_completion message to the controller. Controller, sends a BB_broadcasting_of_action_request message, reports this event to other IDSes. IDSes, which have received the necessary information from BB, send the BB_information_acquisition_completion message to the controller. The BB levels are transit according to these steps. When the BB level is at Serious level, the agent adds the source IP address to the blacklist of the Firewall model, then all packets coming from these sources are blocked.

4.2 Problem Solving by the BB

This section presents the transition procedure of each BB level and response to the transition. As example, the transition of the BB for mailbomb attack case are shown. The levels of the BB are divided by threshold values. Threshold values are selected according to the following contents.

— the policies of the network administrator and system security level : the administrator can enforce the system security configuration to protect the network system. In this case threshold values can be a little low.
— network speed and configuration environment : threshold values can be varied by the network speed and configuration environment. Namely threshold values of network, support the high speed network environment, can be a little higher than relatively low speed. And a case of having many internal processes in a local host is a little higher than many networking processes.
— system performance (CPU, memory) : threshold values of system, have a fast CPU speed, can be a little higher than relatively low. A memory isn't different from a CPU.
— operating system types : threshold values, OS enforce the security level, can be a little higher than those not.
— attack types : threshold values can be varied according to the attack types.

Consider a case out of the various attack scenarios. There are host A and host B which are mail server and these two servers are attacked by mailbomb attack. Though there are various attack scenarios, we select one scenario.

1. An attacker starts to attack from the external network. If a threshold value of the host A reaches at the Minimal level, the initial Passive state is transit to the Minimal state of the Host-Attack.
2. While the host A is attacked, the attack for the host B is started. At last the BB transits the Minimal level. In this case the BB transits the Minimal level of the Network-Attack since the state of the BB was the Minimal state of the Host-Attack.
3. The threshold value of the host A reaches at the Cautionary level by continuing to attack then the BB transits to the Cautionary level of the Host-Attack.
4. The BB level is at the Cautionary level now and the host B is continually attacked. If the threshold value of the host B reaches at the Cautionary level, the BB transits the Cautionary level of the Network-Attack.
5. The BB level is at the Cautionary level of the Network-Attack now and the threshold value of the host A reaches at the Noticeable level. The BB transits the Noticeable of the Host-Attack.
6. The level of the BB is at the Noticeable level and the host B level reaches at the Noticeable level. The BB level transits to the Noticeable level of the Network-Attack. Since the BB level reaches the Noticeable level of the Network-Attack, the controller informs all IP addresses coming from attacker's source to the firewall. Firewall, blocks these packets, protects from damaging the host A and host B.

7. The level of the BB is at the Noticeable level and the host A level reaches at the Serious level. All network packets coming from the external network to the host A are blocked since the BB level is at the Serious level.

8. The level of the BB is at the Noticeable level and the attacker continually attacks the network so the BB level reaches at the Serious level. If the BB level is at the Serious level of the Network-Attack, this case is regarded to seriously damage the network. Finally the controller transfer these information to the firewall then the firewall blocks all packets coming to the network. By these follow-up measures the administrator grasps the extent of the damage, restores the network and prepares for the post attack.

4.3 Coordination between the IDSes and a Firewall Model

The IDSes and a firewall also communicate by two types of messages. One is a control messages, the other is a data messages. The control messages are used between the ID agent and the controller and the data messages are used between the firewall and the controller.

The coordination between the IDSes and the firewall is presented for the host attack case. The BB level is at the level, packets are blocked by firewall. A attacked ID agent transfers the BB_update_completion message to the controller then the controller sends the BB_unicasting_of_action_request message to the firewall. The firewall received this message reads (BB_information_retrieval_action) the intrusion related information on the BB. Reading is done, it transfers the BB_information_acquisition_completion message to the controller. It changes its rule table with these acquisition information and prevents the intrusion packets from coming the network.

5 Simulation Result

We have executed simulations for two cases. One is the case for 5 levels of BB to detect the intrusion, the other is the case for 10 levels of BB to detect the intrusion. Mailbomb attack and jolt attack are used for the simulation in both cases.

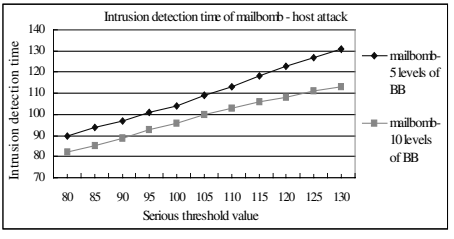


Fig. 4. Intrusion detection time of mailbomb attack – a case of host attack

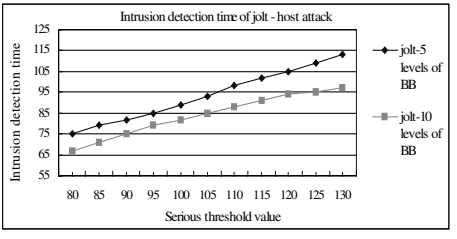


Fig. 5. Intrusion detection time of jolt attack – a case of host attack

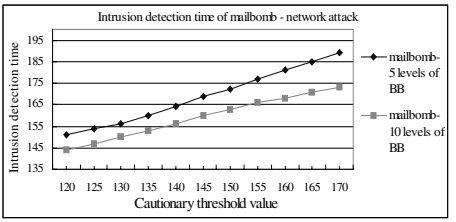


Fig. 6. Intrusion detection time of mailbomb attack – a case of network attack

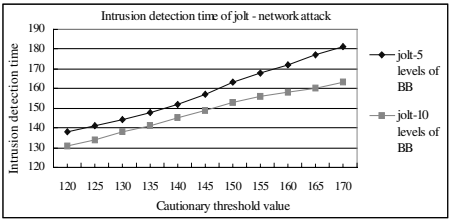


Fig. 7. Intrusion detection time of jolt attack – a case of network attack

Mailbomb attack is a type of DoS (Denial of Service) attacks. It attacks by sending a large number of mails to a mail server. Jolt attack is also a type of DoS attacks. It slices a IP datagram into small pieces and then transmits the packets of these pieces to the target system. Then the CPU (Central Processing Unit) of target system that receives these packets is over loaded since the target system has to store and reassemble all these packets. As a result, the utilization of the CPU reaches close to 100 percents and can't handle any other processes. The intrusion detection time, false positive and false negative error ratio are measured for the performance indexes in the simulation. We present in the previous studies that the multiple IDSes are superior in intrusion detection to single one [10]. In the previous research the levels of the BB are divided into 5 levels-Minimal, Cautionary, Noticeable, Serious and Catastrophic level. We have executed the simulation for the comparison the case of five BB levels with that of ten BB levels. This research shows that the proposed system with ten BB levels detects effectively the intrusion than the previous system with five BB levels.

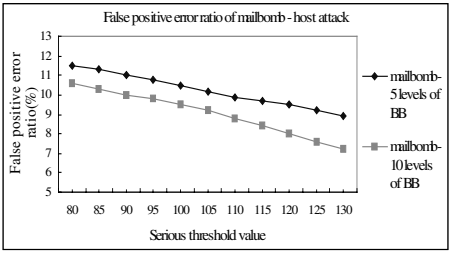


Fig. 8. FPER of mailbomb attack – a case of host attack

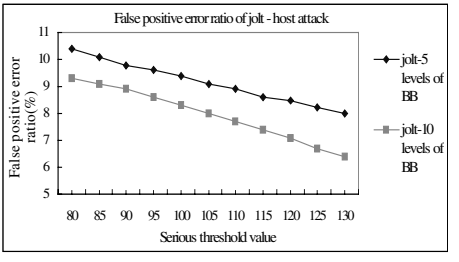


Fig. 9. FPER of jolt attack – a case of host attack

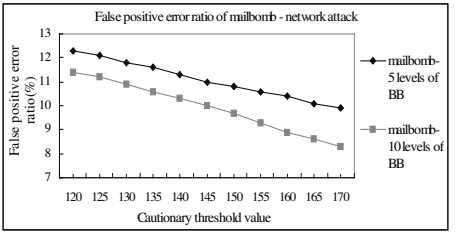


Fig. 10. FPER of mailbomb attack – a case of network attack

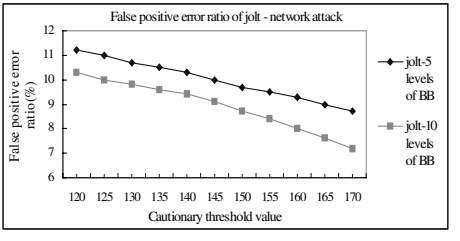


Fig. 11. FPER of jolt attack – a case of network attack

Fig. 4,5,8,9,12,13 are the case for a single IDS is attacked and Fig. 6,7,10,11,14,15 are the case for multiple IDSes are attacked. Fig. 4-7 present the intrusion detection time of the proposed system with 10 BB levels and the previous system with 5 BB levels for the mailbomb and jolt attack. The selected BB levels for the simulation are the Serious level of the Host-Attack and the Cautionary level of the Network-Attack. The other threshold values are changed according to the same ratio of change in these two threshold values. The proposed system with the 10 BB levels detects the intrusion faster than the previous system with the 5 BB levels for all the threshold values. The result shows that the proposed system coordinates effectively through BBA for detecting the intrusion. The faster the intrusion is detected, the earlier the administrators can correspond to the intrusion. It is important that the network administrator to respond at the early stage of the intrusion for the safety of the network. When the security level is weakened by increasing the serious threshold value in our system, the difference in the intrusion detection time between the proposed system and the previous system becomes larger. Because the lower the security level, the stronger the sensitivity becomes due to the information sharing among IDSes. This phenomenon related to the sensitivity applies to all other cases of the simulation results.

Fig. 8-11 show the false positive error ratio of the proposed system with the 10 BB levels and the previous system with the 5 BB levels for the mailbomb and jolt attack. A false positive is basically an alarm on acceptable behavior. Fig. 8-11 show that the false positive error ratio is increased by strengthening of the security level. This increase in the error ratio is due to the fact that the higher the security level, the more error IDSes make in both cases. Fig. 8-11 shows that the proposed system has lower error ratio than the previous system. These results means that the intrusion detection performance is improved since the sensitivity of the detection is increased by the subdivided BB level.

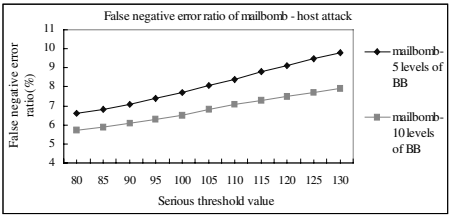


Fig. 12. FNER of mailbomb attack – a case of host attack

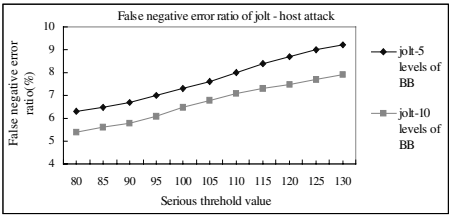


Fig. 13. FNER of jolt attack – a case of host attack

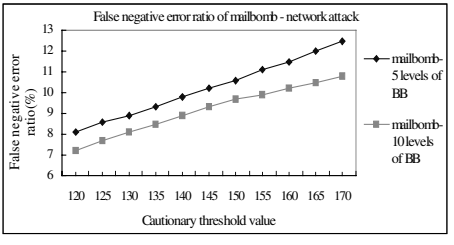


Fig. 14. FNER of mailbomb attack – a case of network attack

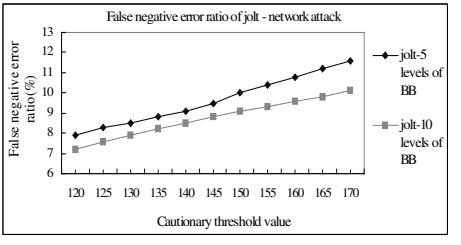


Fig. 15. FNER of jolt attack – a case of network attack

Fig. 12–15 show the false negative error ratio of the proposed system with the 10 BB levels and the previous system with the 5 BB levels for the mailbomb and jolt attack. A false negative is basically an a missed alarm condition. Fig. 12–15 show the decrease of the false positive error ratio as the security level is strengthened. For all cases, the error ratio of the proposed system is lower than that of the previous system since the intrusions are detected based on the shared information.

6 Conclusion and Future Work

As the usage of the network increases, intrusions occur more frequently and become more widespread and sophisticated. If multiple agents share the intrusion related information with one another, the detection capability can be enhanced. The system which uses BBA for the information sharing can be easily expanded by adding new agents and increasing the number of BB levels. The coordination between the Firewall component and IDS will provide the added efficiency in safe guarding the network. In the future, diverse types of intrusions should be simulated and the simulation environment should also provide a proper set of threshold values according to the specific target system being modeled.

References

1. S Mclure, J. Scambray and G. Kurtz, "Hacking Exposed: Network Security Secrets and Solutions," McGraw-Hill, 1999.
2. B. P. Zeigler, "Object-Oriented Simulation with Hierarchical, Modular Models," San Diego, CA, USA:Academic Press, 1990.
3. S. Northcutt, "Network Intrusion Detection – An Analyst's Handbook," New Riders Publishing, 1999.
4. K. Decker, A. Garvey, M. Humphrey and V. R. Lesser, "Control Heuristics for Scheduling in a Parallel Blackboard System," International Journal of pattern Recognition and Artificial Intelligence, Vol. 7, No. 2, pp. 243–264, 1993.
5. E. D. Zwicky, "Building Internet Firewalls," 2nd Ed., O'Relly & Associates, 2000.
6. Dieter Gollmann, "Computer Security," Wiley, 1999.
7. K. J. Biba. "Integrity consideration for secure computer systems," Technical Report ESDTR-76-372, MTR-3153, The MITRE Corporation, Bedford, MA, Apr. 1977.
8. J. Forristal and G. Shipley, "Vulnerability Assessment Scanners," Network Computing, 8 Jan. 2001.
9. J. Barrus and N. C. Rowe, "A Distributed Autonomous-Agent Network-Intrusion Detection and Response System," Proceedings of Command and Control Research and Technology Symposium, Monterey CA, pp. 577-586, Jun. 1998.
10. H.S. Seo and T.H. Cho, "Modeling and Simulation for Detecting a Distributed Denial of Service Attack," Lecture Notes in Artificial Intelligence, Springer Verlag, Dec. 2002.

An Efficient Collective Communication Method for Grid Scale Networks

Kyung-Lang Park[1], Hwang-Jik Lee[1], Youn-Joo Lee[1], Oh-Young Kwon[2], Sung-Yong Park[3], Hyung-Woo Park[4], and Shin-Dug Kim[1]

[1] Dept. of Computer Science, Yonsei University
134 Shinchon-Dong, Seodaemun-Gu, Seoul 120-749, Korea
{lanx, bear22, garfield, sdkim}@parallel.yonsei.ac.kr

[2] Dept. of Computer Engineering, Korea University of Technology and Education P.O. BOX 55, Chonan, 330-600, Korea
oykwon@kut.ac.kr

[3] Dept. of Computer Science, Sogang University
1 Shinsoo-Dong, Mapo-Ku, Seoul 121-742, Korea
parksy@ccs.sogang.ac.kr

[4] Korea Institute of Science and Technology Information,
P.O. BOX 122, Yusong, Taejun, 305-806, Korea
hwpark@hpcnet.ne.kr

Abstract. This research is to design a collection of effective collective operations for the Grid scale network environment. In the Grid, several network features should be specified and adapted into the algorithmic design of collective operations. For this, we characterize the Grid and design hierarchical latency optimal tree algorithm for the MPI (message passing interface) library. The experimental results show that performance gain in performing collective operations can be achieved by around 160% compared with MPICH-G2 which is based on the flat algorithm.

1 Introduction

As the computational Grid [7, 8] is emerging, a lot of existing computational approaches are gradually changed into a new paradigm to utilize ultimately unlimited resource as a single entity. However, it's difficult to design reliable applications to be performed effectively in Grid environments. The difficulty is mainly caused by architectural features of the Grid that is configured with a large number of heterogeneous and wide area networks. In such environments, many researchers attempt to improve collective communication methods because application performance is not affected significantly by the core services of middleware but by programming and communication libraries, where collective communication methods are the central part of these libraries for efficient parallel programming environment.

Previously, a number of collective operations have been designed and implemented to be adapted and optimized for each computing environment, but not for the Grid

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 819–828, 2003.
© Springer-Verlag Berlin Heidelberg 2003

scale network. This research is to design effective collective operations for the Grid scale network environment. Specifically, several features from the Grid scale networks are specified and adapted into the algorithmic design of collective operations. To accomplish this goal, we designed a hierarchical latency optimal tree (HLOT) algorithm and implemented the MPI library, called MPICH-GX, by using multi-level network topology and network weather service (NWS) [12]. The proposed MPI library, MPICH-GX, shows the performance gain by around 160% when performing collective operations under our Grid testbed which comprise 8 clusters.

In Section 2, we introduce general concept of collective operations. In Section 3, we describe our advanced collective communication algorithm and implementation in detail. Section 4 shows several results of our implementation. Finally, we present the conclusions of the research in Section 5.

2 Background and Related Work

Previous MPI related studies [1,2,3,6,10,11] show that significant performance can be achieved by improving communication capability in Grid environments, especially by considering collective operations. In designing the collective operations, the key factors are to maximize the parallelism and to minimize communication cost. Based on these factors, several algorithms had been designed but these are classified into three major groups. The first group is based on binomial tree algorithm and it is focused on the parallelism of communication stages. Second one is based on minimum spanning tree (MST). They attempt to minimize communication cost by selecting the fastest link. Third one is based on the flat tree. These are methods to have reflected characteristics of wide area links and to maximize the overlapping of communication when used postal (non-blocking) model. In general, binomial tree had been chosen as an optimal algorithm when collective communication primitives were designed under MPI programming environment [9]. However, the work in [1] showed that the flat tree could provide better performance at the WAN level, because, in the flat tree, communications do not go through the WAN link more than once. Based on this wide area communication concept, flat tree algorithm was regarded as an optimal algorithm in collective communication primitives in WAN. [1,3,10]

Along with the progress of these communication algorithms, there was also significant progress in the implementation of MPI collective operations. MagPIe [2] communication library and MPICH-G2 [12] Grid-enabled library are the representatives of this research field. They implemented actual MPI library by adding the multi level communication on the previously studied algorithms that can reflect the network features. In the MagPIe system, they divide processes into two levels, i.e., WAN and LAN levels, and each process at WAN level becomes a coordinator or a sub-root manager. When any collective operation like *MPI_Bcast* starts, the root node sends the messages to the coordinators at the WAN level and each coordinator sends them to its slave nodes within its own cluster located at the LAN level. As mentioned before, the flat tree algorithm is used at the WAN level and the binomial tree algorithm is used at LAN level [3]. In the MPICH-G2, communication is divided into four levels, i.e.,

WAN, LAN, intraTCP, and vendorMPI. Also flat-tree algorithm is applied to the WAN level and the binomial tree to other levels. In such WAN environments, a sort of layered topology is chosen and communications are performed based on the order of levels [6].

However, there are still important problems to optimize collective operations in the Grids, because the Grid environments are basically constructed as heterogeneous networks with different performance ranges so that it is rather different from general wide area networks. So, it is inadequate if we define the characteristics of Grid Scale WAN as a uniform entity and apply a simple flat-tree algorithm. Thus a more adaptive algorithm is designed and implemented in this work.

3 Designing an Efficient Algorithm in Grid

3.1 Characterizing the Grid Scale Network

In designing the collective communication primitives, one of the most important features is to reflect the status of network environments under which message passing programs are performed. In early stage, binomial tree was regarded as one of the most suitable algorithms because MPI execution environments are implicitly considered as parallel machine and/or supercomputer environments that are configured with extremely high speed internal communication links. However, execution environment for the Grid scale network is extended to the workstation pools and also widely distributed clusters that are linked via external networks including the Internet. Thus, several major features for the Grid network should be clarified to be reflected to the design of collective communications.

The characteristics of grid network originates from the fact that all resources are geometrically distributed, and the following features of Grid network should be considered for designing the collective communication algorithm.

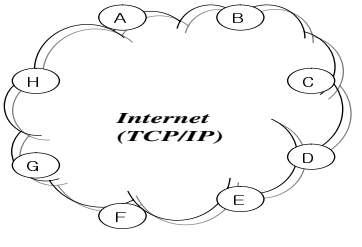
1. *Network status can be changed radically and dynamically*
2. *Latency is the most considerable parameter and not fixed as constant*

The first characteristic is due to the nature of distributed resources. The Grid comprise of the various heterogeneous networks including the Internet allowed to the mass of the people. Therefore status of the networks can be changed dynamically according to various factors such as the network traffic. We can easily experience a terrible network delay in the daytime when a lot of people are connected in networks. It shows implicitly the need of the intelligent communication algorithm which considers dynamic network status.

Second one also can be derived from the dynamic states of the network. When transferring a message from one point to another, the total communication time includes the overhead, gap, and latency [4]. The overhead can be defined as the length of time that a processor in engaged in the transmission or reception of each message. The gap is defined as the minimum time interval between consecutive message transmissions at a processor. Finally, the latency is incurred in communicating a message

containing a word from its source module to its target module, which includes the RTT (Round Trip Time), propagation delay, routing and so on. Among them, the overhead is extremely smaller than other parameters in wide area networks, so that it can be eliminated. The gap is relatively considerable. If the message size is increased, the gap is increased too because of the limitation of bandwidth. However, considering the parallel programming which comprises of only the fixed data array transmission, and not include huge data transmissions like FTP, the gap can be fixed as a relatively small constant. Thus, the most dynamic parameter is the latency and total communication delay is dominated by the latency. Surely, if we can obtain a more generalized parameter like ideal communication cost which covers all factors completely, it will be more useful. But, it also can be a naïve method which cannot be implemented.

An example shown in Figure 1 and 2 shows how the HLOT and other algorithms are performed to utilize these characteristics. Consider a case that a set of clusters is located at geometrically distributed sites as shown in Figure 1. The right side table shows the communication cost. In such an environment, if we attempt to broadcast messages by using such well-known algorithms, different results can be obtained as in Figure 2.



	A	B	C	D	E	F	G	H
A	0	400	800	420	520	500	800	150
B	430	0	200	450	700	540	700	370
C	700	250	0	500	870	850	700	600
D	400	450	500	0	500	950	1300	780
E	500	670	800	500	0	500	700	520
F	580	550	850	800	500	0	280	480
G	920	700	700	1260	720	300	0	200
H	300	520	600	800	550	320	200	0

Fig. 1. Latency matrix of virtual organizations in Grid scale network.

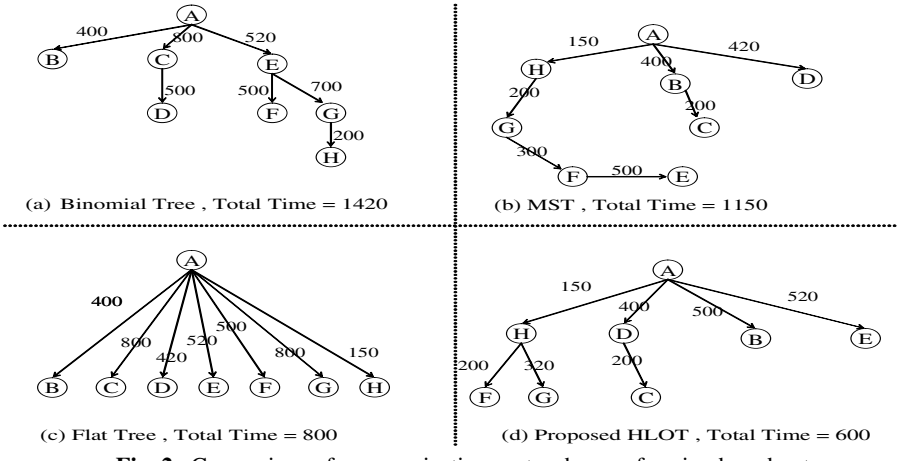


Fig. 2. Comparison of communication costs when performing broadcast

Each algorithm has its own unique characteristics and advantage, but the proposed HLOT (Hierarchical Latency Optimal Tree) algorithm shows the most effective performance in extremely wide area network such as the Grid. We will explain about this HLOT algorithm in more detail in the next section. First candidate is a binomial tree algorithm. We already mentioned that it is focused on minimizing the number of send/receive stages. But, in Grid network, overhead is extremely small so that all transmissions from one node can be treated as just one stage using non-blocking send. Therefore, the merit of binomial tree cannot be effective in this case. As shown in Figure 2, the binomial tree includes a path connected by A, E, G, and H, which takes 1420 time units. The second one is a minimum spanning tree (MST) algorithm which selects only the lowest latency link one by one. It can be latency optimal in Grid network but such a long, inefficient path can be occurred. In Figure 2, a long path of A, H, G, F, and E is determined for broadcasting so that overall communication time can be 1150. To improve the efficiency of MST, that path should be removed. The third one is a flat tree algorithm which can maximize the merit of non-blocking send. It can broadcast in only one stage by sending all messages from the root to others. But, if there are one or more high latency links from root, its performance can be degraded significantly. Thus, long latency occurred between A and G, but flat tree algorithm can't avoid that link. Finally, the proposed HLOT algorithm can complement those disadvantages of other algorithms. Basically, the root node is in charge of transferring messages and communication links with long latencies from the root node are eliminated. In Figure 2, the HLOT shows 600 time units as the lowest communication path for broadcasting message.

3.2 Hierarchical Latency-Optimal Tree (HLOT) Algorithm

In this section, we will present the detailed design specification and analysis of the HLOT algorithm. The HLOT algorithm is based on Prim's algorithm and created by adapting additional two principles basically. The first one is the latency-optimal feature. Namely the algorithm is based on the latency information as an important feature rather than considering any other parameters in the Grid environment. Thus communication links consisting of an optimal latency are selected. The next important feature is to consider direct links first. For the collective operations, using direct links from the root node can be effective due to the characteristics of wide area network. The basic algorithm is shown in Figure 3.

The HLOT algorithm is designed by taking advantage of structural merits of the flat tree and the minimal spanning tree. Especially, the HLOT can provide the effect of non-blocking send operations as in the flat tree, but it also can avoid some high latency links by selecting latency optimal links. Performance evaluation of the HLOT algorithm is appeared in Section 5, but we first analyze the algorithm via LogP model [4] that is commonly used as a performance model for message passing system. LogP is a modeling technique using four parameters such as latency, overhead, gap, and processors. The time to broadcast can be described as following LogP model.

Variables :
Y : a set of selected nodes, V : an universal set of nodes
W[][] : 2-dimensional latency array, e(c,n): the edge from c to n
D[n] : the sum of latencies for the selected path from the root to node n
Output :
F : a set of selected edges,
Algorithm steps :

```
F = Ø ;
Y = { root };
while (V-Y • Ø ) {     // repeat until finding the path to all nodes
    choose fastest link e(c,n);  // (c is an element of Y and n is one of V-Y)
    if( D[n] is higher than W[0][n] )
            continue;          // try again
    add c to Y;
    add e(c,n) to F;
    Update D[n] }
```

Fig. 3. HLOT algorithm.

First, T_B that is the total time to broadcast a particular message can be expressed by selecting maximum value of completion times required for all processes participating in broadcasting.

$$T_B = \sum_{i=0}^{size-1} Max\{T_{pi}\}, \text{ where size = | V |} \tag{1}$$

T_{Pi} means the completion time of the *i*-th process expressed as the following equation.

$$T_{Pi} = \begin{cases} T_{Pj} + (c-1)(T_{SO} + T_G) + L_{ji} + T_{RO} (i \neq 0) \\ 0 (i = 0) \end{cases} \tag{2}$$

where *j* is the parent's index of the *i*-th process, T_{Pj} is the delivery time from the root to j, c is the number of child nodes of *j*, and T_{SO}, T_G, T_{RO} are the time of sending overhead, gap, and receiving overhead in order. That is, the time to receive a message from the *i*-th process can be obtained as the summation of the time delivered to the parent and the time to be transferred from the parent. In the case of flat tree algorithm, T_{pj} becomes zero all the time because the root node is the parents to all the processes. Also L_{ji} becomes L_{0i} as the latency directly from the root from the *i*-th process. As mentioned, the overhead involved in message transfer is extremely smaller than the completion time in Grid environments. Assume fixed size data transfer, gap and overhead parameters can be eliminated. Thus the broadcasting time T_B can be represented as:

$$\text{Flat Tree: } T_B = Max\{L_{0i}\} \tag{3}$$

$$\text{Others: } T_B = Max\{L_j + L_{ji}\} \tag{4}$$

where *Lj* is the sum of latency in paths form 0 to *j*. The biggest value of latency from the root to all processes (L_{0i}) influence the whole performance in flat tree algorithm. But, in others such as MST and HLOT, the biggest value in sum of the latency paths coming to the parent (L_j) and the latency from the parent to child (L_{ji}) influence whole performance. In designing the HLOT algorithm, we assume that a link is selected when the sum of L_j and L_{ji} is smaller than that of L_{0i}, so that HLOT algorithm is faster than flat tree always. But, it is not always true in other algorithms such as MST, binomial, and so on. On the contrary, by the use of several long distance links, performance becomes worse than that of flat tree.

3.3 Implementation

In general, some restrictions exist to design and implement the proposed algorithm as the collective communication primitives in the MPI library. HLOT algorithm also meets two important restrictions for implementation. The first one is how to collect latency information. The overhead to measure the latency of all nodes occurs necessarily. Therefore, we need a mechanism to measure the correct latency and not to countervail the efficiency of the HLOT algorithm. The second one is the overhead of HLOT algorithm itself. The process of finding an optimal latency path is based on the greedy approach. The HLOT algorithm show more than $O(n^2 \log n)$ complexity. Thus, a policy to determine the status of nodes required for the HLOT algorithm is required. Two techniques, the NWS (Network Weather Service) [12] and the multi-level communication mechanism [6] are chosen for this goal.

The NWS is a component that can supply network state between linked nodes. The NWS also provides APIs (Application Programming Interface) so that users can program with NWS directly. To collect latency information, these APIs are modified into the MPI Library. Until now, we utilize latency information of one specific point, but can utilize to compose topology by intelligence using continuous statistical data and prediction.

Multi-level communication is a method that can improve the performance by separating the given network into several layers according to its characteristics. It can select the suitable algorithms for each layer. We also leverage the concept of multi-level communication by adapting our algorithms into MPI library efficiently.

MPICH-G2 that is widely used as the Grid-enabled MPI library divides the network into four layers. But the Grid implicitly means the use of extremely distributed computing and network resources, and four-level topology is a very restrictive communication architecture. Also, other three levels except the WAN level include only local area without considering the WAN level. Consequently, the WAN level should be classified into more levels to cope with the various network environments in Grids. Therefore we divide the WAN into two levels, called as light-WAN and heavy-WAN, and use the flat tree algorithm to the light-WAN level which includes relatively stable and uniform links and apply the HLOT algorithm to the heavy-WAN level which includes various heterogeneous links with four characteristics mentioned in Section 3. A user can divide these levels by using explicit parameters in the RSL file or it can be performed automatically by using latency information obtained by NWS. Figure 4

shows an example of abstract diagram constructed by the five-level topology with HLOT algorithm. In Figure 4, the root process broadcasts a message to sub-root by using HLOT. Thus a heavy latency link can be avoided in heavy-WAN, drawn as the dotted line.

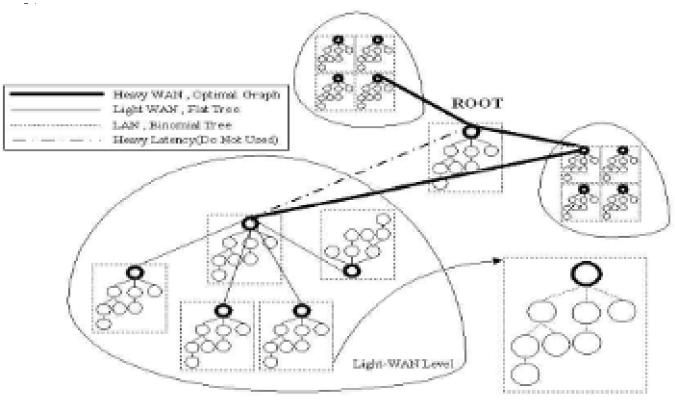


Fig. 4. Multi-level communication with HLOT algorithm.

4 Experimental Result

To evaluate the impact of our HLOT algorithm and its MPI implementation, we will show the experimental results in a practical Grid environment. Our testbed consists of 8 clusters of 1GHz/512MB Pentium machines connected by general purpose Internet and located at 4 distributed organizations – Yonsei, Ajou, KUT and KISTI. Our workload comprises a *simple MPI_Bcast* using 2 byte message size which is a representative collective operation in MPI Libraries, and it can clearly show the behavior of implemented MPI system. There are several accurate measurement techniques, but we execute a program by once for each sample to make the most independent events. It can avoid pipelining effects absolutely. To obtain 99% confidence, we observe the 130 samples which are shown in Figure 5. In this figure, the dotted line means the samples of MPICH-G2 which uses the 4 level topology and flat algorithm and the solid line means the samples of MPICH-GX (eXtension) which uses the extended 5 level topology and the HLOT algorithm.

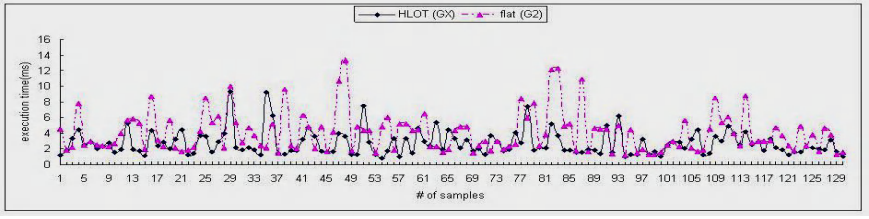


Fig. 5. Observed samples of MPI_Bcast

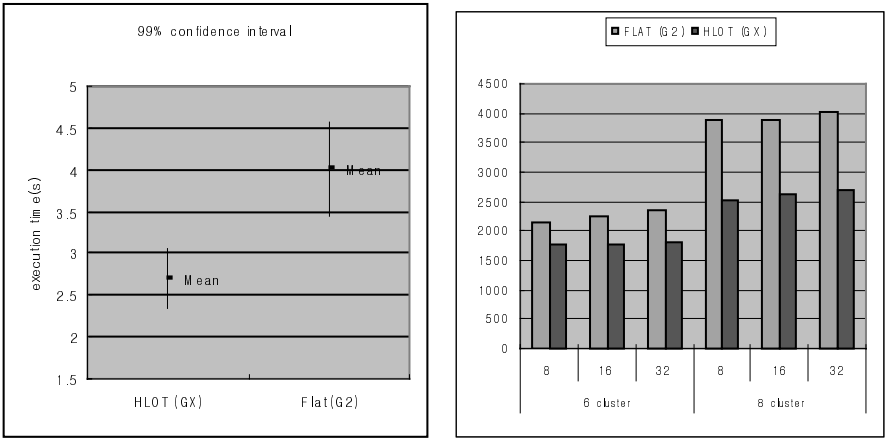


Fig. 6. (a) Approximate visual test between the HLOT (GX) and the flat tree (G2). (b) Average completion time of *MPI_Bcast* in different clusters

We repeated the same experiments by using different numbers of clusters, processes, and message sizes. As shown in Figure 6 (b), if the number of clusters is increased, performance gain increases more significantly because a lot of high latency links are involved. In the 8 cluster system with 32 processes, we can obtain the maximum rate of improvement. In Figure 7, if the size of messages becomes bigger, performance improvement becomes a little larger because the disadvantage caused by using high latency link is dependant on the message size. Message size can affect other parameters like gap. But, we are focused on the latency parameter because latency is the most effective parameter as we mentioned in Section 3. The relation between message size and other parameters is described in [2] by detail.

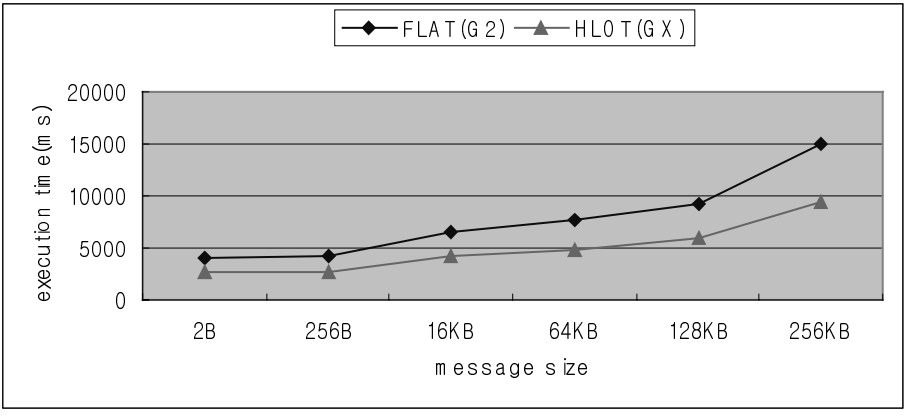


Fig. 7. MPI_Bcast with different message sizes.

5 Conclusion

The goal of this research is to improve the performance of collective communication method in the Grid scale networks. Current various programming libraries attempt to optimize the communication performance, but they still don't consider the various characteristics of the Grid network. So, we design the HLOT algorithm which is based on the features of Grid aggressively and implement the MPI library, MPICH-GX by using the proposed algorithm with two methods, i.e., multi level topology and network weather service. Thus the performance improvement can be achieved by around 160% compared with MPICH-G2 in our testbed which is distributed in 4 locations and comprise 8 clusters. Here, the basic concept of wide area collective communications based on the flat tree should be changed by considering the various network characteristics.

References

1. M. Bernaschi and G.Iannello. Collective Communication Operations: Experimental Results vs. Theory, Concurrency : Paractice and Experience, 10(5), (1998), 359–386.
2. T. Kielmann, H. E. Bal and S. Gorlatch, Bandwidth-efficient Collective Communication for Clustered Wide Area Systems, In proc. International Parallel and Distributed Processing Symposium, Mexico, (2000), 492–499.
3. T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang, MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems, In Proc. Symposium on Principles and Practice of Parallel Programming , Atlanta, GA, 5 (1999), 131–140.
4. D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauers, E. Santos, R.Subramonian and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation, In Proc. Symposium on Principles and Practice of Parallel Programming, CA , 5 (1993) , 1–12.
5. I. Foster and N. Karonis, A grid-enabled MPI: Message passing in heterogeneous distributed computing systems, In Proc. Supercomputing '98, 11 (1998).
6. N. Karonis, B. de Supinski, I. Foster, W. Gropp E. Lusk, and J. Bresnahan, Exploiting hierarchy in parallel computer networks to optimize collective operation performance, In Proc. International Parallel and Distributed Processing Symposium, (2000).
7. I. Foster and C. Kesselman, eds. The GRID : Blueprint for a New Computing Infrastructure, Morgan Kaufmann, (1998).
8. I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit", International Journal of Supercomputer Applications, 11(2), (1997), 115–128.
9. Message Passing Interface Forum, MPI: A Message-Passing Interface standard, International Journal of Supercomputer Applications, 8(3/4), (1994), 165–414.
10. MPICH-G2, Online at http://www3.niu.edu/mpi/.
11. P.B. Bhat, C.S. Raghavendra, and Viktor K. Prasanna, Efficient Collective Communication in Distributed Heterogeneous Systems, International Conference on Distributed Computing Systems, 5 (1999).
12. Network Weather Service, Online at http://nws.cs.ucsb.edu/.

Reliability Theory Model and Expected Life Shortest Path in Stochastic and Time-Dependent Networks

Guo-zhen Tan[1], Xiang-fu Xia[1], and Wen Gao[2]

[1] Department of Computer Science and Engineering, Dalian University of Technology Dalian, Liaoning 116024, China***
gztan@dlut.edu.cn
[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080,China

Abstract. We consider the priori expected shortest path problem from a single origin to a single destination for each departure time in stochastic and time-dependent networks. Such problem requires more than standard shortest path techniques. First, we transform this problem into the problem of systemic reliability, and identify a weaker consistent reliability condition that insures the validity of generalized dynamic-programming method based on reliability priority. Furthermore, we propose a new dominance discriminance, which makes the parameter in traditional discriminance drop from two dimensions to one dimension and reduces the possibility of not doing dominance compare among paths. Finally, we present the expected life shortest path algorithm.

1 Introduction

The study of routing problem in stochastic time-dependent networks is of timely importance, as stochasticity in traffic networks is both conceptually recognized and experimentally proved [1]. A stochastic time-dependent network is a network where link travel times are random variables with time-dependent distributions [2].

Many researchers have tried to use different ways to solve the stochastic shortest path problem, including reducing the search space to find feasible solutions. A method proposed by Alexopoulos is based on iterative partitioning of the network state space and provides bounds that improve each iteration and eventually become equal to the respective measures related to the shortest paths in networks [3]. We have studied the shortest path algorithm in time-dependent networks, and got some valuable conclusions [4]. Hall has studied a specific variant of optimal routing problem in stochastic and time-dependent networks where a routing policy depends only on arrival times at decision nodes [5]. Wellman, Ford, and Larson have demonstrated that standard search algorithms could find

*** This work was supported in part by Grand 99025 of Ministry of Education and Grand 9810200104 of Liaoning Science Foundation, China.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 829–838, 2003.
© Springer-Verlag Berlin Heidelberg 2003

the shortest path when arc travel times have weaker stochastic consistency [6]. This property in turn justifies a generic pruning method using stochastic dominance. But the dominance discriminance used in [6] contains two parameters. J.Boyan and M.Mitzenmacher [7] have studied the problem of the scheduling policy in stochastic networks, using the reliability theory. They consider the waiting time for buses, and they present the optimal scheduling algorithm when the waiting times are independent random variables with increasing failure rate. But they cannot realize this algorithm.

In this paper, we study the shortest path problem in stochastic and time-dependent networks, an important problem in communication, transportation and internet systems. Using reliability theory, this problem is transformed into finding a path based on reliability priority. We consider the travel time of the entire path from source node to destination node, and our goal is to select a statistically optimal path. We identify a weaker consistent reliability condition that insures the validity of generalized dynamic-programming method based on reliability theory. The consistent reliability property essentially is to make use of reliability theory to describe the First-in-First-out property of every arc in the networks. Furthermore, a new discriminance is proposed to reduce the two parameters in extant discriminance [8] to one, which makes this discriminance feasible and reduces the possibility of not doing compare among paths. Finally, we present the Expected-Life-Shortest-Path algorithm, which can be used to solve problems of considerably great size.

2 Preliminary Knowledge

We first briefly introduce the basic concepts associated with reliability theory, which makes this paper easier to understand. Further information can be found in texts such as [8] or [9].

We use a non-negative random variable X to denote a system's life, where X has cumulative distribution function $F(t)$. Then we define the survival function of this system to be $S(t) = 1 - F(t)$. Obviously, $S(t)$ describes the system's viability in the time interval between *0* and *t*.

The system's mean life is

$$EX = \int_0^\infty t dF(t) = \int_0^\infty dF(t) \int_0^t dx = \int_0^\infty dx \int_x^\infty dF(t) = \int_0^\infty S(t)dt. \quad (1)$$

The mean residual life of the system at time t is $E[X - t \mid X \geq t]$, which is dependent of t.

Let $d = sup\{t \mid S(t) > 0\}$, and $D = [0, d)$.

Suppose $f(t)$ is the probability density function of X, then we define the failure rate of this system to be

$$r(t) = \frac{f(t)}{S(t)}, \qquad t \in D \quad (2)$$

On second thoughts, we can get (3) and (4)

$$r(t) = -\frac{d}{dt} \ln S(t) \tag{3}$$

$$S(t) = e^{-\int_0^t r(x)dx} \tag{4}$$

3 Reliability Theory Model

Since the reliability theory is the theory focusing on a system's stochastic life, we first depict the stochastic and time-dependent network as a system, called *STDN-R* network system.Then we transform the shortest path problem in stochastic and time-dependent networks into the problem of the system's life. We construct a system consisting of the entire public transit network together with travel time T, where T is the time that a traveller takes from the source node to the destination node, thus T is the system's life. To give expression to the dynamic process of the system life, we introduce the concept of path life. The life of a path is the travel time that a traveller has spent from the source node to node v_i.

In the *STDN-R* network system, the system's life is stochastic due to the fact that the arcs' lives (i.e.costs) are stochastic. We use a non-negative random variable X to denote the system's life. The optimal goal is to manage to make the system's life as short as possible. That is to say, we expect to find a t small enough to make the system's viability lowest (that is failure rate maximal) in time range from *0* to *t*.

In this system, the traveller's arriving at the last node of the current path leads to an event called failure. The initial failure time of a system is the time when the traveller starts at the source node of the network system; the expected failure time of a system is the expected time when the traveller reaches the destination node of the network system; the initial failure time of an arc is the time when the traveller leaves the tail node of the arc; the expected failure time of an arc is the expected time when the traveller arrives at the head node of the arc.

3.1 *STDN-R* Network System

In this paper, the reliability theory model in stochastic and time-dependent networks is called *STDN-R* network system. Its formalization expression is $STDN - R = (V, A, L, \Phi)$, where V is a non-empty set of nodes; $A \subseteq V \times V$ is a finite set of arcs; L is a finite set of the possibly least life of every arc; $\Phi = \{\varphi_m^I(t) \mid I = (t_{ij}^{m-1}, t_{ij}^m], \langle v_i, v_j \rangle \in A, 1 \leq m \leq N_{ij}\}$. Thereinto, m is a number assigned to certain stream,the numbers are assigned based on their departure schedule at v_i along the arc $\langle v_i, v_j \rangle$, and the quantity of all the streams on the arc $\langle v_i, v_j \rangle$ is called traffic N_{ij}, $m \in \{1, 2, ..., N_{ij}\}$; $I = (t_{ij}^{m-1}, t_{ij}^m]$ is the time range that a stream arrives at v_i, where this stream is certain stream on the former arc adjacent to node v_i; t_{ij}^m is the departure time of m at v_i; $\varphi_m^I(t)$ is

the density function of the probability distribution which t subjects to, where t is the time arriving at v_j for m along the arc $\langle v_i, v_j \rangle$.

This is a abstract model of the network-optimizing problem, which has different meanings in different fields. For example, in bus networks, a stream represents a bus on certain arc; in computer networks, a stream represents a data package on certain link.

Let the sequence $\{s = v_0, v_1, v_2, ..., v_{h-1}, v_h = d\}$ represent the path from s to d and denote it as P_{sd}, where $\langle v_{j-1}, v_j \rangle \in A \quad 1 \leq j \leq h$. Let the expected failure time of path P_{sd} be $ET(P_{sd})$, and let its expected life be be $EL(P_{sd})$. The relationship between them is

$$ET(P_{sd}) = EL(P_{sd}) + \mu_0 \tag{5}$$

where μ_0 is the initial failure time.

In this network system, the problem of expected life shortest path is a priori expected shortest path problem from a single origin to a single destination for each departure time.

3.2 Consistent Reliability Property

Definition 1. In *STDN-R* network system, let $l_{ij}(x)$ represent the actual life of the arc $\langle v_i, v_j \rangle$ whose initial failure time is x. Let $0 \leq s \leq t$, for $\forall \langle v_i, v_j \rangle \in A$, if the network satisfies

$$r(s + l_{ij}(s)) \geq r(t + l_{ij}(t)) \tag{6}$$

we say that the network has the consistent reliability property.

This definition makes it clear that even though the initial failure time is late, failure rate will not increase. That is, the consistent reliability property essentially is to make use of reliability theory to describe the First-in-First-out property of every arc in the network.

3.3 Reliability Priority

Reliability Priority. The reliability priority describes the properties of certain path in the network, while the consistent reliability property describes the characteristics of each arc in the entire network.

Definition 2. Let $r_1(z)$ represent the failure rate of system at time z along the path P_1, and let $r_2(z)$ represent the failure rate of system at time z along the path P_2. For $\forall z$, if $r_1(z) \geq r_2(z)$, we say that path P_1 has the reliability priority relative to path P_2, i.e. $P_1 \leq_{ad} P_2$.

From (3) and (4), we can know that $r_1(z) \geq r_2(z) \Longleftrightarrow S_1(z) \leq S_2(z)$, so we can get the property as follows.

Deduction 1. Let $S_1(z)$ represent the survival function of system at time z along the path P_1, and let $S_2(z)$ represent the survival function of system at time z along the path P_2. Then we can get, for $\forall z$, $S_1(z) \leq S_2(z) \Longleftrightarrow P_1 \leq_{ad} P_2$.

In fact, the intuitionistic meaning of reliability priority is to say, if the expected life of path P_1 is not longer than that of path P_2, then path P_1 has the reliability priority relative to path P_2.

Theorem 1. $P_1 \leq_{ad} P_2 \Longleftrightarrow EL(P_1) \leq EL(P_2) \Longleftrightarrow ET(P_1) \leq ET(P_2)$.

Proof: We first prove $P_1 \leq_{ad} P_2 \Longleftrightarrow EL(P_1) \leq EL(P_2)$.

$\Longrightarrow$: Given that $P_1 \leq_{ad} P_2$, from Deduction 1, for $\forall z$, we have $S_1(z) \leq S_2(z)$. Then, from (1), we can conclude that $EL(P_1) = \int_0^\infty S_1(z)dz \leq \int_0^\infty S_2(z)dz = EL(P_2)$, i.e.$EL(P_1) \leq EL(P_2)$.

$\Longleftarrow$: Obviously.

So $P_1 \leq_{ad} P_2 \Longleftrightarrow EL(P_1) \leq EL(P_2)$.

From (5), we can conclude that $EL(P_1) \leq EL(P_2) \Longleftrightarrow ET(P_1) \leq ET(P_2)$. Proof is end.

Dominance Discriminance based on Reliability Theory. For the failure time of an arc subjects to exponential distribution $EXP(M, \lambda)$, where M is the possibly earliest failure time of the arc and λ is the mean residual life of the arc overstep M, the traditional dominance discriminance contains two parameters [8], i.e. M, λ: A has stochastic dominance relative to B, iff $M_A \leq M_B$ and $\lambda_A \leq \lambda_B$. But when $M_A \leq M_B$ and $\lambda_A \geq \lambda_B$, or $M_A \geq M_B$ and $\lambda_A \leq \lambda_B$, A can't do compare with B. In this paper, a new discriminance is proposed to reduce the two parameters in extant discriminance [8] to one, which reduces the possibility of not doing compare among paths.

Now, we consider the discriminance based on reliability theory when the probability distribution is exponential distribution $EXP(M, \lambda)$. Because the distribution function is $F(t) = 1 - e^{-\frac{t-M}{\lambda}} \quad t \geq M$, the failure rate is $r(t) = \frac{1}{\lambda}$. Let the two paths that can reach the destination node be P and P' respectively. The failure time on the last arc of path P that satisfies $EXP(M_1, \lambda_1)$ has probability p_1. The one that satisfies $EXP(M_2, \lambda_2)$ has probability p_2 and so on. Reorder the exponential distribution sequence of P and let any adjacent terms $EXP(M_i, \lambda_i)$ and $EXP(M_{i+1}, \lambda_{i+1})$ satisfy $\lambda_i \leq \lambda_{i+1}$. Do the same to path P' to obtain

P : $(EXP(M_1, \lambda_1), p_1), (EXP(M_2, \lambda_2), p_2), \cdots, (EXP(M_i, \lambda_i), p_i), \cdots$

P' : $(EXP(M'_1, \lambda'_1), p'_1), (EXP(M'_2, \lambda'_2), p'_2), \cdots, (EXP(M'_i, \lambda'_i), p'_i), \cdots$

Using the notation $ep_i = \sum_{k=1}^{i} p_k$ and the above two sequences, we can get the sequences of partial sums of probability as follows:

P : $ep_1,\ ep_2,\ \cdots,\ ep_i, \cdots,\ 1$

P' : $ep'_1,\ ep'_2,\ \cdots,\ ep'_i, \cdots,\ 1$

Let us rearrange the elements of the sequences of partial sums of probability in ascending order. We obtain the following sequence:

$mp_1,\ mp_2,\ \cdots,\ mp_i,\ \cdots,\ 1.$

Taking any element mp_i from above sequence, and letting $q = min_k\{k \mid ep_k \geq mp_i\},\ q' = min_k\{k \mid ep'_k \geq mp_i\}$.

For $\forall mp_i$, if $\lambda_q \leq \lambda'_{q'}$, then $P \leq_{ad} P'$.

Theorem 2. The dominance discriminance based on reliability theory is true.

Proof: For each $[mp_{i-1}, mp_i)$ $(i \geq 1, mp_0 = 0)$ in the probability range $[0, 1]$, there exist $\lambda_q \leq \lambda'_{q'}$, so for $\forall z$, we can get $r(z) = \frac{1}{\lambda_q} \geq \frac{1}{\lambda'_{q'}} = r'(z)$ in $[mp_{i-1}, mp_i)$, i.e. $P \leq_{ad} P'$ in $[mp_{i-1}, mp_i)$. Then let different probability interval $[mp_{i-1}, mp_i)$ multiply their singly relevant failure rate, and then find the sum. We conclude that $r(z) = \Sigma_i(mp_i - mp_{i-1}) \cdot \frac{1}{\lambda_q} \geq \Sigma_i(mp_i - mp_{i-1}) \cdot \frac{1}{\lambda'_{q'}} = r'(z)$. In fact, $r(z)$ represents the failure rate of system at time z along path P, $r'(z)$ represents the failure rate of system at time z along path P'. So, for $\forall z$, $r(z) \geq r'(z)$, from Definition 2 we can conclude $P \leq_{ad} P'$. Proof is end.

3.4 The Monotonicity Principle

Theorem 3. In the given *STDN-R* network system, let $P_1 = \{s = v_0, v_1, v_2, \cdots, v_{i-1}, v_i\}$, $P'_1 = \{s = v_0, v_1, v_2, \cdots, v_{i-1}, v_i, v_j\}$, $P_2 = \{s = v_0, v'_1, v'_2, \cdots, v'_{i-1}, v_i\}$, $P'_2 = \{s = v_0, v'_1, v'_2, \cdots, v'_{i-1}, v_i, v_j\}$. If the *STDN-R* network has the consistent reliability property, then $P_1 \leq_{ad} P_2 \Longrightarrow P'_1 \leq_{ad} P'_2$.

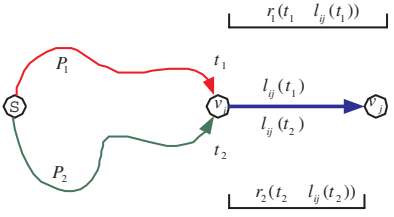


Fig. 1. Monotonicity Demonstration

Proof: Let t_1 be the failure time of path P_1 and t_2 be the failure time of path P_2. Because of $P_1 \leq_{ad} P_2$, we can conclude there must be $t_1 \leq t_2$ according to Theorem 1. Let $r_1(z)$ represent the failure rate of system at time z along path P_1, $r_2(z)$ represent the failure rate of system at time z along path P_2, and let $l_{ij}(x)$ represent the actual life of the arc $\langle v_i, v_j \rangle$ whose initial failure time is x. Because *STDN-R* network has the consistent reliability property, from Definition 1, we can conclude $r_1(t_1 + l_{ij}(t_1)) \geq r_2(t_2 + l_{ij}(t_2))$. Because the above conclusion is always tenable for starting from source node at every time, we must have $P'_1 \leq_{ad} P'_2$.

4 Calculation of Expected Failure Time and Expected Life

Let event A_{ij}^m represent a trip in which a traveller takes bus m at node v_i along arc $\langle v_i, v_j \rangle$, $m \in \{1, 2, \cdots, N_{ij}\}$, where m is a number assigned to the

bus starting from v_i to v_j.The numbers are assigned based on their departure schedule. N_{ij} represents the total number of buses travelling on arc $\langle v_i, v_j \rangle$. Then the set$\{A_{ij}^m \mid 1 \le m \le N_{ij}\}$ is the partition of the sample space Ω. That is,

$$A_{ij}^m \cap A_{ij}^n = \Phi, \ \bigcup_{m=1}^{N_{ij}} A_{ij}^m = \Omega \ \ and \ \ Pr(A_{ij}^m) \ge 0 \ \ where \ \ 1 \le n \ne m \le N_{ij}.$$

If the first bus that departs from node v_i at time μ_i is m, then $Pr(A_{ij}^m) = 1, Pr(A_{ij}^n) = 0$, where $1 \le n \ne m \le N_{ij}$, μ_i is the initial failure time of arc $\langle v_i, v_j \rangle$.

Assuming that we have already known $Pr(A_{ij}^n)$, where $1 \le n \le N_{ij}$, the time that bus n arrives at v_j satisfies $t \sim \varphi_n(t)$, where $t_{ij}^{n-1} < \mu_i \le t_{ij}^n$. The arrival time of bus m which departs from v_j and arrives at v_k satisfies $t \sim \varphi_m(t)$, where $t_{jk}^{m-1} < \mu_j \le t_{jk}^m$, then $Pr(A_{jk}^m \mid A_{ij}^n) = \int_{t_{jk}^{m-1}}^{t_{jk}^m} \varphi_n(t)dt$. According to the total probability formula, we can get

$$Pr(A_{jk}^m) = \sum_{n=1}^{N_{ij}} Pr(A_{ij}^n) \cdot Pr(A_{jk}^m \mid A_{ij}^n) = \sum_{n=1}^{N_{ij}} Pr(A_{ij}^n) \cdot \int_{t_{jk}^{m-1}}^{t_{jk}^m} \varphi_n(t)dt.$$

Now that, we have known path $P = \{v_0, v_1, \cdots, v_{i-1}, v_i, v_{i+1}, \cdots, v_{h-1}, v_h\}$ and its initial failure time μ_0, then the calculation method of the expected failure time $ET(P)$ and the expected life $EL(P)$ as follows:

$$\begin{cases} Pr(A_{01}^m) = 1, & if \ t_{01}^{m-1} < \mu_0 \le t_{01}^m, \\ Pr(A_{01}^m) = 0, & else \\ \cdots \ \ \cdots \ \ \cdots \ \ \cdots \\ Pr(A_{i(i+1)}^m) = \sum_{n=1}^{N_{(i-1)i}} Pr(A_{(i-1)i}^n) \cdot \int_{t_{i(i+1)}^{m-1}}^{t_{i(i+1)}^m} \varphi_n(t)dt \\ \cdots \ \ \cdots \ \ \cdots \ \ \cdots \\ ET(P_{1h}) = \sum_{m=1}^{N_{(h-1)h}} Pr(A_{(h-1)h}^m) \cdot \int t \cdot \varphi_m(t)dt \\ EL(P_{1h}) = ET(P_{1h}) - \mu_0 \end{cases}$$

5 *RELSP* Algorithm

If one path reached node v_i has the reliability priority relative to another, then the consistent reliability property insures that the latter cannot become one part of the optimal path. According to the dominance discriminance based on reliability theory, *RELSP* algorithm deletes the sub-paths without reliability priority. If there are lots of paths without reliability priority, then the algorithm will not use bigger storage space and computing time.

Input: an expanded stochastic and time-dependent network, which has consistent reliability property; source node s; destination node d; initial failure time μ_0; a non-increasing utility function $u(x)$.

Output: the optimal path from source node s to destination node d.

Data structures:

open: the set of expanded nodes. Let each expanded node only map to one path. These expanded nodes are represented by the numbers assigned to each node. Each node can map to multiple expanded nodes. Sort the elements in *open* according to utility function $u(x)$.

$SET(v) = \{w \mid \langle v, w\rangle \in A\}$represents the set of all nodes adjacent to node v.

DP_v saves all paths with reliability priority from source node s to node v, where $v \in V$.

Algorithm:

Step1: Initializes the expanded node that source node s is corresponding to, and put the expanded node into *open*;

Step2: If *open* is non-empty, goto Step3; otherwise, goto Step9;

Step3: Take out the element q whose value of the utility function is maximal from *open*, and let the last node that q is corresponding to be v, if v is destination node d, goto Step9; otherwise, goto Step4;

Step4: Review all the arc $\langle v, w\rangle$ that is derivate from node v, and get the set $SET(v)$ of nodes adjacent to v;

Step5: If $SET(v)$ is non-empty, then take out a node w from $SET(v)$, and expand the path corresponding to q to w, and get a new expanded node *num*, goto step6; otherwise, goto step2;

Step6: If there is a path has the reliability priority relative to the path that *num* is corresponding to, then the algorithm do nothing, goto Step5; otherwise, put the path that *num* is corresponding to into DP_w, goto Step7;

Step7: Calculate the expected failure time and expected life that *num* is corresponding to;

Step8: According to utility function u insert *num* into the appropriate place in *open*; goto step5;

Step9: Output the optimal path from source node s to destination node d, the algorithm is end.

Theorem 4. If the input network of the *RELSP* algorithm has consistent reliability property, and the utility function $u(x)$ is non-increasing, then the *RELSP* algorithm will find the optimal path from source node s to destination node d.

Proof: During the iteration, the algorithm divided path spaces having been searched into two parts: one part is the space of deleted paths, and let the set of nodes that the space contains be *NDP*; the other part is the space of saved paths, and let the set of nodes that the space contains be *DP*. But, there still exists path space having not been searched when the algorithm is end, and let the set of nodes that the space contains be *NSP*. Thus, V is divided into three subclasses, the relationship among them is shown by Fig. 2. Obviously, $\{s, d\} \in DP \cap NDP \cap NSP$.

The terminative condition of the algorithm is that *open* is empty or searching d. The first condition means that there are no path from s to d at all; We merely prove that the second condition means that we have found the optimal path from s to d.

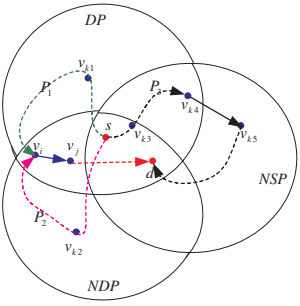


Fig. 2. The Validity of RELSP algorithm

Let P_1 be the sub-path passing through $v_{k1} \in DP - NDP$ and reaching $v_i \in DP \cap NDP$, P_2 is the sub-path passing through $v_{k2} \in NDP - DP$ and reaching $v_i \in DP \cap NDP$, and $P_1 \leq_{ad} P_2$. During the iteration, the algorithm deleted sub-path P_2. Then we prove the validity of the deleting operation as follows. Let the node set adjacent to v_i be $SET(v_i) = \{v_j \mid \langle v_i, v_j \rangle \in A\}$. For $\forall v_j \in SET(v_i)$, expand P_1 and P_2 respectively, thus we can get P_1' and P_2'. Because $P_1 \leq_{ad} P_2$, we can conclude $P_1' \leq_{ad} P_2'$ according to Theorem 3. Therefore, P_2 will never become the sub-path of the optimal path from s to d, so we may delete it.

In addition, the paths not being searched when the algorithm is end will not become the optimal path. This because that if we want to search these paths, we can carry our point only by expanding the sub-paths that are corresponding to those residual elements in *open*. For example, as Fig. 2 is shown, let $P_3' = \{s, \cdots, v_{k3}, \cdots, v_{k4}, v_{k5}, \cdots, d\}$ be certain path having still not been searched when algorithm is end, if we want to search P_3', then we must expand $P_3 = \{s, \cdots, v_{k3}, v_{k4}\}$ to v_{k5}, where P_3 is the path corresponding to certain element *num* in *open*, $\cdots$, continue to expand till *d*. However, in *open*, *num* is behind the element corresponding to the path just searched, i.e.the value of utility function of P_3 is not bigger than that of the path just searched. We get P_3' by expanding P_3, so the value of utility function of P_3' is smaller than that of P_3. Thereby, the value of utility function of P_3' must not be bigger than that of the path just searched, i.e. it is impossible for P_3' to become the optimal path. As well as, during the course of iterative, the algorithm takes out the optimal element from *open* every time. Therefore, the path that has just been found must be the optimal path from s to d. Proof is end.

Finally, we show an application example of the *RELSP* algorithm in [10], and make an comparison with the *PFS-Dominance* algorithm of Wellman (see [6]), and our *RELSP* algorithm is proved very efficient.

6 Conclusions

Compared with traditional deterministic networks, stochastic and time-dependent networks have more applications, in particular, in the fields such as wireless communications, intelligent transportation systems, and the Internet. All these new network applications have stochastic and time-dependent properties. Therefore, research on these stochastic and time-dependent networks is very important. Besides transportation networks, our study results should be applicable to other similar networks, such as computer networks and communication networks. Note that how to determine reliability priority is not an essential and necessary condition but a necessary condition. If an essential and necessary condition can be found, the search space can be further reduced.

References

1. J.A. Bottom, Anticipatory Consistent Route Guidance, Ph.D.thesis, Massachusetts Institute of Technology, Step, 2000.
2. Song Gao; Chabini, I. Optimal routing policy problems in stochastic time-dependent networks part I: framework and taxonomy. Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on, 2002.Page(s): 549–554.
3. Christos Alexopoulos, "State Space Partitioning Methods for Stochastic Shortest Path Problems," Networks, vol. 30, no. 1, 1997, pp. 9–21.
4. Tan Guozhen, Gao Wen. Shortest path algorithm in time-dependent networks. Chinese J.Computers, 2002, 25(2) 165–172 (in Chinese).
5. R.W. Hall, "The fastest path through a network with random time-dependent travel times", Transportation Science, vol.20, no.3, pp.91–101, 1986.
6. Michael P. Wellman, Matthew Ford, and Kenneth Larson, "Path Planning under Time-Dependent Uncertainty," Proc. Eleventh Conf. On Uncertainty in Artificial Intelligence, Aug. 1995, pp. 532–539.
7. Justin Boyan and Michael Mitzenmacher.Improved Results for Route Planning in Stochastic Transportation Network. SODA. 2001.
8. D.Stoyan. Comparison Methods for Queues and Other Stochastic Models. John Wiley and Sons, New York, 1983.
9. S.Ross. Stochastic Process. John Wiley and Sons, New York, 1983.
10. TAN Guo-zhen and XIA Xiang-fu. Algorithms for routing in stochastic and time-dependent networks based on reliability theory. Research report of Dept. of Computer Science and Engineering, Dalian University of Technology. No 200201.

Using a GA Adaptor in Multi-applications

Cheng Wei Hang and Yen Cheung

School of Business Systems,
Monash University, VIC 3800, Australia

Abstract. The topic discussed in this paper is a Genetic Algorithm (GA) for multi-applications without any user involvement in the process of the problem design. This user-friendly GA can be used as part of an intelligent tool for problem solving. The main idea of the design is to build a GA based on an adaptor, which encodes problems into a standardized format. This standardized format is then presented to the GA. Using this method; we can separate the GA from any particular application and ensures that it becomes a reusable tool. An extension of this concept is that we can build a number of format adaptors so that it can tackle a wider range of problems. We tested the GA using a number of combinatorial optimisation problems, particularly the Traveling Salesman Problem (TSP). The results indicated that the adaptor helped to improve the overall performance of the GA.

1 Introduction

Research in GA (genetic algorithm) theory and its applications are still being explored in recent years. Efforts in improving the performance of GAs and tackling problems in new applications have been found in numerous publications, especially applications in solving optimization problems. In applying a genetic algorithm (GA) for a particular application, it involves: identification of the object(s), representation of the problem, design of a GA, and interpretation of the search result(s) to the solution. In this process, the design of a GA strongly relies on the particular problem. Different GAs will use different data structures and hence different input formats. The research work discussed in this paper is an attempt to separate the design of a GA from any particular application. A reusable adaptor to the GA is needed. The pre-processor is able to adapt different types of problems before they are presented to the GA.

In achieving this purpose, the idea is to build a GA that associates with a predefined representation format(s). For a given application, the work is to encode the problem into the adaptor(s), and the GA is then able to return search result(s). The overall concept of this approach is as shown in Figure 1.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 839–848, 2003.
© Springer-Verlag Berlin Heidelberg 2003

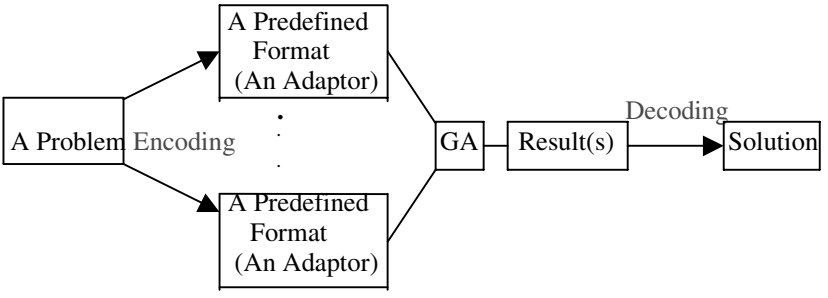


Fig. 1. Adaptors and the GA

Various forms of GA tools have been developed, but they are mostly for specific applications. Very little work can be found in building GA tools for general-purpose use. Similar recent work of this attempt can be found in (Reddy, Jain, and Pius, 2002). The authors use Borland C++ 5.0 to build a GA tool-package. In this package, it contains a set of tools that use various forms of genetic algorithms including both forms of "Binary Genetic Algorithms" and "Continuous Parameter Genetic Algorithms" (Gerez, 1999). Through a Graphical User Interface (GUI), it is able to provide the solutions for two different types of problems: Traveling Salesman Problem (TSP) and locating an Emergency Response Unit (ERU). Another report in (Walsh, 1999) discussed the development of level GA programs in functional style.

2 A GA Tool Design

2.1 A Form of Representation

We define a sequential symbol as:

$$S_1S_2S_3 \ldots S_j \text{ , } j \text{ is a finite number;}$$

S_i (i = 1, 2, 3, …, j) can be a kind of number (a binary number, a decimal number, …,); a kind of symbol (an alphabet, an image, …,); even it can be a data structure.

We define K as a knowledge that is contained in a sequential symbol of $S_1S_2S_3 \ldots S_j$ (j is a finite number). It can be represented as follows:

$$S_1S_2S_3 \ldots S_j \sim K$$

For examples, in the traveling salesman problem (TSP), assume that there are ten cities from 0 to 9 and the routing 0->1->2->…->9->0 is represented in the sequential number of 012…9 and its distance is assumed to be 100. We can say the knowledge of 100 is contained in the sequential symbol of 012…9. In a stock portfolio, $Stock_1Stock_2 \ldots Stock_j$ represent stocks holding; EV(r) represent total return value expected in a certain time period with risk r. The relationship can represent as: $Stock_1Stock_2 \ldots Stock_j \sim EV(r)$.

2.2 A GA Design

Domain Space

Base on the notation above, we define a domain space ***P*** that is formed by the elements that have following format:

$$S_1^{(i)}S_2^{(i)}S_3^{(i)} \ldots S_j^{(i)} \quad (1)$$

j is a finite number; and i (= 1,2, …) is the number of elements in ***P***, and define a solution space ***S*** which is formed by the elements that have following format:

$$S_1^{(i)}S_2^{(i)}S_3^{(i)} \ldots S_j^{(i)} \sim K^{(i)} \quad (2)$$

$K^{(i)}$ represents that *i*th element contains knowledge. If an element contains no knowledge, we consider it as having no value since it does not present any possible solution and there is no learning opportunity. The relationship between ***P*** and ***S*** has:

$$\boldsymbol{P} \supseteq \boldsymbol{S}$$

For examples, in TSP, we have ***P*** = ***S*** (Since all combinations represent possible solution.). With a stock portfolio, if any stock has its expected return value with risk in a certain time period, we may obtain the total expected return value for that portfolio; so in this case, we have ***P*** = ***S***. Otherwise, we have ***P*** ⊃ ***S***. In strategic pricing problem (Cheung and Hang, 2002), we have ***P*** ⊃ ***S***.

We use the format in expression (2) as an adaptor of the GA tool that we will discuss. So the GA tool should be able to solve the problems that are encoded in this format. In other word, the possible solution(s) of these problems can be represented in form of expression (2) or can be interpreted from it.

Obviously, the GA tool only searches solution(s) from ***S***. The solution space ***S*** can be associated with an intelligent database or case-base reasoning system that it is able to provide the information required by the GA tool. The meaning of an intelligent database or case-base reasoning system is not a simple data system. For example, in TSP, the database may only store the cities and the distances between each city. If using the GA tool to solve the problem, the database must be able to provide the length of any particular routing.

Chromosomes (Set of Population)

The design of individuals in the population is structured as in (Cheung and Hinde, 2000):

Gene [n]	Phenocell

The Gene[1] [n] and Phenocell[2] have following structures:

Gene [n]:

Operator	gene value	gene value	…

Phenocell:

$S_1^{(i)}$	$S_2^{(i)}$	…	$S_j^{(i)}$

The gene value takes a random from the phenocell. n is the gene value size. This will vary between different systems. Crossover is operated over the gene value between individuals which be discussed in the following section. In the GA tool, the value of n cannot be an option since we do not know what problem the tool will be used for. It also cannot be chosen as a constant, because this will reduce the randomness in search process, particularly if the value of j is much larger than the value of n. We choose n as:

$$n = \text{rand}(\max(1, (j*10^{-1}/2)), \max(1, j*10^{-1}))$$

In this way we can separate n from a particular problem; at same time it is able to keep sufficient randomness in the search process.

The operator in the gene structure is a specially designed genetic operator named as the parasitic operator (Cheung and Hinde, 2000) in which along with above structure of individuals, it adds further randomness in the search process to increase the search space.

Genetic Operators

Crossover occurs when the gene values have been exchanged between individuals. For example, if there are two individuals as follow:

Individual1:

operator	gene value	…	$S_1^{(i)}$	$S_2^{(i)}$	…	$S_1^{(i)}$

Individual2:

operator	gene value	…	$S_1^{(n)}$	$S_2^{(n)}$	…	$S_m^{(n)}$

the exchange position can be as follows:

$$\text{position} = \text{rand}(1, \min(j, m)).$$

---

[1] A gene is a stretch of DNA being located in a cell's nucleus.

[2] Phenocell is the collective physical characteristics and properties that an organism develops through the expression of its genes.

This guarantees that an exchange position will be found between two individuals. Once the crossover is completed, the gene value will be inserted back into the phenocell. Thus two offspring are formed. Before they are inserted into the population, the mutation operator will check if any individual has the opportunity to be mutated. The population uses an overlap method over the generations, and the individual is allowed to have different lengths. This method provides an opportunity to adapt the GA to a wider range of problems.

As mentioned the parasitic is special genetic operator. It uses six pre-defined values which affects the crossover and mutation operations on the phenocell. The purpose of this is also to increase randomness in the search space.

Fitness Function

Fitness function is used to measure each individual during the evolution process. It often works together with object function. In a GA tool, the fitness is designed as an input function as defined by the user. The method described in (Walsh, 1999) can be used in the tool where the fitness evaluation scheme assigns fitness value through building a population based on input arguments (such as $K^{(i)}$ as the input arguments).

Parameters

The parameters in GA work in a complex way. Their settings affect the performance in various problems. We attempt to build a GA tool with a high capability of adaptation for problems that is at the same time, user-friendly.

The optimized setting of parameters is unlikely to be found for a particular application; since a group setting of parameters must be made before applying a GA. Normally we use many different settings to test the results and choose the best one. This method can be automated in the GA tool without needing to input parameters' setting. The recent work in the design of parameter-less GA can be found in (Harik & Lobo, 1999). The authors described a method to build a user-friendly GA without asking the user to choose parameters setting and suggested doubling the size of the population in each test based on its previous test.

The optimized setting of parameters is unlikely to be found for a particular application; since a group setting of parameters must be made before applying a GA. Normally we use many different settings to test the results and choose the best one. This method can be automated in the GA tool without needing to input parameters' setting. The recent work in the design of parameter-less GA can be found in (Harik & Lobo, 1999). The authors described a method to build a user-friendly GA without asking the user to choose parameters setting and suggested doubling the size of the population in each test based on its previous test.

Normally, increasing the population size will improve the GA performance. We use the population size as a major parameter in testing results. The tool starts with a small population size:

$$p^{(1)} = j * 10^{-1}$$

j is the length of phonecell, also $p^{(1)} \geq 4$ (otherwise just select $p^{(1)} = 4$ to start.). The fitness value of $p^{(1)}$ is $f^{(1)}$ ($f^{(1)}$ is the value of last remaining population when the algorithm is terminated.). The second test is doubling the population size:

$$p^{(2)} = 2 * p^{(1)},$$

and the fitness value is $f^{(2)}$. For the *i*th test, we use population size:

$$p^{(i)} = (1 + \frac{|f^{(i-1)} - f^{(i-2)}|}{|f^{(2)} - f^{(1)}|}) * p^{(i-1)}.$$

From the above expression, when $|f^{(i-1)} - f^{(i-2)}| \rightarrow 0$, the population size used in the each test becomes stabilized. This means that: when the GA performance is getting better, it is not necessary to increase population size any further for improved performance.

For the genetic operators' probability rates, with a certain population size, we do not know what rates will be best for its performance. However too low probability rates may make the algorithm to converge in local maxima. It also may cause premature convergence when the population size is relatively small. Since the tool uses the population size as a major parameter in improving its performance and for simplicity reasons, the operations rates is randomly chosen from the range of 0.5 to 0.9.

Condition of Terminators

The GA tool is terminated by the condition of convergence that is specified by the user. In many cases, the users are not able to provide such a condition because they need to see the test results before making any judgments. So two terminators are needed in the tool. Firstly, how many times do we need to test? Secondly, for a given population size, how many generations are needed to terminate the algorithm?

The only way to obtain the judgment from users is to output the results each time after the test and get feedback from users to determine if further tests are needed. In the second question, for a given population size, it is unknown how many generations should run to reach the convergence. Since the tool uses a multi-test method to find a better solution, we can choose the number of generations as:

$$5 * p^{(i)}$$

where $p^{(i)}$ is population size of *i*th test. From our experiments, a factor of 5 seems to be reasonable.

3 Discussions

3.1 Search Space

The maximum population space of problems, which can be tackled by the tool, is j^j (j is the length of phonecell.). For some problems, their solution space may differ from population space as mentioned, since some individuals may not represent any possible solution. Even the initial population is chosen from solution space, because the method used in the evolution process reproduces new individuals that may be located outside the solution space as shown in Figure 2.

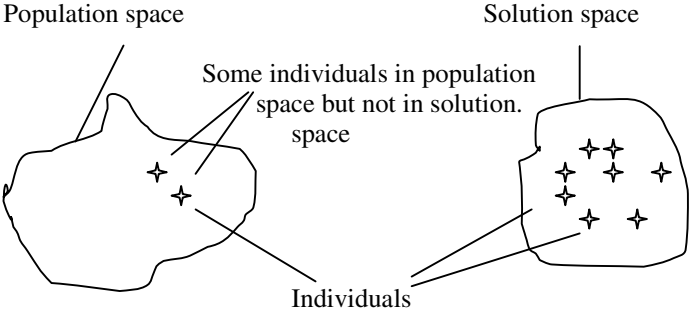


Fig. 2. The Relationship of Population and Solution Space

There are two methods that can solve this problem. Firstly, we can build a function to select an individual from the solution space and replace that individual that is locates outside the solution space. So the evolution process can be guaranteed to converge at solution space. Secondly, we can assign a special value (for example, the value represents the worst fitness.) to the individual that is out of the solution space and use the selection process to eliminate it from the population in every generation.

3.2 Constraints

Constraints of a problem set the limit of possible solutions. Assuming constraints are a set of $\{\mathbf{A}_\alpha\}$, α is a constraints number; solution space is **S**. After imposing constraints to the solution space, we have a new solution space **S'** as:

$$\mathbf{S'} = \mathbf{S} - (\bigsqcup_{\alpha} \mathbf{A}_\alpha)$$

Actually, the constraints of a problem reduce the solution space size. The tool handles constrains during the formation of the solution space, and the GA does not need to deal with. It only needs to associate with the solution space of **S'**.

4 Experiments

Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is most focused in combinational optimization. We use the tool to tackle this problem. In our TSP, there are 318 cities; assume the distance d_{ij} is from city(i) to city(j), we have $d_{ij} = d_{ji}$. Object function is:

$$\text{Object} = \min\Big(\sum_{\text{all individuals}} d_{ij}\Big),$$

and the fitness function is:

$$\text{fitness} = (\text{Number of individuals}) * \max\Big(\sum_{\text{all individuals}} d_{ij}\Big) - \sum d_{ij}$$

We can easily represent the TSP in the adaptor as:

$$\text{City(start)>City(next)>…>City(end)} \sim \sum d_{ij}$$

Genetic operators' rates are randomly selected from 0.5 to 0.9.

Test 1

Parameters setting output:

Mutation rate = [0.8]
Crossover probability = [0.7]
Parasite rate = [0.6]
Population = [32]
Number generations = [160]

Object function return: 48174

Fitness value in last remaining population: 1109306

Test 2

Parameters setting output:

Mutation rate = [0.6]
Crossover probability = [0.9]
Parasite rate = [0.7]
Population = [64]
Number generations = [320]

Object function return: 48174

Fitness value in last remaining population: 1247119

Test 3

Parameters setting output:

Mutation rate = [0.7]
Crossover probability = [0.9]
Parasite rate = [0.7]
Population = [128]
Number generations = [640]

Object function return: 48174

Fitness value in last remaining population: 1104046

Test 4

Parameters setting output:

Mutation rate = [0.5]
Crossover probability = [0.7]
Parasite rate = [0.9]
Population = [133]
Number generations = [665]

Object function return: 48174

Fitness value in last remaining population: 1180242

Test 5

Parameters setting output:

Mutation rate = [0.6]
Crossover probability = [0.9]
Parasite rate = [0.9]
Population = [207]
Number generations = [1035]

Object function return: 48174

Fitness value in last remaining population: 1124406

Test 6

Parameters setting output:

Mutation rate = [0.8]
Crossover probability = [0.8]
Parasite rate = [0.7]
Population = [290]
Number generations = [1450]

Object function return: 48124

Fitness value in last remaining population: 1099882

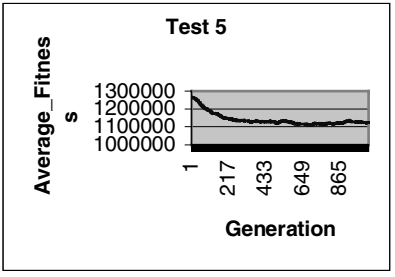


Fig. 3. Test 5

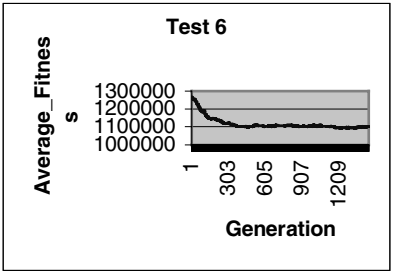


Fig. 4. Test 6

From Figure 4, we can see that convergence of the tool is very stable and the result has improved from previous tests. So the tool will stop further tests from this point.

5 Conclusion and Recommendations for Further Work

An attempt to build a GA tool for different types of problems is achievable through the use of an adaptor. The end-user or AI agent can use the tool in their applications. Initial experiments have demonstrated that the tool can find solutions for problems such as the TSP. At the same time, other types of problems can be encoded using the adaptor described in this paper. We can use the tool in many applications as long as the problems can be encoded into the adaptor. Other format adaptors can be developed to tackle a wider range of problems. We can also develop a technique-set that associates with a GA such as search with bias. An advantage of this design is that it is easy to include new techniques without changing the GA design. Test results obtained have shown that the adaptor helps to improve the overall performance of the GA. Further research work involve testing the tool, in particular the adaptor with a wider set of problems as well as investigations into other format adaptors.

References

Reddy G.N., Jain M., Pius L. (2002) *The Genetic Algorithms – A* Tutorial*,* Artificial and Computational Intelligence, Proceedings of the IASTED International Conference, September 25-27, 2002, Tokyo, Japan. P. 90–95.

Reddy G.N., Jain M., Pius L. (2002) *The Genetic Algorithms – A* Tutorial*,* Artificial and Computational Intelligence, Proceedings of the IASTED International Conference, September 25-27, 2002, Tokyo, Japan. P. 90–95.

Cheung,Y., Hang,C. (2002) *Strategic Pricing using Genetic Algorithm,* Artificial and Computational Intelligence, Proceedings of the IASTED International Conference, September 25-27, 2002, Tokyo, Japan. P. 280–285.

Cheung Y., Hind C. (2000) *Economic Modelling with A General Purpose GA*, Proc. Int. Conference Advances in Intelligent Systems: Theory and Applications, AISTA '2000, 2-4 Feb 2000, IEEE ACT Section, Canberra, Australia, p. 96–101.

Walsh P. (1999) *Afunctional Style and Fitness Evaluation Scheme for Inducting High Level Programs,* GECCO-99, July 13–17,1999, San Francisco, California, Volume 2, p 1211–1216.

Harik G., Lobo F. (1999) *A Parameter-Less Genetic Algorithm,* GECCO-99, July 13-17,1999, San Francisco, California, Volume 1, p. 258–265.

Automated Debugging in Java Programs Using HDM*

Hoon-Joon Kouh and Weon-Hee Yoo

School of Computer Science and Engineering, Inha University
#253 Yong-Hyun Dong, NamKu, Incheon, KOREA
hjkouh@hanmail.net, whyoo@inha.ac.kr

Abstract. Debugging has always been a costly part of software development. At the present time, Java programs locate logical errors using traditional debugging methods. Unfortunately, the traditional methods are often inadequate for the task of isolating specific program errors. This paper proposes a method to use HDM(Hybrid Debugging Method) for locating logical errors in Java programs. This method is a debugging technique that combines an algorithmic debugging method with a traditional step-wise debugging method. This approach can reduce the number of user's interaction. In this paper we describe the issues for HDM in the Java programs. Also, we walk through an example.

1 Introduction

Nowadays programs are becoming more large and complex than the past programs. Specially, the programs have many functions/methods and the number of statements in the functions/methods is increasing more and more. So debugging has always been a costly part of software development[4]. Users may spend much time in debugging the various programs using the traditional sequential debugging methods.

Techniques for debugging Java programs, until now, have been traditional debugging methods that are used at procedural-oriented programs. But debugging the object-oriented programs are not the same as procedural-oriented programs. Unfortunately, these traditional methods are often inadequate for the task of isolating specific program errors. So, many researches are in progress for efficiently debugging Java programs. These approaches tried to apply the existing techniques to Java programs. Recent works for efficiently debugging Java programs are slicing[3,8], query-based debugger[9], design by contract methodology[2] and model-based diagnosis[10]. The focus of these techniques is to reduce the user's interaction to the minimum.

This paper proposes a method to use HDM(Hybrid Debugging Method)[6] for locating logical errors in Java programs. HDM is a debugging technique that combines an algorithmic debugging method[1,5,7,11] with a step-wise debugging method. An algorithmic debugging method can easily locate functions/methods including a logical error. And a step-wise debugging method can easily locate statements including a

* This work was support by INHA UNIVERSITY Research Grant. (INHA-22000).

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 849–858, 2003.
© Springer-Verlag Berlin Heidelberg 2003

logical error in the function/method. So HDM locates an error included in the functions with an algorithmic debugging method and locates the statements with an error with a step-wise debugging method in the erroneous function. The HDM can improve the drawback of an algorithmic debugging method and a step-wise debugging method. In this paper we describe the issues for parameter value presentation, constructor, method, Java API and member variables for HDM in Java programs. Also, we walk through an example.

The rest of this paper is organized as follows: In Section 2, we describe traditional debugging methods in Java and an algorithmic debugging method. In Section 3, we describe the functional structure of HDM and the algorithm. In Section 4, we present issues for HDM in Java programs. We walk through an example in Section 5, and conclude in Section 6.

2 Background

In this section, we describe the problem of the traditional debugging methods in Java programs and the concept of an algorithmic debugging method.

2.1 Traditional Debugging Methods in Java Programs

There are two ways in traditional methods for debugging logical errors in a Java program. First, the users can directly analyze a source program or insert the screen output instructions like "*System.out.println*" in the suspected location. But these methods are not easy to locate the logical errors because it is difficult for them to correctly anticipate the error location. Second, there is a step-wise debugging method. Users can sequentially debug a Java program with instructions like "*step-over*", "*step-into*", "*go*" and "*break-points*", but the method requires much time and much labor from the user because the user should execute all statements in the worst case.

2.2 Concept of an Algorithmic Debugging Method

An algorithmic debugging method is a semi-automated debugging technique that builds an execution tree from a source program and locates the logical errors included in the program from an execution tree.

The execution tree has the trace information such as function name or input/output parameter values etc. And the tree is built from the root to the leaves and from left to right. A called function becomes a child node of the calling function in the execution tree. And then the algorithmic debugging method traverses an execution tree using the top-down method and interacts with the user by asking about the expected behavior at the each node. The user can answer "correct" or "incorrect". And an algorithmic debugging automatically locates the erroneous functions from the information. The "correct" answer means that the current and child functions don't have any logical error. The "incorrect" answer means that the current or child function has one or more logical errors. So if the debugging process stops the traversal, some function *f* may have

some errors in two cases: In one case, the function *f* contains no function call. In the other case, all function called in the function *f* satisfy user's expectation.

3 HDM (Hybrid Debugging Method)

HDM is a debugging technique that combines an algorithmic debugging method with a step-wise debugging method.

This method debugs as follows. First, HDM system translates a source program into a program transformation tree and builds an execution tree from the tree. Second, the debugging system locates a node(function) including an error using an algorithmic debugging method. Third, the HDM system locates a statement including an error with a step-wise debugging method in the erroneous function through an editor view(see Fig. 1). Fourth, the user rebuilds an execution tree again and can debug from the root node of the program to a erroneous node or a last node. Finally, HDM iterates from first step to fourth step until there are no errors in a Java program.

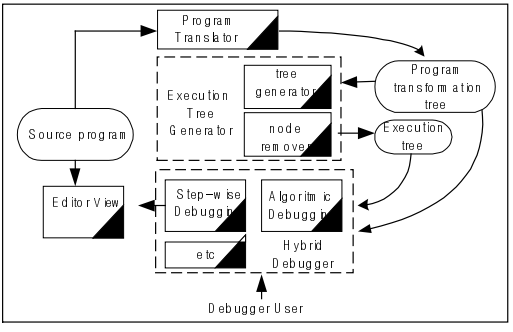


Fig. 1. The functional structure of HDM system

4 Issues for Java Programs

In this section, we describe issues for debugging Java programs using HDM. The issues are parameter values, constructor, method, Java API and member variables. We build an execution tree with the calling relationship between methods(including constructor) instead of functions in Java programs.

4.1 Parameter Value Presentation

An object is a class instance and receives information through the parameters of a constructor when the object is created by a class instance creation expression. Also an object does behave the desirable jobs using methods. At this time the method sends and receives information through many parameters. The primitive data types of these parameters are described as follows.

- Boolean variables are True or False at the input/output variables.
- For integer variables (byte, short, integer, long), the input variables are represented as binary digit, octal digit, decimal digit, hexadecimal digit and the output variables are represented as all decimal digits.
- For float and double variables, the input variables are represented as decimal digit, exponentiation and the output variables are represented as decimal digits.
- Character variables are represented as 'character'.

Parameter values with reference data types are described as follows.

- String object variables are represented as 'strings'.
- Array object variables are represents as sequence of elements. For example, one dimensional array with n+1 elements is represented as $[a_0, a_1, a_2,\ldots, a_n]$. An element a_i is i+1 element of the array
- Class object variables are represented as pairs of variable and value about the member variables. For example, an object with member variables $x_1, x_2, x_3,\ldots, x_n$ and values $v_1, v_2, v_3, \ldots, v_n$ is represented as $(x_1=v_1, x_2=v_2, x_3=v_3,\ldots, x_n=v_n)$.

4.2 Constructor

A constructor is executed the only once when an object is created from class. And it is mostly used to initialize member variables.

To build an execution tree, we assume that the left-hand side variables of the assignment in the constructor are all output variables. Fig. 2 shows an example of a constructor.

```
class SimpleCal  {
     int num1 ,num2;
     public SimpleCal() {
          this(0,0);
     }
     public SimpleCal(int num1, int num2) {
          this.num1=num1;   this.num2=num2;
     }
}
```

Fig. 2. An example of constructor in class *SimpleCal*

The class *SimpleCal* has two constructors in Fig. 2. They are overloaded constructors. The first constructor has no parameters. And the second constructor has two parameters and assigns the parameter values to two member variables. The instructions that create objects *Obj1*, *Obj2* from *SimpleCal* is as follow.

```
SimpleCal Obj1 = new SimpleCal();
SimpleCal Obj2 = new SimpleCal(3, 4);
```

The process that creates object *Obj1,Obj2* for building an execution tree is as follows. The method "*this()*" calls a constructor that has two parameters, input values 0 and 0. And the output variables are all the left-hand side variables of the assignment.

Obj1.SimpleCal();
Obj1.SimpleCal(in:num1=0,num2=0 out:this.num1=0,this.num2=0);
Obj2.SimpleCal(in:num1=3,num2=4 out:this.num1=3,this.num2=4);

An execution tree that is built from these objects is shown in Fig. 3. The nodes of an execution tree are constructed from the information of constructor. The tree makes the hierarchical structure from main method according to the execution order. The HDM system doesn't build the node of no input/output value like a method *Obj1.SimpleCal()*. Two nodes have the same method name, but we can identify the method from the object names and the input/output parameters.

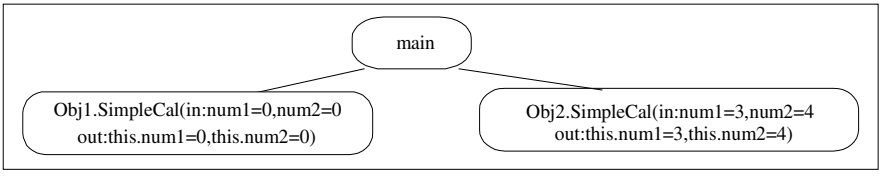


Fig. 3. An execution tree that is built from the constructor *SimpleCal*

4.3 Method

A method defines object's behavior in a class. We define two methods(*incre*, *addsum*) in a class *SimpleCal* (Fig. 4).

```
class SimpleCal  {
     int incre(int num) {
            num = num + 1;   return num;
     }
     int addsum(int a, int b)  {
            int c;
            a = incre(a);  b = incre(b);
            c = a + b;  return c;
     }
}
```

Fig. 4. An example of method in class *SimpleCal*

Suppose that an object Obj1 calls a method *addsum* with arguments 3, 4 and the result value is assigned to a variable *sum,* then we can express the program such as follows in Java program.

sum = Obj1.addsum(3,4);

A method *addsum* adds two integer values and a method *incre* increases a integer variable by one. The execution of these statements is as follow.

Obj1.addsum(in:a=3,b=4 out:c=9);
Obj1.incre(in:num=3 out:num=4);
Obj1.incre(in:num=4 out:num=5);

An execution tree built from these methods is shown in Fig. 5.

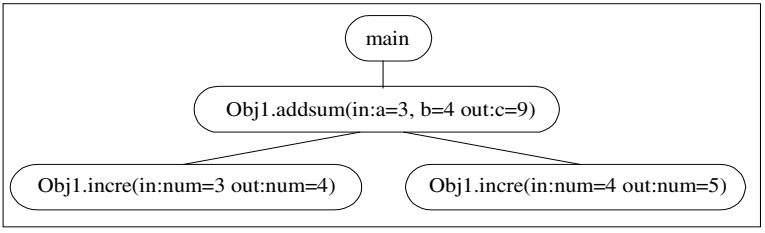


Fig. 5. An execution tree created from methods in a class *SimpleCal*

4.4 Java API

Java API has a lot of the built-in packages. Java has API that is libraries to provide much functions. Java programmers use the predefined software packages in Java API. The packages have a lot of classes and other packages. Also the classes have a great many primitives and library methods. They may reasonably be assumed to behave correctly. And then, debugging users should not be asked about a great many library methods included in classes. The HDM system doesn't build an execution tree with theses methods. Obviously, this will reduce the number of questions that are to be asked and the time taken to build the execution tree.

4.5 Member Variable

Java classifies member variables into an object variable, a class variable and a final variable. Also Java classifies an object variable into an object property variable and an object reference variable. These can be assigned or changed values in the methods of the class. So, changing of the variable's value becomes the important information to a user.

```
class Person {
  long idNum;
  static long personID=0;
  public Person() {
        idNum = personID;
        personID = personID + 1;
  }
}
class  Employee  {
  public static void main(String args[]) {
        Person p1 = new Person();
        Person p2 = new Person();
    }
}
```

Fig. 6. The example of member variables : idNum is an object property variable and personID is a class variable.

A class *Person* has an object property variable idNum, a class variable personID and a constructor *Person*. The class *Employee* instantiates objects p1 and p2 from the class *Person*. Whenever one object is created from the class of Fig. 6, personID is increased a value 1 in the constructor. And the value of the personID is assigned to idNum. But nodes of the class *Person* don't have parameters and only have member variables. So a user doesn't, generally, know that the values of member variables personID and idNum change in the constructor *Person*. We should add member variables to input/output information of each node at the execution tree as Fig. 7.

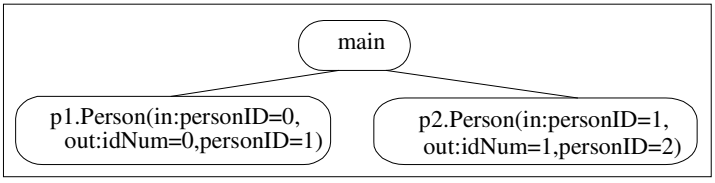


Fig. 7. The information about the definition and assignment of the member variable is added to each node of an execution tree

Also, if the variables are classified to an object property variable, an object reference variable and class variable in the nodes, the user would easily understands the constructor and the result. And so, we define the member variables as follows.

- If an object property variable is defined in the method/constructor, the variable uses a symbol OpvIn(object property input variable). And if an object property variable is used in the method/constructor, the variable uses a symbol OpvOut(object property output variable).
- If an object reference variable is defined in the method/constructor, the variable uses a symbol OrvIn(object reference input variable). And if an object reference variable is used in the method/constructor, the variable uses a symbol OrvOut(object reference output variable).
- If a class variable is defined in the method/constructor, the variable uses a symbol CvIn(class input variable). And If a class variable is used in the method/constructor, the variable uses a symbol CvOut(class output variable).

From the definition of the upper part, the execution tree is modified from Fig. 7 to Fig. 8. Then debugger user can get the correct debugging information from the execution tree.

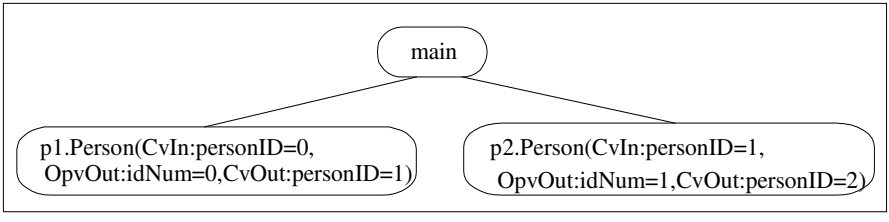


Fig. 8. The information about the kind of the member variables is added to each node of an execution tree

5 An Example of Applying HDM in Java Programs

Now, we can explain the HDM with an example. Fig. 9 shows a Java program with two classes(*SimpleCal*, *Calculation*). A class *SimpleCal* has two constructors and three methods. A class *Calculatio*n has one main method. We assume two cases for the debugging test.

- There is an error in the method *incre*. A statement "num=num-1" is a logical error. And then a statement "num=num+1" is correct.
- There is an error in the method *totalsum*. A statement "var2=var1-num" is a logical error. And then a statement "var3=var2+num" is correct in *totalsum*.

```
class SimpleCal  {
    int num;
    public SimpleCal()
    {    this(0);    }
 public SimpleCal(int num)  {
      this.num=num;
    }
 int incre(int num) {
   num = num - 1;
   return num;
 }
 int addsum(int a, int b)  {
   int c;
   a = incre(a);
   b = incre(b);
   c = a + b;
   return c;
}
```

```
    int totalsum(int a, int b) {
        int var1,var2;
        var1=addsum(a,b);
        var2= var1-num;
       return var2;
    }
   }
public class Calculation  {
public  static  void  main(String  args[])
{
     int a, b;
     SimpleCal Obj1 = new SimpleCal();
     a = Obj1.addsum(3,4);
     SimpleCal Obj2 = new SimpleCal(5);
     b = Obj2.totalsum(3,4);
     System.out.println(a+ "    " + b);
 }
}
```

Fig. 9. An example of a Java program

If we start to debug a Java program of Fig. 9, the HDM system builds an execution tree as Fig. 10. The user expects the result 9 and 14, but gets the result 5 and 0.

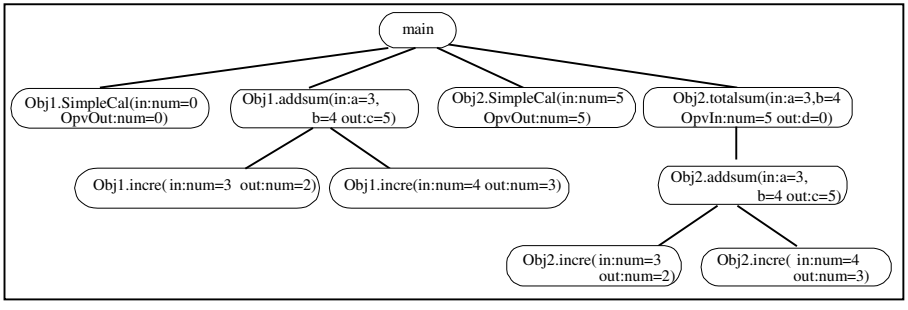


Fig. 10. An execution tree that is built from Fig. 9

The debugger user starts an algorithmic debugging at the HDM. The debugging process from the execution tree is as follows.

Obj1.SimpleCal(in:num=0 OpvOut:num=0)? $ correct
Obj1.addsum(in:a=3, b=4 out:c=5)? $ incorrect
Obj1.incre(in:num=3 out:num=2)? $ incorrect

HDM system locates an error in the method *incre* and shows the source program of the method *incre* in a text editor. The user can start a step-wise debugging and locate a statement including a logical error with "step-over", "go" and "break-points". The user can modify "num=num-1" to "num=num+1". A step-wise debugging stops when a user selects "stop" or the current method is returned. HDM rebuilds an execution tree and traverses an execution tree from the top node.

Obj1.addsum(in:a=3, b=4 out:c=9)? $ correct
Obj2.SimpleCal(in:num=5 OpvOut:num=5)? $ correct
Obj2.totalsum(in:a=3, b=4 OpvIn:num=5 out:var2=4)? $ incorrect
Obj2.addsum(in:a=3, b=4 out:c=9)? $ correct

HDM system locates an error in the method *totalsum* and shows the source program of the method *totalsum* in a text editor. A user can locate and modify a logical error, "var2=var1-num" to "var2=var1+num". Then HDM system restarts the program debugging and builds an execution tree again. When HDM system builds an execution tree again, he/she doesn't build the following nodes because they are the nodes that users answered in "correct" at the previous execution tree.

Obj1.SimpleCal(in:num=0 OpvOut:num=0);
Obj2.SimpleCal(in:num=5 OpvOut:num=5);

The user confirms that the program has no errors.

Obj1.totalsum(in:a=3,b=4 OpvIn:num=0 out:var2=9)? $correct
Obj2.totalsum(in:a=3,b=4 OpvIn:num=0 out var2=14)? $correct

There are no errors. A user has localized two errors with HDM in a Java program of Fig. 9. So HDM is a suitable technique for debugging Java programs.

6 Conclusion

In this paper we have suggested a HDM for debugging Java programs. The HDM is a debugging technique that combines a step-wise debugging method with an algorithmic debugging method. An algorithmic debugging method can easily locate functions/methods including a logical error. And a step-wise debugging method can easily locate statements including a logical error. So users can locate a method/constructor including a logical error with algorithmic debugging method and then locate a statement including a logical error with a step-wise debugging method in the erroneous method/constructor.

We proposed to build an execution tree with methods(including constructor) instead of functions for debugging a Java program. Also we defined the parameter values and solved the problem of the member variable for building an execution tree. And this paper has walk through an example that HDM is a suitable technique for debugging Java programs.

As a result, HDM reduced the number of the user's interaction than the traditional debugging methods in Java programs.

References

1. Z. Alexin, T. Gyim'othy and G. Kokai, IDT: Integrated System for Debugging and Testing Prolog Programs in Proceedings of the Fourth Symposium on Programming Languages and Software Tools, Hungary June 9-10, (1995) 312–323
2. M. Auguston, A language for debugging automation in: Proceedings of the 6th International Conference on Software Engineering and Knowledge Engineering, Jurmala, June 1994, Knowledge Systems Institute, pp. 108–115.
3. D. Bartetzko, C. Fischer, M. Moller and H. Wehrheim, Jass-Java with Assertions in Workshop on Runtime Verification, 2001. held in conjunction with the 13th Conference on Computer Aided Verification, CAV'01
4. Z. Chen and B. Xu, "Slicing Object-Oriented Java Programs," ACM SIGPLAN Notices, V.36(4):pp 33–40 April, 2001
5. P. Fritzson, M. Auguston and N. Shahmehri, "Using Assertions in Declarative and Operational Models of Automated Debugging," The Journal of Systems and Software 25, 1994, pp 223–239
6. P. Fritzson, N. Shahmehri, M. Kamkar, T. Gyimothy, "Generalized Algorithmic Debugging and Testing," ACM LOPLAS – *Letters of Programming Languages and Systems.* Vol. 1, No. 4, December 1992.
7. H J Kouh, W H Yoo, Hybrid Debugging Method: Algorithmic + Step-wise Debugging in Proceedings of the 2002 International Conference on Software Engineering Research and Practice, June 2002.
8. G. Kokai, L. Harmath, and T. Gyim'othy, Algorithmic Debugging and Testing of Prolog Programs in Proceedings of ICLP '97, The Fourteenth International Conference on Logic Programming, Eighth Workshop on Logic Programming Environments Leuven, Belgium, 8-12 July 1997, 14–21.
9. G. Kovacs, F. Magyar, and T. Gyimothy, Static Slicing of JAVA Programs in Research Group on Artificial Intelligence(RGAI), Hungarian Academy of Sciences, Jozsef Attila University, HUNGARY, December 1996
10. R. Lencevicius, On-the-fly Query-Based Debugging with Examples in Proceeding of the Fourth International Workshop on Automated and Algorithmic Debugging, AADEBUG '2000, Munich, Germany, August 2000.
11. C. Mateis, M. Stumptner and F. Wotawa, Locating bugs in Java Programs-first results of the Java Diagnosis Experiments(Jade) project in Proceedings of IEA/AIE, New Orleans, 2000, Springer-Verlag.
12. H. Nilsson and P. Fritzson. Algorithmic debugging for lazy functional languages in Proceedings of PLILP'92 – Symposium on Programming Language Implementation and Logic Programming, Leuven, Belgium August 1992. LNCS 631, Springer Verlag.

Pattern Based Approaches to Pre-processing Structured Text: A Newsfeed Example

Paul Bogg

University of Technology Sydney
plbogg@it.uts.edu.au

Abstract. Text documents presenting a structured format allow the reader the ability to quickly run their eye over the page and read information relevant to them. By presenting information in this manner, the author allows ease of information extraction by the reader. If the structure used throughout the document involves a pattern or set of patterns to describe the text, then if text pre-processing methods can identify the patterns involved, those methods can also extract the same text as that of the naked eye. This extraction of meaningful text can then be used for further text mining applications. This paper describes a text pre-processing program that identifies text patterns and extracts the appropriate text.

1 Introduction

Text mining is the process of identifying and extracting information and/or statistics from text documents that are of interest [2]. Before the mining activity begins on a document, the text is prepared so that the raw text presented to the text mining algorithm is precisely the interesting information that the mining process is seeking (and none that might skew the information incorrectly). This step is called the *pre-processing* stage [3].

The pre-processing stage is typically the most complex stage in a text (or data) mining exercise. Structured text documents contain identifiable patterns of text that may be extracted by a program for the purpose of pre-processing. If these patterns can be identified correctly, and consequently so too the interesting text, then we can speed up the whole text mining process significantly [5]. A technique to achieve this is described.

2 Process

To demonstrate the theory behind the pattern identification approach to pre-processing, we will use email newsfeeds as our base type of text document. An email newsfeed is a purely text based news document sent to a user to brief him or her on daily events around the world (ie, news). These newsfeeds are emailed to a user on a regular basis informing them of potentially useful information. Generally, most

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 859–867, 2003.
© Springer-Verlag Berlin Heidelberg 2003

internet news paper sites provide a free email newsfeed service. For example http://www.usatoday.com and http://www.smh.com.au.

An example of a newsfeed is shown below in Courier New format:
(extract take from the New York Times email newsfeed received on the 10th September 2001).

```
   NATIONAL
   ========================
   Black Families Resist Mississippi Land Push

   Mississippi's right to emminent domain is being
challenged by
   black property owners whose claims date back to
Reconstruction.
http://www.nytimes.com/2001/09/10/national/10LAND.html?to
daysheadlines

   -----

   Sacramento Shooting Suspects Dies in Gun Battle

   Joseph Ferguson, sought in the killing of 5 people in
Sacramento,
  reportedly shot himself  after a violent gun battle
with police officers  early today.
   http://www.nytimes.com/aponline/national/AP-
Sacramento-Shootings.html?todaysheadlines

   -----
```

Here, 2 articles of news content are described in the form of a heading, body and corresponding URL. We adopt the hypothesis that the pattern that is to be identified is: heading, body and URL. If this hypothesis is valid then we should be able to identify the common structure involved and so too extract the text information based on that pattern.

2.1 Classification

The first step in identifying patterns within a document is to classify certain sections of text. This is the primary initialisation step that any text mining program should perform if it is to extract information from a given document [4]. Classification of parts of the document may require some initial prior knowledge of possible document layout (depending on the type of document), but generally, for most purposes, domain experts are not needed.

Each unique part of the document which can be identified is called an 'Identifier'.

Given a sample chunk of text, we identify two base types of text identifiers: Single line text, and Multiple line text. Short of identifying words and concepts within the text (which belongs more to the mining stage – remember all we want is to pre-process), this is the highest level of abstraction that is necessary to consider. *Single line text* is a solitary line within the document neither preceded nor followed by another line of text (for example, a heading). *Multiple line text* is a block of 2 or more lines of consecutive text.

Using our sample newsfeed above, an example of single line text is:

```
Black Families Resist Mississippi Land Push
```

And an example of multiple line text is:

```
Mississippi's right to emminent domain is being challenged by
black property owners whose claims date back to reconstruction.
```

Furthermore, in line with our hypothesis that is specific to the domain of newsfeeds, we identify the third type of text that we require: the URL. This chunk of text will be a line with a URL pointing to a co-related news article that may be in a different format (for example HTML). In our sample newsfeed above, an example of a URL is:

```
http://www.nytimes.com/2001/09/10/national/10LAND.html?todaysheadlines
```

So now we have all three primary identifiers that can be used to classify patterns within the documents, and to identify the sections that we would like to keep (ie, patterns containing any or all of these identifiers).

To supplement these three primary identifiers, we introduce identifiers that describe the surrounding document sub sections. There are three of these secondary identifiers:

- Space – for lines within the document which contain only space and newline characters
- Potential Boundary – for lines which contain significant percentage of non-alphanumeric characters, they have the potential to be a boundary
- Unknown – for lines which there is can be no classification given

Together these six identifiers are sufficient to identify patterns in newsfeed documents as is shown below.

2.2 Identifying Patterns

If we look at the sample newsfeed, we intuitively see that there is some kind of pattern occurring, even if we cannot spot it. As the first step in the identification of the pattern, we illustrate the classification of the newsfeed in terms of the identifiers discussed in the previous section. Thus, we can now re-write the document in terms of these identifiers – the result of this operation is:

FREE_TEXT
POTENTIAL_BOUNDARY
FREE_TEXT
SPACE
MULTIPLE_TEXT
URL
SPACE
POTENTIAL_BOUNDARY
SPACE
FREE_TEXT
SPACE
MULTIPLE_TEXT
URL
SPACE
POTENTIAL_BOUNDARY

From this representation, the patterns are more obvious. Instinctively, we see that the main pattern we can use to extract the correct text is FREE_TEXT – SPACE – MULTIPLE_TEXT – URL. However, there is another less obvious case where by we can use the POTENTIAL_BOUNDARY as a kind of tag to identify when an extractable pattern may exist, and consequently attempt to parse the text between each one. We will discuss this later, but for now, we follow the initial approach.

2.3 Initial Approach

We start identifying main patterns through a process of first grouping sequential identifiers together into 2-tuples, and then removing any 2-tuple that does not occur more than once in the document. To further reduce the number of 2-tuples, we finally remove any 2-tuple that does not contain an identifier that has been flagged (we are looking for either FREE_TEXT or MULTIPLE_TEXT. URLs are supplementary). Thus, after the first run through our classified newsfeed the following 2-tuples remain:

FREE_TEXT – SPACE (occurs 2 times)
SPACE - MULTIPLE_TEXT (occurs 2 times)
MULTIPLE_TEXT – URL (occurs 2 times)

Once again, we can start to see the main patterns beginning to emerge. Using these new 2-tuples, the next step is to attempt to further ‘drill-down’ the pattern to come to a list of 3-tuples. Once this new list has been constructed, to improve the chance of

reaching a final dominant pattern, any 3-tuple that overlaps another 3-tuple is removed, including itself (with any overlaps resolved by the keeping the 3-tuple which occurs generally before the other. If a pattern begins to overlap itself, then we cease 'drilling-down' for that pattern). The following is an example of this phenomenon.

FREE_TEXT – SPACE – MULTIPLE_TEXT (occurs 2 times)
SPACE – MULTIPLE_TEXT – URL (occurs 2 times)
MULTIPLE_TEXT – URL – SPACE (occurs 2 times)

Is the list of 3-tuples that is obtained by 'drilling-down' the 2-tuple patterns. To identify potential patterns, any overlapping patterns are removed. This results in a more refined list:

FREE_TEXT – SPACE – MULTIPLE_TEXT (occurs 2 times)
MULTIPLE_TEXT – URL – SPACE (occurs 2 times)

If we continue this distillation of the patterns within the document, further drilling-down and refining, we will come to a point where we can identify a dominant pattern (or set of patterns) which we can use to extract meaningful text. A pattern is *generally dominant* when it either is repeated a number of times after being drilled-down, or is the solitary pattern remaining after refinement. Using our example above, post-refinement, the following pattern dominant is derived:

FREE_TEXT – SPACE – MULTIPLE_TEXT – URL – SPACE (occurs 2 times)

As you can see, this pattern mirrors (or very similarly resembles) the same one you instinctively viewed earlier on from the original email newsfeed. Using this pattern, the meaningful information (or "articles") is extracted and then passed to the text mining procedure.

2.4 Alternative Approach

As we mentioned in Section 2.2, there is an alternative approach to extracting information from the structured text. In our initial approach (described in Section 2.3), we attempted to deduce a primary pattern (or set of patterns) that contained the text we wanted to extract and pass to the text mining procedure. If that approach found a pattern, then we could use it as the basis for identifying an "article" of information.

As an alternative to that approach, we now utilise the properties of "boundaries" as flags for identifying where meaningful information occurs. Within a structured text, boundaries may exist between articles which identify when a new article commences and finishes. If we look at our original email newsfeed, we can identify several potential boundaries:

```
  NATIONAL
   ========================        ← Boundary
   Black Families Resist Mississippi Land Push

   Mississippi's right to emminent domain is being
challenged
   by black property owners whose claims date back to
   Reconstruction.

http://www.nytimes.com/2001/09/10/national/10LAND.html?to
daysheadlines

   -----                ← Boundary

   Sacramento Shooting Suspects Dies in Gun Battle

   Joseph Ferguson, sought in the killing of 5 people in
   Sacramento, reportedly shot himself  after a violent
gun
   battle with police officers  early today.
   http://www.nytimes.com/aponline/national/AP-
Sacramento-Shootings.html?todaysheadlines

   -----                ← Boundary
```

The boundaries are shown above with the italicised label "← *Boundary*". If we identify the main boundaries within the document, then by analysing the text between the boundaries for common patterns, we are also able to extract meaningful text.

3 Application and Results

By incorporating the two approaches (in section 2.3 and section 2.4), we have constructed a program that takes a structured text document, parses it, attempts to identify dominant patterns that capture meaningful text, and consequently parses the text into "articles" ready for use by the text mining procedure.

We sampled our program across 300 email newsfeeds from a variety of different sources from the USA, UK and Australia.

The results are shown in Table 1 below.

An *article* is anything that appears repetitively within a pattern and is extracted from the document by the pre-processing procedure. In this case, for email newsfeeds, an article consists primarily of the identifiers FREE_TEXT, MULTIPLE_TEXT, and/or URL. For example, for the sample emails received from boston.com, the information within each article extracted was typically FREE_TEXT and URL.

Table 1. Results obtained by the pre-processing program.

Newsfeed Subscription	Total Articles	Articles Identified	% Bad Articles	% Articles missed
boston.com	482	464	4.310344828	7.883817427
guardianunlimited.co.uk	269	240	7.083333333	17.10037175
usatoday.com – tech	252	273	7.692307692	0
usatoday.com – regular	230	230	0	0
age.com.au	816	827	5.07859734	3.799019608
afr.com.au	370	671	44.56035768	0
smh.com.au	205	251	21.9123506	4.390243902
ltnews_daily (smh.com.au)	218	285	23.50877193	0
nytimes.com.au	395	423	9.692671395	3.291139241
			Average Bad Articles	Average Articles Missed
	3237	3664	13.75985942	4.051621325

Total Articles – total number of proper articles identified manually from emails sent by this provider
Articles Identified – total number of articles identified by the program
%Bad Articles – unwanted articles which were extracted by the program
%Articles missed – wanted articles which were not extracted by the program

As the results in Table 1 show, the program was overall quite accurate in the number of articles extracted. Of all the articles extracted, only about 4% of the articles viewed as being wanted (in terms of meaningful text) were not identified by the program. One of the goals of the exercise was to explore the use of quite general patterns, giving leniency towards more generic patterns that picked up wanted articles rather than specific, refined patterns that were more accurate in removing bad articles. This is reflected in the Average Bad Articles statistic of nearly 14% - this means almost 14% of all articles found were not wanted.

In a closer inspection of these two statistics, much of the inaccuracy of this program could be attributed to a small handful of documents without proper structure. Occasionally, the odd newsfeed would not have a proper structure from which to extract a reasonable pattern and consequently the entire batch of articles extracted would be unwanted. Likewise, many of the bad articles found by the program were infact advertisements for differing products and were not placed in a distinguishable location within the email.

It was interesting to note that the accuracy of the program was closely related to the consistency and rigidity of the structure involved in representing the email text newsfeed. The more structured emails (such as the ones from usatoday.com) presented an easier task of extraction for the program. Meanwhile, newsfeeds with a more ad-hoc text structure (such as smh.com.au) presented a more guess-like approach to pattern identification.

4 Future and Direction

Based on the results, the primary improvements to future attempts in utilising patterns for pre-processing is to reduce percentage of bad articles found and to improve the percentage of articles missed.

One way to screen articles for meaningful text or not would be to involve some kind of light weight Natural Language Processing [6]. As mentioned earlier, in our example for newsfeeds, many of the bad articles found were actually advertisements placed within the text. We could possibly screen for advertisements[1] in this case by analysing patterns of words, searching for the articles' 'audience'. In the newsfeeds case, the majority (if not all) articles found were aimed at the general reader, and did not refer to them specifically. This is unlike an advertisement where the language is slightly catered for a more personal feel.

5 Conclusion

Pre-processing is an important step in the overall data mining process. The ability to extract the appropriate information is vital for subsequent algorithms and mining activities. Domain experts are excellent for writing programs for specific domains, though this is very cumbersome.

Two different approaches to identifying patterns have been described. Each with its own strength. These two approaches have been combined in a program that identifies patterns from newsfeed data. Despite the simplicity of these approaches the program performs rather well. Structure text presents an opportunity for programs to utilise the patterned approach in setting out text to extract meaningful information. If we can identify the dominant patterns, and can extract the text according to them, then perhaps we can unburden the load placed on creating data and text mining applications.

References

1. Domingos, P. and Richardson, M. Mining the Network Value of Customers, In: Proc. 7th Int. Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, 2001, ACM Press, pp. 57–66
2. Han, J. and Kamber, M. Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco, CA.
3. Hand, D., Mannila, H. and Smyth, P. Principles of Data Mining. The MIT Press, Cambridge, MA, 2001.
4. Hilderman, R. J. and Hamilton, H. J. Knowledge discovery and interestingness measures: A survey. Technical Report CS 99-04, Department of Computer Science, University of Regina, October 1999.
5. Leake, D. and Kolodner, J. Learning through case analysis. In: Encyclopedia of Cognitive Science. Macmillan, London 2001.
6. Lin, D. and Pantel, P. DIRT – Discovery of Inference Rules from Text. In Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining KDD 2001, pp. 323–328.

Evolutionary Rule Generation Classification and Its Application to Multi-class Data

Susan E. Bedingfield and Kate A. Smith

School of Business Systems Monash University Clayton, Victoria 3168 Australia
{sue.bedingfield, kate.smith}@infotech.monash.edu.au

Abstract. This paper considers an evolutionary algorithm based on an information system for generating classification rules. Custom genetic operators and a multi-objective fitness function are designed for this representation. The approach has previously been illustrated using a binary class data set. In this paper we explore the possibility of using the algorithm on a multi-class data set. The accuracy of the rules produced by the evolutionary algorithm approach are compared to those obtained by a decision tree technique on the same data. The advantages of using an evolutionary classification technique over the more traditional decision tree structure are discussed.

1 Introduction

The classification of data into distinct groups is one of the main tasks of data mining. While many data sets involve a binary target attribute (for example good or bad credit risk), a significant number of important data sets involve the division of the data into multiple groups which refer to as multi-class data. There are many techniques for classifying both binary and multi-class data, one of the most widely used being decision trees [1]. The main reasons for the popularity of decision trees is that they are a rule based approach and they classify every data point. However for many real world data sets clear classification boundaries may not exist and this may be useful information that becomes lost when adopting a technique that forces such clear cut classifications. In addition decision trees only explore a subset of all possible rules due to the greedy nature of their decision making [2]. The generation of branches in a decision tree is dependent upon previous splitting, each split seeks to maximise the accuracy of the classification at that level rather than looking ahead to consider the entire search space. Given this limitation, and the attractiveness of a rule based approach to classification for ease of interpretability, we consider the benefits of using an evolutionary approach to rule generation.

In this paper we first introduce an evolutionary classification technique and test its viability on a database with a multi-class target attribute. We then compare the evolutionary classification technique with See5, a widely used decision tree software package used for classification. The evolutionary technique used in this paper has been reported upon in [3],[4] and [5]. So far the technique has been used a database

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 868–876, 2003.
© Springer-Verlag Berlin Heidelberg 2003

with a binary consequence attribute – the well known German credit scoring data. However the technique can be generalised to any number of classes corresponding to the number of attribute values belonging to the consequence attribute. In the example that follows we use the well known Iris database with 3 classes.
We then compare the evolutionary technique implemented in CS-classifier with the decision tree algorithm implemented by See5, both in terms of the classification rules generated and the error rates observed by testing the rules on randomly generated test sets.
CS-classifier is a global evolutionary search technique that makes no assumptions about the attribute values, the consistency or accuracy of the data. See5, on the other hand, is a greedy search algorithm which forces classification irrespective of the data quality.
Our aim is to compare the performance of the two methods and determine if the more flexible nature of the evolutionary approach leads to more insightful and accurate rules.

2 The Evolutionary Algorithm

2.1 Rules and Rule Structure

Each population consists of a collection of rule sets and each rule set is simply a collection of rules, the number of rules in each rule set is specified as one of the parameters in the evolutionary algorithm. Each individual rule is made up of a set of atoms. An atom is a structure of the form (α,β,γ) where α represents an one of the attributes, β is a comparison operator ($\leq$, $\geq$, $=$) and γ a is number within the range of values assumed by the attribute α.
A rule has a structure of the form *(atom1, atom2, …,atomN)* where $N \leq M$ and $M \geq 2$ is a parameter defining the upper bound on the number of atoms allowed to make a rule. A restriction is placed on the number of atoms allowed to ensure that the rules are easily interpreted and understood by humans. For each rule, $atom_1, atom_2, \ldots, atom_{N-1}$ have a condition attribute as the attribute component, and $atom_N$ has a decision attribute as its attribute component.
Note that a particular attribute may appear more than once in an atom, for example when an upper and lower bound for a specific attribute is required. For example, a rule might be:
(petal length>50, petal width <=60, species = iris setosa).
A rule system, $\Re$, comprises a finite set of rules $\{rule_1, rule_2, \ldots, rule_k\}$ where $1 \leq k \leq K$, K being a parameter defining the number of allowable rules in a rule set. The evolutionary algorithm population consists of P rule systems each containing k individual rules.

2.2 Genetic Operators

The genetic operators used in this paper were an elitist selection criteria, crossover and mutation.

Crossover
Crossover between individual rule systems is accomplished randomly based on a fixed crossover probability P_c.The first *C* rules from each rule system are selected and swapped.(The starting point for the swap is also determined as part of the crossover process). *C* may vary from 0 to *M* where *M* is half the size of the smaller rule system. The rule systems produced by the crossover are added to the population, thus temporarily increasing its size.

Mutation
Mutation occurs to the rules within the individual rule bases and takes various forms random changes to the components of each atom, swapping one or more atoms between two rules, deleting a rule, adding a new rule. The probabilities of each type of mutation is parameterised .

Selection
The selection criteria used is elitist - the best *P* rule sets of the current generation are selected to form the next generation.

2.3 Fitness Function Components

There are several measures of fitness of the individual rules in each rule base that we are interested in recording and improving over time. We first consider accuracy measures – both row and column accuracy – of the resulting confusion matrix. The particular information system we are experimenting with has a decision attribute with 3 possible values. This leads to 3 row accuracy measures $M_1,\ldots.. M_3$, and *3* column accuracy measures $M_4,\ldots.. M_6$,. $M_1,\ldots.. M_3$, effectively measure the coverage by the rule system of the *3* classification groups. Whereas $M_4,\ldots..,M_6$ measure the accuracy of the predictions for each classification group. These 6 expressions should optimally be equal to 1. Analagous to the three accuracy measures $M_1,\ldots.. M_3$, are three measures of inaccuracy, $M_7,\ldots.. M_9$, designed to reduce incorrect classifications.
We let *O* represent the set of data points in the Iris database, *A* the set of attributes. Let $V_d = \{ v_1 , v_2 , v_3 \}$ represent the set of attribute values that the target attribute can take. If $v \in V_d$, where *d* is the target attribute, we define O_v to be the subset of *O* whose decision attribute value is *v*.
If we let $\Re$ represent a rule set and if $\rho \in \Re$, then we define O_ρ to be the subset of *O* whose members satisfy the condition attribute values associated with ρ and $O_{\rho ,d(\rho)}$ to be the subset of *O* whose members satisfy the rule ρ (i.e. they satisfy the condition and target values for ρ).

We can express the measures $M_1,\ldots.. M_9$, as follows:

$$M_i, i = 1,...,3 \text{ are defined as } \frac{Card\left(O_{v_i} \cap \bigcup_{d(\rho)=v_i} O_\rho\right)}{Card\left(O_{v_i}\right)}$$

$$M_{i+3},\ i = 1,...,3 \text{ are defined as } \frac{Card\left(O_{v_i} \cap \left(\bigcup_{d(\rho)=v_i} O_\rho\right)\right)}{Card\left(\bigcup_{d(\rho)=v_i} O_\rho\right)}$$

$$M_{i+6},\ i = 1,...,3 \text{ are defined as } \frac{Card\left(O_{v_i} \cap \left[\bigcup_{d(\rho)\neq v_i} O_\rho\right]\right)}{Card(O_{v_i})}$$

Combining Fitness Function Components

The total fitness function for the evolutionary algorithm can be represented as the weighted function of the nine components M_1-M_9 defined above, and attempts to generate rules over time with maximal coverage, maximal accuracy, minimal contradictions.

The Fitness Function

The fitness function used for the work presented in this paper has the following form:

$$F = \lambda F^1 + (1 - \lambda) F^2 \text{ where } 0 < \lambda < 1 \text{ and}$$

$$F^1 = \left(\frac{\sum_{i=1}^{6} M_i}{1 + \sum_{i=7}^{9} M_i}\right), \qquad F^2 = \left(\sum_{i=1}^{6} M_i + \sum_{i=7}^{9} \frac{1}{1 + M_i}\right)$$

This form of the fitness function has been discussed previously in [5]. It has been observed that the component F^1 if used solely as the fitness function tends to induce rules which are highly accurate at the cost of overall coverage. Whereas if the component F^2 is used as the fitness function, the rule systems generated tend to have a

higher classification coverage at the cost of classification accuracy. A linear combination of the two components F^1 and F^2 using the parameter lambda has been used to introduce greater versatility to the fitness function.

Parameter Values
For the experiments reported upon in this paper, the fixed parameter values listed in table 1 were used

Table 1. Parameter values

Parameter name	Value	Explanation
PopSize	10	The number of rule sets in each population
MaxAtoms	6	The maximum initial number of atoms in each rule
RulesPerSet	10	The initial number of rules in each rule set
ChgComponent	0.5	Probabilty of changing a component of an atom
SwapAtom	0.2	Probabilty of swapping atoms between rules
DelRule	0.5	Probabilty of deleting rule from a rule set
AddRule	1	The probability of adding a new rule to a rule set – in this case a new rule is always added. Any empty rules are deleted.
AddAtom	0.2	Probability of adding an atom to a rule
CrossOver	0.5	Probabilty of crossing two rulesets

3 Experimental Design and Data Set

3.1 The Data

The data on which the rule system generation is based is the Iris database which originated with Fisher [6]. This data set was chosen because the target attribute is multiclass and also the data set has been widely reported upon in the literature. The data contains 150 examples of irises belonging to three different species, iris setosa, iris versicolor and iris virginica. There are 5 attributes available for modelling the iris species. These include species, sepal width and length, petal width and length. There is an equal distribution of cases amongst the three species. For this experiment a 5 fold cross validation was performed on the data to establish the viability of the technique on a dataset with a non-binary valued consequence attribute. A 5 fold cross validation was then performed using See5 with the same test and training sets, and a comparison was made between the two techniques.

5-Fold Cross-Validation
Five distinct training and test sets containing 80% and 20% of the total data set respectively were randomly generated from the data set. Each pair of sets in turn was used to test the classification accuracy of the of the rule system generated by CS-

Classifier and compared with the classification accuracy of decision tree generated by See5.

CS-classifier was run for each training set, for each value of λ = 0, .2, .4 .6 .8, 1. For the following comparisons, the most accurate rule set for each corresponding test set was then selected.

There is one major difference in the rule sets that are generated by CS-classifier and See5.

The rules generated by See5 provide exactly one classification for all data points in the training/test sets. Whereas the rules generated by CS-classifier may provide zero, one or many classifications for any given data point in the training/test sets. Thus there are

there are four ways of calculating the error rates for the CS-classifier predictions:

1. error rate = (number of misclassifications)/(total number of data points classified)
2. error rate = (number of misclassifications)/(total number of data points). In this case, if for a specific data point there is no rule providing a prediction, this is interpreted as a misclassification.
3. error rate = (total number of misclassifications using most accurate rule)/(total number of classifications). In this case, if for a specific data point there is a rule which provides a correct classification and there are no rules of greater accuracy providing an incorrect classification, then this is interpreted as a correct classification.
4. error rate = (total number of misclassifications)/(total number of data points). In this case also, if for a specific data point there is a rule which provides a correct classification and there are no rules of greater accuracy providing an incorrect classification, then this is interpreted as a correct classification.

4 Results

The error rates for training and test sets using each of the four interpretations are listed in tables 2 and 3 respectively.

Table 2. Error rates for training sets

Method	Set 1	Set 2	Set 3	Set 4	Set 5	mean	Standard deviation
See5	2.5	1.67	2.5	2.5	2.5	2.33	0.37
1. CS-classifier	3.57	1.75	4.00	5.98	5.08	4.08	1.61
2. CS-classifier	10.00	6.67	20	8.33	6.67	10.33	5.58
3. CS-classifier	2.68	0	3	4.27	3.39	2.67	1.61
4. CS-classifier	9.17	5	19.17	6.67	5.00	9.00	5.93

Table 3. Error rates for test sets

Method	Set 1	Set 2	Set 3	Set 4	Set 5	mean	Standard deviation
See5	0	13.33	6.67	6.67	13.33	8.00	5.00
1. CS-classifier	0	0	6.67	3.33	6.9	3.38	3.04
2. CS-classifier	0	10	6.67	3.33	10	6.00	3.89
3. CS-classifier	0	0	3.33	0	3.45	1.36	1.66
4. CS-classifier	0	10	3.33	0	6.67	4.00	3.89

Observing Tables 2 and 3 we see that Training accuracy results for See5 are much more consistent than those for CS-classifier. Also, error rates for See5 on the training sets appear lower than on the test sets. Conversely, error rates for CS-classifier on the test sets appear lower than on the training sets.
An analysis of variance was performed between the error rates for See5 on the test sets and the error results for CS-classifier using the four separate methods for assessing the error rate. It was found that for methods 1,2 and 4 there is no statistical difference in error rates, (p-values being .081, .456 and .152) between the two techniques. However if method 3 is used to assess the error rate for CS-classifier, the analysis indicates an improvement in the classification accuracy (with p-value .011).

For training set 1, the rule derived from the decision tree generated by See5 is as follows:

```
If petal-length <= 190: Iris-setosa (40,0)
Else if petal-length > 190:
        if petal-width > 170: Iris-virginica (39,1)
        else if petal-width <= 170:
                if .petal-length <= 490: Iris-versicolor (35,1)
                else if  petal-length > 490:
                        if .petal-width <= 150: Iris-virginica (3,0)
                        else if petal-width > 150: Iris-versicolor (3,1)
```

The numbers in brackets indicate the number correctly classified followed by the number misclassified by the rule at each branch. This rule completely classifies test set 1 correctly.
In comparison, one of the rule sets generated by CS-classifier using training set 1 is as follows:

Rule 1: (petal_length<239) and (petal_width<=95) then Iris setosa (40,0)
Rule 2: (petal_width>91) and (petal_width<=172) and (sepal_length<=704) then Iris-versicolor (40,4)
Rule 3: (petal_width>182) then Iris-virginica (29,0)
Rule 4: (petal_length>517) then Iris-virginica (28,0)
This rule set also correctly classifies test set 1. The numbers in brackets indicate the total number classified followed by the number misclassified by the rule. Some brief observations on each of the rules follows.

Rule 1 of CS-Classifier
In this case either condition for rule 1 of CS-classifier would be sufficient to classify Iris setosa correctly with complete accuracy on the training set. The minimal class boundary for petal length of 190 has been found by See5. however the class boundary found by CS-classifier is 239 rather than 190. This is because for the purposes of classification, 239 works just as well so that fitness measure is not improved with a value of 190 rather than 239 (in fact any value between 190 and 299 would work just as well).

Rule 3 and Rule 4 of CS-Classifier
Rules 3 and 4 classify Iris virginica completely with complete accuracy on both the training set and test set. There is some classification overlap here – i.e. both Rule 3 and Rule 4 correctly classify some of the same data points in the training and test sets.

Rule 2 of CS-Classifier
In this case all three conditions in rule 2 are necessary to ensure the same level of accuracy (90% on the training set and 100% on the test set).

5 Conclusion

We have demonstrated the viability of the evolutionary algorithm on a multi-class data set. We have also demonstrated that the classification rules generated by CS-classifier compare favourably, in terms of accuracy, with those generated by See5. See5 uses a greedy search algorithm which forces classification rules on the training data. This can be regarded as an advantage as all data points in the test set can then be given a classification. However, the disadvantages of this is that the rules may well provide less accurate classifications on test set and the rules produced may provide less insight into the true nature of the data divisions. Conversely, although CS-classifier is encouraged to classify by the components M_1,..,M_3 in the fitness function, it is not forced to classify and will not do so if it means reducing the value of the fitness function. CS-classifier has the ability to recognise the possible uncertain and contradictory nature of a data set. The result of this is that using the rules generated by CS-classifier, we see a decrease in the classification errors on the test set compared to the training set, rather than the increase of classification errors generated on the test set compared with the training that we see when we use the rule generated by See5. The ability of CS-classifier to provide accurate classification rules can be enhanced by weighting the fitness function components M_1-M_9. These weights can be defined in a way that reflects the actual misclassification costs associated with each of the components [5]. The technique also lends itself to hybridisation with fuzzy and rough set approaches, which will be the subject of future work.

References

1. Quinlan, J. Ross, “C4.5:Programs for Machine Learning”, The Morgan Kaufmann Series in Machine Learning, Pat Langley, Series Editor, 1992.
2. Information Discovery Inc., “Rules are much More than Decision Trees”, http://www.datamining.com/trees.htm, access date: 10 Dec., 2002.
3. Bedingfield, S. E. and Smith, K. A., “Evolutionary Rule Generation in an Information System”, in Dagli, C. H. et al. (eds.), Intelligent Engineering Systems Through Artificial Neural Networks, ASME Press, NY, volume 9, pp. 485–492, 1999.
4. Bedingfield, S. and Smith, K. A., “A Comparison of Fitness Functions for Evolutionary Rule Generation”, in M. Mohammadian (ed.), Advances in Intelligent Systems: Theory and Applications, IOS Press, 2000.
5. Bedingfield, S. and Smith, K. A., “Evolutionary rule generation and its application to credit scoring”, L.Reznik and V.Kreinovich (eds.), Soft Computing in Measurement and Information Acquisition, Physica-Verlag, Heidelberg, accepted for publication, 2001.
6. Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annals of Eugenics, 7, Part II, 179–188 (1936).

Self-Organizing Hybrid Neurofuzzy Networks

Sung-Kwun Oh[1], Su-Chong Joo[1], Chang-Won Jeong[1], and Hyun-Ki Kim[2]

[1] School of Electrical, Electronic and Information Engineering, Wonkwang University, South Korea
{ohsk, scjoo, mediblue}@wonkwang.ac.kr

[2] Department of Electrical Engineering, University of Suwon, South Korea
{hkkim}@mail.suwon.ac.kr

Abstract. We introduce a concept of self-organizing Hybrid Neurofuzzy Networks (HNFN), a hybrid modeling architecture combining neurofuzzy (NF) and polynomial neural networks(PNN). The development of the Self-organizing HNFN dwells on the technologies of Computational Intelligence (CI), namely fuzzy sets, neural networks, and genetic algorithms. The architecture of the Self-organizing HNFN results from a synergistic usage of NF and PNN. NF contribute to the formation of the premise part of the rule-based structure of the Self-organizing HNFN. The consequence part of the Self-organizing HNFN is designed using Self-organizing PNN. We also distinguish between two types of the Self-organizing HNFN architecture showing how this taxonomy depends on connection points. Owing to the specific features of two combined architectures, it is possible to consider the nonlinear characteristics of process and to get output performance with superb predictive ability. The performance of the Self-organizing HNFN is quantified through experimentation that exploits standard data already used in fuzzy modeling. These results reveal superiority of the proposed networks over the existing fuzzy models.

1 Introductory Remarks

With the continuously growing demand for models for complex systems inherently associated with nonlinearity, high-order dynamics, time-varying behavior, and imprecise measurements there is a need for a relevant modeling environment. Efficient modeling techniques should allow for a selection of pertinent variables and a formation of highly representative datasets. The models should be able to take advantage of the existing domain knowledge (such as a prior experience of human observers or operators) and augment it by available numeric data to form a coherent data-knowledge modeling entity. The omnipresent modeling tendency is the one that exploits techniques of CI by embracing fuzzy modeling, neurocomputing, and genetic optimization. In this study, we develop a hybrid modeling architecture, called Self-organizing Hybrid Neurofuzzy Networks (HNFN). In a nutshell, Self-organizing HNFN is composed of two main substructures, namely a neurofuzzy (NF) and a polynomial neural network(PNN). From a standpoint of rule-based architectures, one can

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 877–885, 2003.
© Springer-Verlag Berlin Heidelberg 2003

regard the NF as an implementation of the antecedent part of the rules while the consequents (conclusion parts) are realized with the aid of PNN. The role of the NF is to interact with input data, granulate the corresponding input spaces (viz. converting the numeric data into representations at the level of fuzzy sets). NF has two connection points for combination with PNN. By considering two types of connection methods, PNN can be used effectively and organized dynamically (through a growth process) to meet the required approximation error. The role of the PNN is to carry out nonlinear transformation at the level of the fuzzy sets (and corresponding membership grades) formed at the level of NF. The PNN is constructed on a basis of a Group Method of Data Handling (GMDH [1]) method and is a flexible and versatile structure [9]. In this network, the number of layers and number of nodes in each layer are not predetermined (unlike in case of most neural-networks) but can be generated in a dynamic fashion. To assess the performance of the proposed model, we exploit a nonlinear function. Furthermore, the network is directly contrasted with several existing fuzzy models.

2 The Architecture and Development of Self-Organizing HNFN

In this section, we elaborate on the architecture and a design process of the Self-organizing HNFN. These networks result as a synergy between two other general constructs such as NF [2] and PNN [9].

2.1 The Architecture of Self-Organizing HNFN

The topology of the Self-organizing HNFN is constructed by combining NF for the premise part of the Self-organizing HNFN with PNN being used as the consequence part of Self-organizing HNFN. As visualized in Fig. 1, NF can be designed by using space partitioning in terms of individual input variables. We are concerned with a granulation carried out in terms of fuzzy sets defined in each input variable.

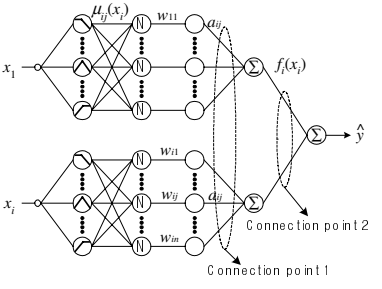


Fig. 1. NF structure with two connection points of interaction with PNN

Also, NF structure has two possible connection points. The location of this point implies the character of the network (viz. its flexibility and learning capabilities). Note that the first connection point allows perceiving each linguistic manifestation of the original variables (viz. these variables are transformed by fuzzy sets and normalized). The location of the other connection point implies that the PNN part of the network does not "see" the individual fuzzy sets in the input space. We use PNN in the consequence structure of the Self-organizing HNFN. The PNN algorithm based on the GMDH method can produce an optimal nonlinear system by selecting significant input variables among dozens of these and forming various types of polynomials. PNN is used in selecting the best ones in partial descriptions (PDs) according to a discrimination criterion. PDs use regression polynomials, Table 1. Successive layers of the Self-organizing HNFN are generated until we reach a structure of the best performance.

Table 1. Types of regression polynomial

No. of inputs / Order of the polynomial	1	2	3	4
1 (Type 1)	Linear	Bilinear	Trilinear	Tetralinear
2 (Type 2)	Quadratic	Biquadratic-1	Triquadratic-1	Tetraquadratic-1
2 (Type 3)		Biquadratic-2	Triquadratic-2	Tetraquadratic-2

The following types of the polynomials are used

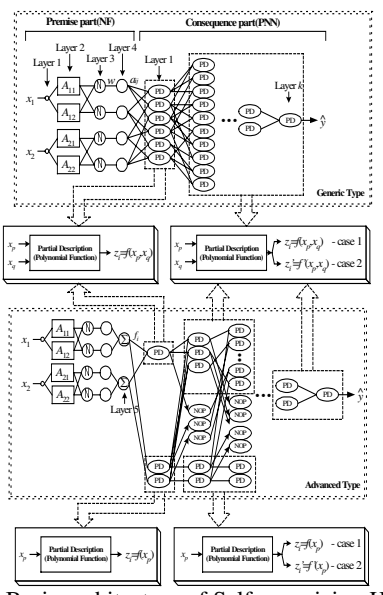
- Bilinear $= c_0 + c_1 x_1 + c_2 x_2$
- Biquadratic-1 = Bilinear+ $c_3 x_1^2 + c_4 x_2^2 + c_5 x_1 x_2$
- Biquadratic-2 = Bilinear $+ c_3 x_1 x_2$

The Self-organizing HNFN is an architecture combined with the NF and PNN as shown in Fig. 2. These networks result as a synergy between two other general constructs such as NF and PNN. According to the alternative of two connection points, combination of NF and PNN is implemented for generation of Self-organizing HNFN architecture such as Fig. 2. Also each type of Self-organizing HNFN includes two architectures, namely basic and modified architectures. Moreover, for each architecture of the topology we identify two cases.

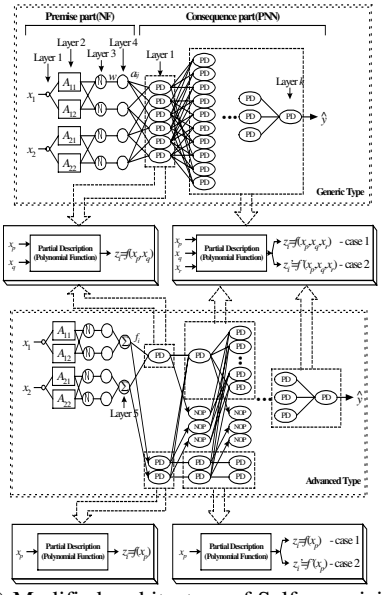
- Generic type of Self-organizing HNFN;Combination of NF and PNN by connection point 1
- Advanced type of Self-organizing HNFN; Combination of NF and PNN by connection point 2
  - (a) Basic architecture – The number of input variables of PDs of PNN is same in every layer.
  - (b) Modified architecture – The number of input variables of PDs of PNN is different in each layer.

  Case 1. The polynomial order of PDs of PNN is same in every layer.

  Case 2. The polynomial order of PDs in the 2nd layer or higher of PNN has a different or modified.



(a) Basic architecture of Self-organizing HNFN



(b) Modified architecture of Self-organizing HNFN

Fig. 2. Self-organizing HNFN architecture; Combination of NF and PNN using two connection points

In order to enhance the learning of the Self-organizing HNFN and augment its performance of a Self-organizing HNFN, we use genetic algorithms (GAs) to adjust learning rate, momentum coefficient and the parameters of the membership functions of the antecedents of the rules. GAs is a stochastic search technique based on the principles of evolution, natural selection, and genetic recombination by simulating "survival of the fittest" in a population of potential solutions (individuals) to the problem at hand. GAs are capable of globally exploring a solution space, pursuing potentially fruitful paths while also examining random points to reduce the likelihood of setting for a local optimum. The main features of genetic algorithms concern individuals viewed as strings, population-based optimization (search through the genotype space) and stochastic search mechanism (such as selection and crossover). A fitness function (or fitness, for short) used in genetic optimization is a vehicle to evaluate a performance of a given individual (string) [3].

2.2 The Design of Self-Organizing HNFN

The design procedure for each layer in the premise and the consequence of Self-organizing HNFN is as follows (refer to Fig. 1 and Fig. 2).

The premise of Self-organizing HNFN

[Layer 1] Input layer.

[Layer 2] Computing activation degrees of linguistic labels.

[Layer 3] Normalization of a degree activation (firing) of the rule.
[Layer 4] Multiplying a normalized activation degree of the rule by connection weight.

$$a_{ij} = \overline{\mu}_{ij}(x_i) \times w_{ij} \tag{1}$$

If we choose Connection point 1 for combining architecture of the NF and PNN, a_{ij} is given as the input variable of the PNN. If we choose Connection point 2, a_{ij} corresponds to the input signal to the output layer of NF viewed as the input variable of the PNN.
[Layer 5] Fuzzy inference for output of the rules: The output of each node in the 5th layer of the premise part of Self-organizing HNFN is inferred by the center of gravity method. If we choose Connection point 2, see Fig. 1 and 2, f_i is the input variable of PNN that is the consequence part of Self-organizing HNFN

$$f_i(x_i) = \sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} \overline{\mu}_{ij}(x_i) \cdot w_{ij} = \frac{\sum_{j=1}^{n} \mu_{ij}(x_i) \cdot w_{ij}}{\sum_{j=1}^{n} \mu_{ij}(x_i)} \tag{2}$$

The consequence of Self-organizing HNFN
[Step 1] Configuration of input variables: $x_1=a_{11}$, $x_2=a_{12},\ldots$, $x_n=a_{ij}$ (n=i×j).
[Step 2] Forming a PNN structure: We select input variables and fix an order of a PD.
[Step 3] Estimate the coefficients of a PD: The vector of coefficients of the PD's (C_i) in each layer is produced by the standard least-squares method and expressed in the form

$$\mathbf{C}_i = (\mathbf{X}_i^T \mathbf{X}_i)^{-1} \mathbf{X}_i^T \mathbf{Y} \tag{3}$$

Where i denotes node number. This procedure is implemented repeatedly for all nodes of the layer and also for all layers of consequence part of Self-organizing HNFN.

[Step 4] Choosing PDs with the highest predictive capability: Each PD is constructed and evaluated using the training and testing data, respectively. Then we compare the values of the performance index and select PDs with the highest predictive capabilities (lowest values of the performance index). We may use i) the predetermined number W of the PDs or ii) go for all of them whose performance index is lower than a certain prespecified value. Especially the method of (ii) uses the threshold criterion θ to select the node with the best performance in each layer.

$$\theta = E_{\min} + \delta \tag{4}$$

where θ is a new value of the criterion, δ is a positive constant (increment) and E_{min} denotes the smallest value obtained in the each layer.
[Step 5] Termination condition: It is prudent to take into consideration a stopping condition ($E_{min} \geq E_{min^*}$) for better performance and the number of iterations (size of the network) predetermined by the designer. Where E_{min} is a minimal identification error at the current layer while E_{min^*} denotes a minimal identification error at the previous layer.
[Step 6] Determining new input variables for the next layer: The outputs of the preserved PDs serve as new inputs to the next layer. In other words, we set $x_{1i}=z_{1i}$, $x_{2i}=z_{2i}$,

…, $x_{Wi}=z_{Wi}$. The consequence part of Self-organizing HNFN is repeated through steps 3-6.

3 Simulation Results

This section presents simulation results of the Self-organizing HNFN for the time series data of gas furnace utilized by Box and Jenkins [4-8]. Consider a gas furnace system in which air and methane are combined to form a mixture of gases containing CO2 (carbon dioxide). We use 2 inputs 1 output, [$u(t$-3), $y(t$-1); $y(t)$] and 3 inputs 1 output, [$u(t$-2) , $y(t$-2), $y(t$-1); $y(t)$] for evaluating the proposed models. Where, $u(t)$ denotes the flow rate of methane gas and $y(t)$ stands for the carbon dioxide density. We consider 296 pairs of the original input-output data. 148 out of 296 pairs of input-output data are used as learning set; the remaining part serves as a testing set. The performance index (PI) is defined by (5).

$$PI = \frac{1}{N}\sum_{p=1}^{N}(y_p - \hat{y}_p)^2 \tag{5}$$

GAs help optimize learning rate, momentum coefficient, and the parameters of the membership function in premise part of Self-organizing HNFN. GAs use serial method of binary type, roulette-wheel as the selection operator, one-point crossover, and an invert operation in the mutation operator [3]. GAs was run for 100 generations with a population of 60 individuals. Each string was 10 bits long. The crossover rate was set to 0.6 and probability of mutation was equal to 0.1. Fig. 3 shows the preferred architecture in generic type of the Self-organizing HNFN that is composed of NF and PNN with 3→4 inputs-Type 3→2 topology. The way in which learning has been realized is shown in Fig. 4 where both training and testing data are illustrated. Fig. 5 and 6 show the preferred architecture in advanced type of the Self-organizing HNFN.

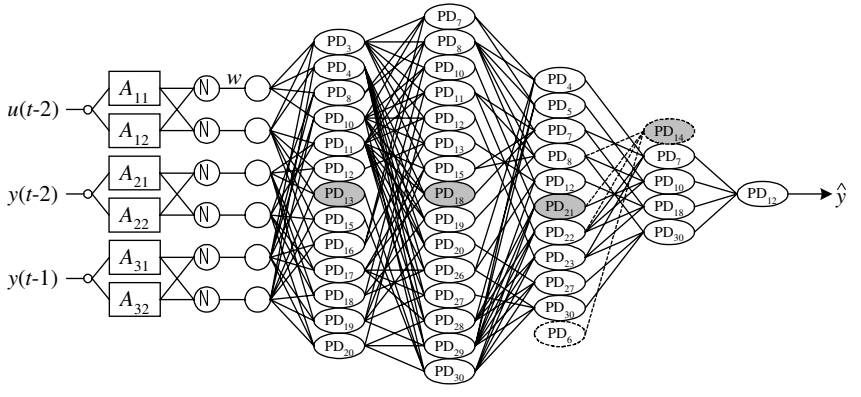
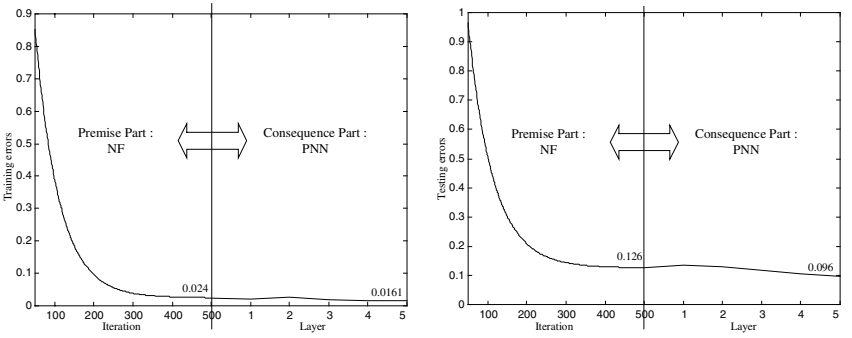


Fig. 3. Modified architecture-case 2 in generic type of the Self-organizing HNFN



(a) Training errors (PI) (b) Testing errors (E_PI)

Fig. 4. Learning procedure of modified architecture-case 2 in generic type of the Self-organizing HNFN

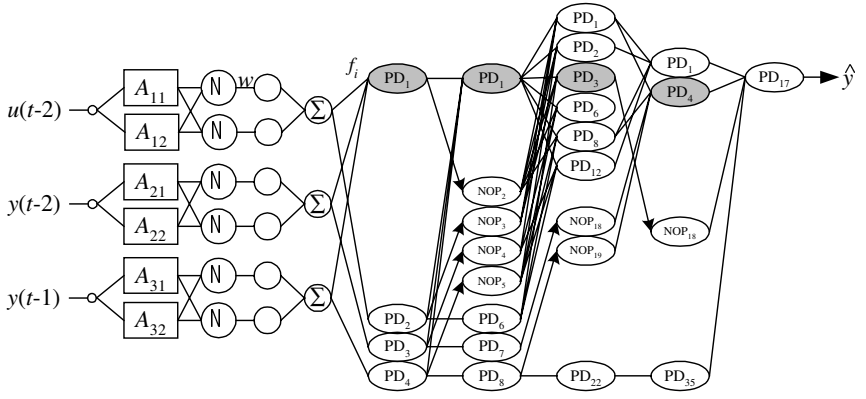
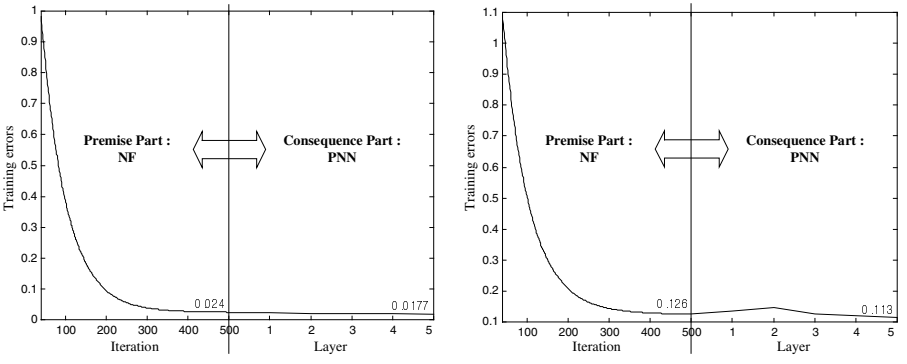


Fig. 5. Modified architecture-case 2 in advanced type of the Self-organizing HNFN



(a) Training errors (PI) (b) Testing errors (E_PI)

Fig. 6. Learning procedure of modified architecture-case 2 in advanced type of the Self-organizing HNFN

Table 2 contains a comparative analysis including several previous models. Compared with these models, the Self-organizing HNFN comes with high accuracy and improved prediction capability.

Table 2. Comparison of identification error with previous modeling methods

Model					PI	E_PI
Lin and Cunningham's model[4]					0.071	0.261
Kim's model[5]					0.034	0.244
Oh's Fuzzy model[7]					0.020	0.264
Oh's Adaptive FNN[8]					0.021	0.332
Oh and Pedrycz's Fuzzy model[6]					0.020	0.271
Our model (HNFN)	[$u(t$-3), $y(t$-1); $y(t)$]	Generic type	Basic architecture	Case 1	0.0216	0.270
				Case 2	0.0235	0.268
			Modified architecture	Case 1	0.0181	0.246
				Case 2	0.0176	0.250
		Advanced type	Basic architecture	Case 1	0.0231	0.270
				Case 2	0.0237	0.277
			Modified architecture	Case 1	0.0194	0.267
				Case 2	0.0199	0.264
	[$u(t$-2) , $y(t$-2), $y(t$-1); $y(t)$]	Generic type	Basic architecture	Case 1	0.0164	0.104
				Case 2	0.0172	0.103
			Modified architecture	Case 1	0.0172	0.106
				Case 2	0.0161	0.096
		Advanced type	Basic architecture	Case 1	0.0198	0.121
				Case 2	0.0205	0.119
			Modified architecture	Case 1	0.0178	0.115
				Case 2	0.0177	0.113

4 Conclusions

In this study, we have introduced a class of Self-organizing HNFN regarded as a modeling vehicle for nonlinear and complex systems. Self-organizing HNFN is constructed by combining NF with PNN. In this sense, we have constructed a coherent platform in which all components of CI are fully utilized. The model is inherently dynamic - the use of the PNN is essential to the process of "growing" the network by expanding the depth of the network. A comprehensive design procedure was developed. The series of experiments helped compare the network with other fuzzy models - in all cases the previous models came with higher values of the performance index.

Acknowledgements. This paper was supported by the Korea Science & Engineering Foundation (KOSEF: grant No. R05-2000-000-00284-0) and partially by Wonkwang University in 2002.

References

1. A. G. Ivahnenko: The group method of data handling: a rival of method of stochastic approximation. Soviet Automatic Control. Vol. 13, No. 3 (1968) 43–55
2. T. Yamakawa: A New Effective Learning Algorithm for a Neo Fuzzy Neuron Model. 5th IFSA World Conference (1993) 1017–1020
3. D. E. Goldberg: Genetic Algorithms in search, Optimization & Machine Learning. Addison-Wesley (1989)
4. Y. Lin, G. A. Cunningham III.: A new approach to fuzzy-neural modeling. IEEE Trans. Fuzzy Systems. Vol. 3, No. 2 (1995) 190–197
5. E. Kim, H. Lee and M. Park: A simply identified Sugeno-type fuzzy model via double clustering. Information Sciences. Vol. 110 (1998) 25–39
6. S.K. Oh, and W. Pedrycz.: Fuzzy Identification by means of Auto-Tuning Algorithm and Its Application to Nonlinear Systems. Fuzzy Sets and System. Vol. 115, No. 2 (2000) 205–230
7. B.J. Park, W. Pedrycz and S.K Oh: Identification of Fuzzy Models with the Aid of Evolutionary Data Granulation. IEE Proc.-Control Theory and Applications. Vol. 148, Issue 05 (2001) 406–418
8. S.K. Oh, B.J. Park and C.S. Park: On-line Modeling of Nonlinear Process Systems using the Adaptive Fuzzy-Neural Networks. The Transactions of The Korean Institute of Electrical Engineers. Vol. 48A, No. 10 (1999) 1293–1302
9. S.K. Oh and W. Pedrycz: The Design of Self-organizing Polynomial Neural Networks. Information Sciences. Vol. 141, Issue 3-4 (2002) 237–258
10. B.J. Park, W. Pedrycz and S.K. Oh: Fuzzy Polynomial Neural Networks: Hybrid Architectures of Fuzzy Modeling. IEEE Trans. on Fuzzy Systems. Vol. 10, No. 5 (2002) 607–621

FOM: A Framework for Metaheuristic Optimization

J.A. Parejo[1], J. Racero[1], F. Guerrero[1], T. Kwok[2], and K.A. Smith[2]

[1]Escuela Superior de Ingenieros, Camino de los Descubrimientos, s/n, 41092 Sevilla, Spain
[2]School of Business Systems, Monash University, 3800 Clayton, Vic Australia

Abstract. Most metaheuristic approaches for discrete optimization are usually implemented from scratch. In this paper, we introduce and discuss FOM, an object-oriented framework for metaheuristic optimization to be used as a general tool for the development and the implementation of metaheuristic algorithms. The basic idea behind the framework is to separate the problem side from the metaheuristic algorithms, allowing this to reuse different metaheuristic components in different problems. In addition to describing the design and functionality of the framework, we apply it to illustrative examples. Finally, we present our conclusions and discuss futures developments.

1 Introduction

In the past 15 years, a large amount of work has been carried out on development and research about heristics and metaheuristics. However, most of the efforts were directed to solve classical OR problems using this techniques, trying to improve results obtained using traditional techniques, and showing that this alternative methods are relevant and advantageous. However, that generated a lot of documentation, algorithms and researches about this techniques applied and adapted to this problems.

Until now, to apply this techniques we had to make a big effort to develop and adapt them to specific problems. This effort is added to the inherent complexity of some of this techniques, and make that in some cases these techniques were not applied. To avoid this, we have developed a tool that is aimed at making applicable these techniques in real production environments, with a minimum effort on development and adaptation..

The developed tool is called a Framework[1] in terms of computer science. This tool is flexible and extendable enough so as to be used in heterogeneous environments and to let the user to tune and retouch the needed items in order to adapt it to his needs.

In this sense our tool try to avoid ad-hoc implementations for specific problems, that reduce to the maximum the computational time to obtain a solution insignificantly better. However our tool try to make applicable these methods to a increasing number of problems in a easy, simple and fast way. And our tool is open to the numerous

[1] A framework is something more than a library. A frameworks is not only functional bat, also contains a design which can be used, adapted and completed so as to fit the needs of the user. A framework defines a entire structure for an application. The reader can find a more detailed definition in [5]

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 886–895, 2003.
© Springer-Verlag Berlin Heidelberg 2003

variants and chances that this methods bring us. It also let us make implementations of efficient methods that can be adapted to the special characteristics liable to have problems.
Although the object-oriented paradigm has been in use for many years, the production of mainstream, quality frameworks is a comparatively new phenomenon. The application of frameworks to combinatorial optimization is even more recent. Therefore we can say that this is a new area of research and development, that can contribute many advantages in order to approach optimization problems.
There are some precedents of frameworks of this type. Relevant developments in this way have been the Templar framework [1-2], and the HotFrame [3]. However we think that our framework, FOM, has enough innovations on both its design and implementation to consider it an advance in this area. Finally a relevant reference about the state of the art in these developments, and a compendium of the relevant references can be found in [4].

2 Framework Design

As a previous stage to design the framework, we have analysed the relationships between the optimization techniques, their components, the abstract problems and the specific instances of the problems that we want to solve. We have delimited and expressed in a formal way the interfaces between these elements, which are required to solve the optimization problems. Starting from this study, we have designed a structure that let us apply these techniques to the problems, and we have implemented them using the previous interfaces.
We show below a class diagram that shows the final structure of some of these interfaces. These diagrams express the relationships between the solutions to our problems and similar problems, as well as the characteristics that our solutions should have in order to apply our different metaheuristics. The most simple interface, called *Solution*, only defines the requirements that an object must comply with for being a solution to one of our problems: should be able to be evaluated and provide a new random solution to our problem. The design defines furthermore a class that implements this interface, called *AbstractSolution,* and a interface that represents the problem we want to solve, called *Problem*; likewise, it is established a relationship by means of which all *AbstractSolution* is associated to a *Problem.* Through this relationship the problem can evaluate the actual data for the solution, and determine if it is a valid solution. Finally the diagram defines some interfaces that bring the required functionalities to apply the different techniques. For example, we define an interface called *Neighbourhood* that defines the neighbourhood of a specific solution and give us the required operations to move through it.. There is another interface called *ExplorableSolution* that implements the interfaces *Solution* and *Neighbourhood.* In this way, using an object that implements this interface we, can explore the neighbourhood of this solution and "jump" from one solution to other, moving us around the solution space of the problem. This let us apply techniques based on local search. In the same way the different interfaces that are needed to apply the different techniques are defined. Also, the elements used in the different techniques and their functionality are defined.

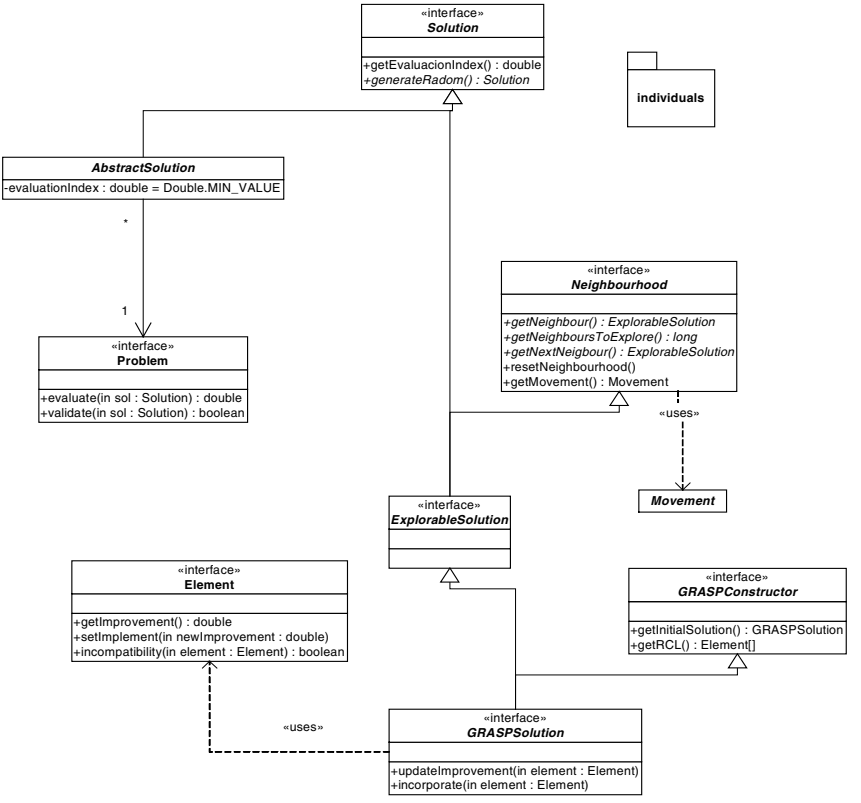


Fig. 1. Class Diagram of Problems and Solutions

In our framework we have implemented the metaheuristics listed below: Steepest Descent, Iterative Steepest Descent, Tabu Search [6], Simulated Annealing [7], GRASP, Variable Neighbourhood Search [8] and Genetic Algoritms [9-10]. One might think that the first two techniques are not important and could not be called metaheuristics. However, they have been implemented because they are useful when they are combined with the other techniques. Thus, the ability to combine several metaheuristics is included in the framework in techniques like GRASP and Genetic Algorithms.

Figure 2 shows both definitions and relationships between the metaheuristics.

A class called *Metaheuristic* is defined in the previous diagram. This class defines the interface that must have every elements that we want to use as an optimization technique in the framework. It only defines a public method, that will be the optimization algorithm, and that has a solution as parameter (the initial solution) and returns another solution (the optimal solution we found). All optimization methods that can be expressed and called in this way could be used in the framework in order to find an optimal solution to a problem.

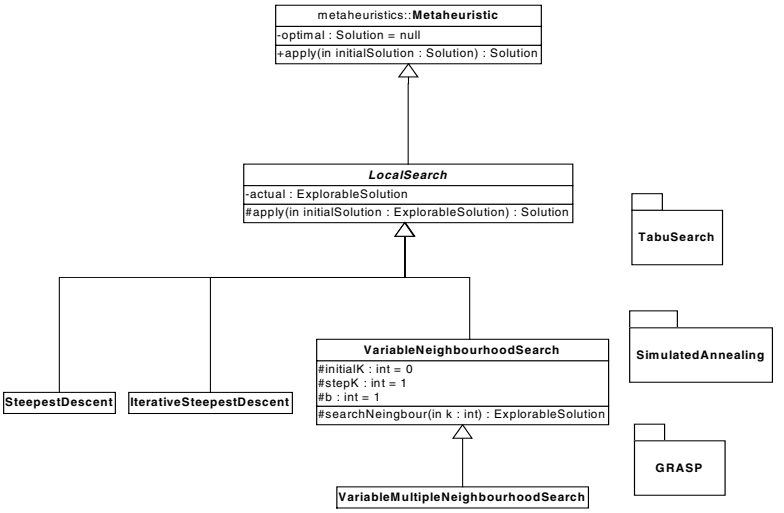


Fig. 2. Class Diagram of Metaheuristics and Local Search

In the previous diagram another class, called *LocalSearch* is defined; it only inherits the previous class, with the particularity that the initial solution must be a *ExplorableSolution* just as was defined in the Figure 1. In this way we can explore the neighbourhood of our solution and implement several techniques based on local search.

Figure 3 shows a diagram that defines the behaviour of Simulated Annealing. In this diagram the class *SimulatedAnnealing* is defined, that implements *LocalSearch*. This class uses a interface called *Cooler*, in order to decrease the value of their attribute *actualTemperature.* The interface *Cooler* defines a method called *toCool*, this method will give us the value of the temperature parameter in the next iteration. Therefore, that which determines the cooling schedule of this algorithm is the concrete *Cooler* object with what is connected the *SimulatedAnnealing* object.

As a result, we have techniques like Genetic Algorithms, Tabu Search or Simulated Annealing implemented in an abstract way. Also they are implemented in a independent way from the details of the problem we want to solve, based only on the specifications defined by our interfaces. Our framework defines a set of elements that could be useful in order to tune the concrete parameters and variants of this metaheuristic techniques that we should use in order to solve a specific problem.

It is widely accepted that a correct tuning of the parameters that control the behaviour of the metaheuristic techniques has big impact on the success of this techniques when solving problems. It is very useful in order to adapt and tune those parameters follow the value of some data along the execution of the metaheuristic technique. We could determine the best initial values that we should use for a metaheuristic, and even

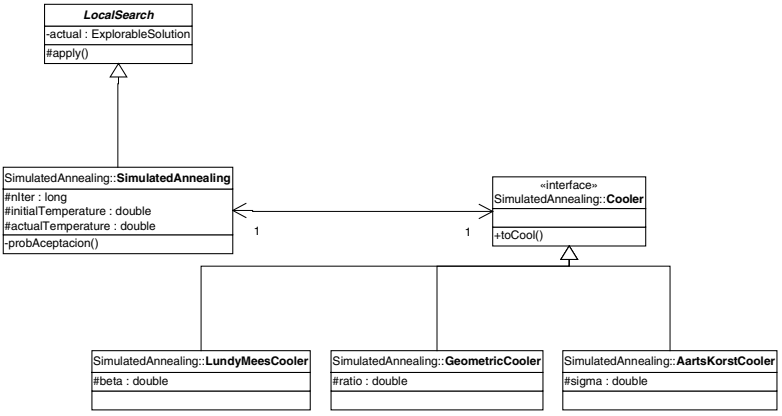


Fig. 3. Class Diagram of Simulated Annealing

determine what variant of a technique should be choosen when a specific problem is being solved This functionality is built into our framework by the use of some elements called "observers". Those elements will trace the value of the property they are following in every iteration of the algorithm. Those elements will let us to follow the evolution of any parameter of a metaheuristic. These parameters could be any of the following: the function value for the best solution we have found, the average function value of the population of a Genetic Algorithm in a iteration, the evolution of the temperature parameter in Simulated Annealing, etc. Moreover, our observers should be implemented only once, and could be used for any problem and metaheuristic technique. Also, it had been developed a set of elements, called "visualizators", that let us treat the information obtained by the observers in different ways. Using those visualizators we could by example, save in a file all the values obtained by the "observers", or show charts about their evolution. Furthermore those visualizators can be applied to any observer we have defined, even those defined for a specific problem by the user. As an example, all charts that we show later in the computational experiences had been obtained using a charting visualizator.

Finally, it is clear that using our framework we can experiment with different variants of this metaheuristic techniques without any codification effort. For example, as we have seen previously figure 3, the cooling schedule that uses Simulated Annealing is determined by the *Cooler* object that has associated. In this way we could change the *Cooler* object using one implementation or another, and subsequently using one cooling schedule or another. In this way we could experiment different cooling schedules for any problem defined for the framework, only changing the implementation of the *Cooler* object that has our Simulated Annealing object. In the same way we could change the type of memory used by Tabu Search, etc. Other relevant point of the framework is the reuse of functional elements across different metaheuristic techniques. Thus, abstraction and division of responsabilities of different functional elements can be reused. For example: in Genetic Algorithms is necessary to have criteria to choose some individuals: choose the individuals that could have a mutation or inversion, choose the individuals that could be crossed and

obtain from them other individuals, and choose the individuals that could survive and to be a member of the population in the next iteration.
In order to achieve this problem we have defined a set of elements called “selectors”. We show a class diagram with their structure:

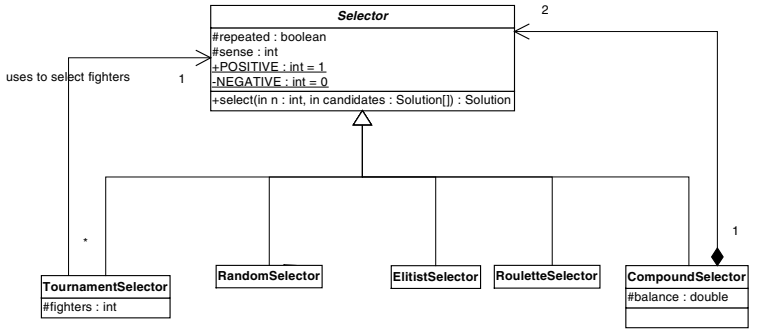


Fig. 4. Class Diagram of selector

Class *Selector* define the selection method, that taking a set of candidate solutions returns a subset that would be chosen. Moreover we define different classes that implements different criteria to choose elements, like tournaments of n elements, random, better, etc. Those elements could be used by the Ant System metaheuristic in order to choose ants that will update the feromona’s trail.
There are present again the advantages of a correct abstraction and definition of the responsibilities of functional elements.

3 Computational Experiences

With the aim of proving the efficiency and applicability of the framework. We have approached two classical problems in the area of Operational Research, the SAT problem and the TSP problem. We used standard and public libraries of instances of these problems. We used the SATLib for the SAT problem, that we gathered from: http://www.satlib.org. We used the TSPLib for the TSP problem, that we gathered from: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.
We will show some results obtained applying the metaheuristic techniques to one instance of the selected problems, in order to show the capabilities and possibilities that our framework give us. We will show the obtained solutions, and finally we will present (if we consider it is relevant) a chart showing the evolution of interesting parameters. Because of space limitations we will show only the results for techniques that we consider interesting. Moreover, detailed values of parameters used for every technique will not be listed, but we have roughly used standard values from literature However, we have a detailed description of these values and a more exhaustive study of the results obtained applying all techniques, as well as applications of the techniques that form the framework to other instances of this problems.
The SAT problem. We have chosen the SAT’s problem instance defined in the file JNH207.SAT of the SATLib to realize our tests. We will show the results obtained applying different techniques.

Tabu Search: Results: We firstly executed this algorithm using a little size for the tabu memory, and few iterations too. We could not find an optimal solution. However, we executed this algorithm again using a bigger memory size and more iterations and we found an optimal solution. Finally we executed this algorithm again in order to show the what is the critical factor, memory size or number of iterations, so we choose the same memory size than in the first application and a large number of iterations, and we could not found an optimal solution. However and according to the results that we can see in the next figures there were necessary a bigger number of iterations. Interesting facts: We show now two charts that show the evolution of the actual solution in every iteration, first with a little memory size and later with a bigger memory size. In the second chart we used a higher number of iterations.

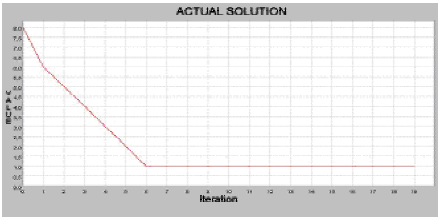


Fig. 5. Tabu Search. Non Optimal Soluts.

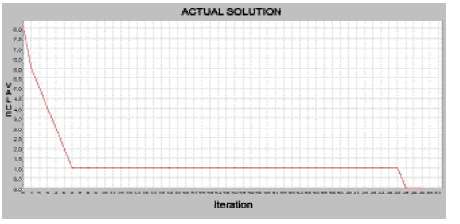


Fig. 6. Tabu Search. Optimal Soluts.

Genetic Algorithms: Results: We found an optimal solution every time we applied this technique. Interesting facts: We think that is interesting to have a look to the charts that show the evolution of the number of clauses not satisfied for every individual, and the figure that shows de evolution of the diversity of the population[2]. We can notice the evolution of our population to better solutions, and at the same time how their diversity decrease.

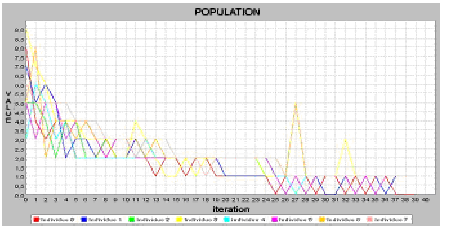


Fig. 7. Evaluation value for the individuals

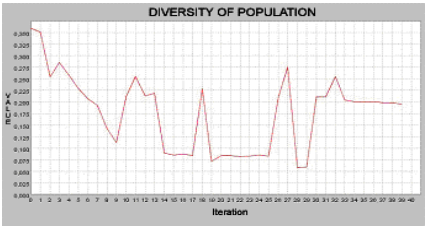


Fig. 8. Diversity of the population

The TSP problem. In order to carry out our tests we have chosen the TSP's problem instance defined in the file att48.tsp from TSPLib. We will show results we obtained for some techniques:

Tabu Search: Results: We obtained a solution with objective function of 37711.35. It is the best solution obtained in the tests shown. However, It should not be forgotten

[2] Diversity of population is calculated comparing all pairs of individuals evaluating their equality with a value between 0 and 1, finally dividing the sum by the number of individuals.

that we used the highest iteration number of whole of the tests shown in this article. Interesting facts: We show the evolution of the evaluated solution in every iteration.

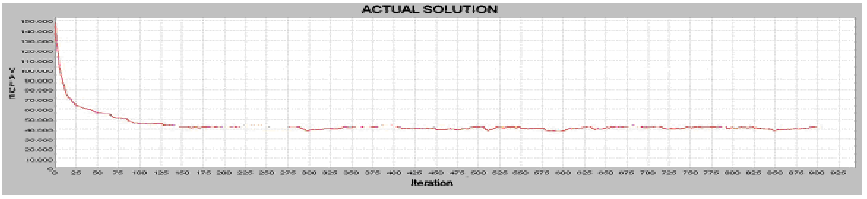


Fig. 9. Evaluated solution, Tabu Search

Simulated Annealing: Results: We obtained a solution with objective function of 54500.51 (the number of iterations was smaller than in tabu search). Interesting facts: We have chosen a cooling schedule that makes the temperature decrease geometrically. Figure 10 shows the curve of the descending temperature. Moreover, a chart that show the value of the actual solution allow us to discern the operation of this algorithm.

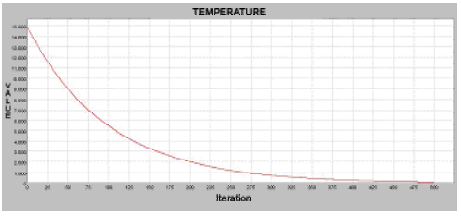


Fig. 10. Temperature Parameter

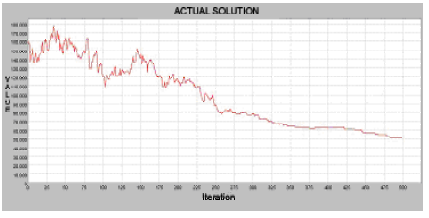


Fig. 11. Actual Solution

Genetic Algorithms: Results: We executed two variants of this technique in order to show their efficiency. In one hand we applied this technique without other mechanism for optimization than mutation, inversion and crossover of individuals, and we obtained a solution with objective function of 81012.90. On the other hand we used an advanced technique, i.e. a steepest descent algorithm applied to all individuals, we obtained better results, the obtained solution was 45817.78. Interesting facts: It is interesting to show the variation between Figures 12 and 13 which show the evolution of the individuals in the population. Evolution showed in Figure 13 is more "jagged", and has a bigger initial improvement of the population. It was because of the effect of the local search.

4 Conclusions and Future Research

We have verified through the development and use of the framework to solve diverse problems that it can be a useful tool, and that it gives us interesting possibilities and innovative elements. We will enumerate the advantages and drawbacks of the use of our framework.

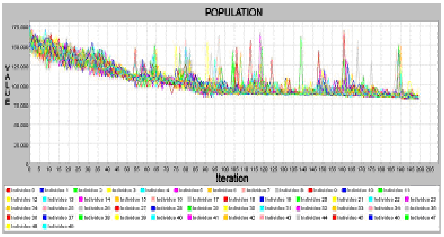


Fig. 12. Simple Genetic Algorithm

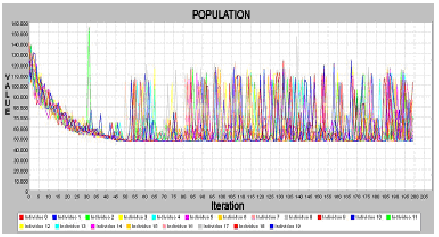


Fig. 13. Hybrid Genetic Algorithm

Advantages obtained by the use of our framework:

- We don't have to make a complete implementation of every technique for every problem we want to solve.
- It makes easy to reuse code. Moreover, it promotes that different techniques made use of the previous development of common elements.
- It induces a correct and robust structure in our implementation. It is produced thanks to the structures where our classes are embed on the framework, and to the study, separation and delimitation of the functional elements that have a role in the optimization process.
- It brings additional elements like "observators", "terminators", "selectors", etc.
- It is flexible and let us to use advanced techniques and variants. Moreover, It is open, so we can add new techniques that take advance of the common elements that had been implemented for other problems or techniques.

Drawbacks of the use of the framework:

- The use of the framework makes our implementation less efficient.
- It is necessary to learn the structure and key classes of the framework, now that it brings us all the possibilities of the framework.
- The use of the framework makes our implementation more complex, in spite of there are less code and elements to implement..

Finally we will show some future research that we think could be useful and interesting about the framework:

- We could add more metaheuristic techniques to the framework, like Ant Systems [11] or Scatter Search [12].
- We could make a distributed implementation of the metaheuristics, and as the user only have to implement the elements about solutions, it could be used without additional effort.
- We could implement a library of standard solutions. The most of the problems use the same data structures in order to represent solutions (arrays, lists, trees). We could implement general implementations of this structures as solutions (adding the necessary elements in order to can apply all techniques). In that way, the user should only use these basic structures in order to express their solutions to the problems.
- Finally we could implement a system that, based on a model of the problem expressed in a modelling language, could "compile" the required elements to can apply our techniques. We would have to use the library of basic standard solutions from the previous item and implement a compiler that compounds the solutions and problems in the correct way. It would bring us to a pseudo-declarative

paradigm of problem solving We only should describe the problem, and could obtain solutions using metaheuristics (probably that solution will not be so good as if we implement the problems and techniques "at hand", but maybe could give us a solution that solve our problem to our needs).

References

1. M. Jones. An Object-Oriented Framework for the Implementation of Search Techniques. Doctoral thesis, University of East Anglia. 2000.
2. M. Jones, G. McKeown, V. Rayward-Smith. Templar: An object-oriented Framework for distributed combinatorial optimization. In Proccedings of the UNICOM Seminars on Modern Heuristics for Decision Support. UNICOM Ltd, Brunel University, UK
3. Adreas Fink, Stefan Voβ and David L. Woodruff. Building Reusable Software Components for Heuristic Search. 1998.
4. Stefan Voβ, David Woodruff. Optimization Software Class Libraries. Kluwer Academic Publishers, 2002.
5. Grady Booch. Object-Oriented Analysis and Design with Applications (2nd Edition). Addison-Wesley. 1994
6. Fred Glover, Manuel Laguna. Tabu Search. Kluwer Academic Publishers. 1997
7. S. Kirkpatrick, C. D. Gellat and M. P. Vecchi. Optimization by Simulated Annealing. Science, 220, 671–680. 1983.
8. N. Mladenovic, P. Hansen. Variable Neighbourhood Search. Computers & Operations Research, 24, 1097–1100. 1997.
9. David E. Goldberg. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley. 1989.
10. Randy L. Haupt, Sue Ellen Haupt. Practical Genetic Algorithms. Wiley-Interscience. 1998.
11. M. Dorigo, V. Maniezzo, A. Colorni. Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, B-26: 29–41
12. M. Laguna. Scatter Search. Handbook of Applied Optimization (Pardalos et al editoral). Oxford University Press, 183–193. 2002

ClassdescMP: Easy MPI Programming in C++

Russell K. Standish[1,2] and Duraid Madina[2]

[1] School of Mathematics
[2] High Performance Computing Support Unit
University of New South Wales,Sydney, 2052,Australia
{R.Standish,duraid}@unsw.edu.au
http://parallel.hpc.unsw.edu.au/classdesc

Abstract. ClassdescMP is a distributed memory parallel programming system for use with C++ and MPI. It uses the Classdesc reflection system to ease the task of building complicated messages to be sent between processes. It doesn't hide the underlying MPI API, so it is an augmentation of MPI capabilities. Users can still call standard MPI function calls if needed for performance reasons.

1 Classdesc Reflection

MPI is an *application programming interface* (API - in other words library of functions calls) for passing data from one unix process to another. The processes may be running on the same computer, or completely distinct computers, so this system provides a means of implementing *distributed memory parallel processing*.

MPI has been used to great success in a variety of Engineering and Scientific codes where large arrays of the same type of data (eg floating point numbers) need to be exchanged between processes. However, with object oriented codes, one really needs to send *objects* (which may well be compound) between processes. MPI does provide the `MPI_Type` functionality, which allows the description of compound structures to be built up, however this is difficult to use, and doesn't handle the case where the structure contains references to other objects (eg pointers).

Object *reflection* allows straightforward implementation of serialisation (i.e. the creation of binary data representing objects that can be stored and later reconstructed), binding of scripting languages or GUI objects to 'worker' objects and remote method invocation. Serialisation, for example, requires knowledge of the detailed structure of the object. The member objects may be able to serialised (eg a dynamic array structure), but be implemented in terms of a pointer to a heap object. Also, one may be interested in serialising the object in a machine independent way, which requires knowledge of whether a particular bitfield is an integer or floating point variable.

Languages such as Objective C give objects reflection by creating class objects and implicitly including an *isa* pointer in objects of that class pointing to the class object. Java does much the same thing, providing all objects with the

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 896–902, 2003.
© Springer-Verlag Berlin Heidelberg 2003

native (i.e. non-Java) method `getClass()` which returns the object's class at runtime, as maintained by the virtual machine.

When using C++, on the other hand, at compile time most of the information about what exactly objects are is discarded. Standard C++ does provide a run-time type enquiry mechanism, however this is only required to return a unique signature for each type used in the program. Not only is this signature be compiler dependent, it could be implemented by the compiler enumerating all types used in a particular compilation, and so the signature would differ from program to program!

The solution to this problem lies (as it must) outside the C++ language per se, in the form of a separate program (classdesc)[2,5] which parses an input program and emits function declarations that know about the structure of the objects inside. These are generically termed *class descriptors*. The class descriptor generator only needs to handle class, struct and union definitions. Anonymous structs used in typedefs are parsed as well. What is emitted in the object descriptor is a sequence of function calls for each base class and member, just as the compiler generated constructors and destructors are formed. Function overloading ensures that the correct sequence of actions is generated at compile time.

Once a class definition has been parsed by classdesc, and the emitted class descriptors included into the original program, an object of that class can be serialised into a buffer object using the `<<` operator. The `>>` operator can be used to extract the value back.

```
#include "foo.h"  //foo class definition
#include "foo.cd" //foo class descriptor
...
pack_t buf;
foo a=...;
bar b=...;
foo c;
buf << a << b;  // serialise a into buffer.
buf >> c;  // unpack buffer into c. Now c == a.
```

Once a buffer is packed, it can be saved to a file, or transferred over a network connection by using its public members `data`, which points to the data, and `size` which gives the number of bytes currently in the buffer.

A variant of `pack_t` is the `xdr_pack` type, which uses the XDR library to represent the objects in a machine independent form. This is needed if the receiver of the serialised object is of a different architecture (eg endianess, or wordsize) to the sender.

The only real reason for using `MPI_Type` in MPI is to support messages of compound objects between differing processors in a heterogenous cluster. `xdr_pack` gives heterogenous cluster support "for free".

2 MPIbuf

`MPIbuf` is a derived type which inherits from `pack_t` (or `xdr_pack` if the `HETERO` flag is set). Similar to the notion of *manipulators* with the standard iostream library, `MPIbuf` has manipulators, such as `send()`:

```
buf << a << b << send(1);
```

which sends `a` and `b` to process 1.

Process 1 can receive this message with:

```
buf.get() >> a >> b;
```

Other manipulators and methods of MPIbuf implement collective operations such as broadcast, gather and scatter:

```
buf << a << b << bcast(0) >> a >> b;
```

suffices to broadcast the values of `a` and `b` to all processors.

By default, an `MPIbuf` uses the `MPI_COMM_WORLD` communicator, which includes all processes started by the MPI system. Collective operations can be restricted to subsets of processes by defining an MPI *communicator* object using `MPI_Comm_create` and assigned to the `Communicator` member of the `MPIbuf` object.

3 MPISPMD

An MPI program requires a certain amount of structure to just to setup the processes, and tear them down at the finish, before any real coding starts.

A typical MPI program might do the following:

```
main(int argc,char**argv)
{
  MPI_Init(&argc,&argv);
  MPI_Comm_size(&nprocs);
  MPI_Comm_rank(&myid);
  ... computation ...
  MPI_Finalize();
}
```

One problem that frequently occurs is forgetting to call `MPI_Finalize()`, which needs to be called from all processes. This may occur because of an error condition, or otherwise abnormal exit from the program. This is a problem, as typical MPI implementations have resources such as shared memory segments that need freeing, and the slave processes need to be terminated. On a shared high performance computing system, stray processes consume CPU resources that may be better employed by other users.

`MPISPMD` arranges for `MPI_Finalize()` to be called from its destructor. This has two advantages: `MPI_Finalize()` will be called as soon as the `MPISPMD` leaves its scope. Also, the destructor will be called if an exception is raised, leading to cleaner error handling. Unfortunately, if the program terminates from a conventional low level call to `abort()`, the cleanup will not take place.

The above piece of code can be replaced by:

```
main(int argc, char** argv)
{
  MPISPMD C(argc,argv);
  ... computation ...
}
```

The number of processes and processor ID can be queried from MPISPMD::nprocs and MPISPMD::myid respectively. Note that you should not declare the MPISPMD object as a global variable, as some MPI implementations object to `MPI_Finalize()` being called after `main()` exits. If you need to refer to the MPISPMD object from global scope, declare a global pointer to MPISPMD, and initialise this from with `main()`:

```
MPISPMD *C;
main(int argc, char** argv)
{
  MPISPMD comp(argc,argv); C=&comp;
  ...
```

4 MPIslave

MPISPMD gives rather minimal support, as the SPMD programming model is already implicit in MPI. Another programming model which MPI is often put to is master-slave processing. Setting up the structure of a master-slave program is very tedious and error prone. The `MPIslave` class is designed to make master-slave algorithms simple to program.

When a `MPIslave` object is instantiated, a slave "interpreter" object is instantiated on each process to receive messages from the master. As `MPIslave` needs to know the type of object to be instantiated on the slave processes, it is implemented as a template, with the type of slave object passed as the template parameter.

A message sent to the slave process starts with a method pointer of type: `void (S::*)(MPIbuf&)` where `S` is the slave object type, followed by the arguments to be passed. That method of the slave object is then called, with the arguments passed through `MPIbuf` argument, and any return values also passed through `MPIbuf` argument:

```
struct S
{
  void foo(MPIbuf& args)
```

```
  {
    int x,y,r;
    args >> x >> y;
    ...
    args.reset() << r;
  }
};

main(int argc, char** argv)
{
  MPIslave<S> C;
  MPIbuf buf;
  int x=1, y=2;
  buf << &S::foo << x << y << send(1);
}
```

When the `MPIslave` object is destroyed on the master process, it arranges for all the slave objects to be `MPI_Finalized()` and destroyed also.

`MPIslave` also has features for managing a pool of idle slaves:

```
MPIslave<S> C(argc,argv);
vector<job> joblist;
for (int p=1; p<C.nprocs && p<joblist.size(); p++)
   C.exec(C << &S::do_job << joblist[p]);
while (p<joblist.size())
  {
    process_return(C.get_returnv());
    C.exec(C << &S::do_job << joblist[p++]);
  }
while (!C.all_idle())
  process_return(C.get_returnv());
```

5 Performance

Serialisation within classdesc is very efficient, as the relevant class descriptor functions are all inlined. The performance is comparable with that of simply copying the data into the buffer object. Using the MPIbuf mechanism incurs the overhead of copying into, and back out of an extra buffer, which is particularly wasteful if only one object is being sent. Clearly, within performance critical code, Classdesc MPIBuf operations may be replaced manually by an equivalent sequence of `MPI_Send()`s and `MPI_Recv()`s. As with any optimisation technique, this should be performed after profiling the code to ensure benefit.

At the time of writing, Classdesc has only been deployed in one production code (Eco-Tierra) [4,3]. Figure 1 shows the speedup curves on two machines: *Napier*, a 28 processor SGI Power Challenge, which is now a rather dated SMP

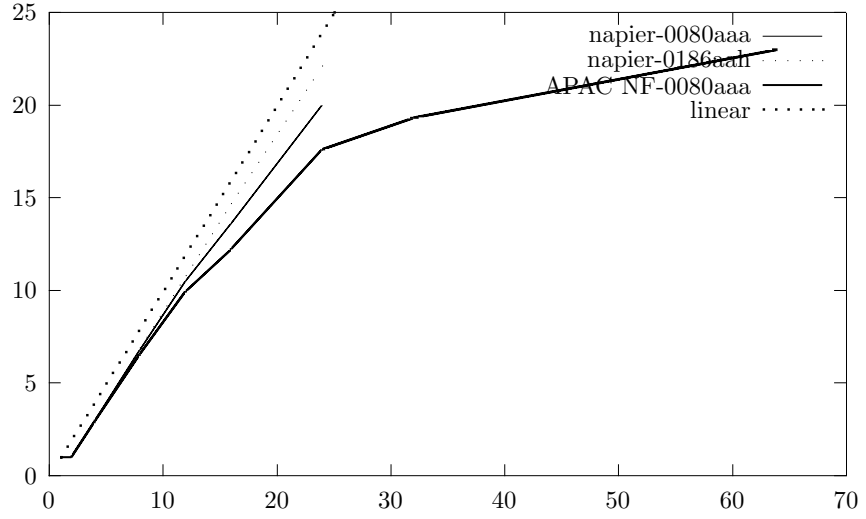


Fig. 1. Speedup curves for Eco-Tierra single site entropy jobs. Two organisms (0080aaa and 0186aah) were run on Napier, and one (0080aaa) on the APAC National Facility.

architecture, and the *APAC National Facility*, a Compaq SC cluster, consisting of 4 processor nodes coupled by a Quadrics switch.

0186aah is a larger job than 0080aaa, so has more parallelism. It is still scaling linearly at 24 processors on Napier. The constant offset from linear scaling is due to the fact that the master thread does very little, if any computation. For some reason, the 0186aah job is not running correctly on the APAC NF, so no benchmark results are available.

With 0080aaa, good scaling is obtained up to 24 processors, but on the APAC NF performance falls off after that. This is most likely an effect of the master process being unable to keep slaves busy. However, the scaling discrepancy between Napier and the APAC NF at lower process numbers indicates that network performance is partially limiting. It should be borne in mind though that at 24 processes, the APAC NF is around 6 times faster than Napier!

6 Discussion

Classdesc was originally developed as part of the *EcoLab* package to allow it to handle arbitrary agent-based models[5]. It has been used in a project studying combinatorial spacetimes[1], and in several simulations using Graphcode, a C++ framework for agent-based simulation. Graphcode considers agents to be the vertices of arbitrary hypergraphs and is equipped with a graph partitioner to allow for the efficient execution of agent-based simulations on parallel computers. To

achieve this, Graphcode uses Classdesc internally, so that objects may be serialized for transmission between cooperating processes. ClassdescMP is a natural outgrowth of Classdesc, and has been successfully employed in the Eco-Tierra project[4,3].

ClassdescMP is distributed as part of the Classdesc package, available from http://parallel.hpc.unsw.edu.au/classdesc. It should compile and be usable with any ANSI standard C++ compiler. It has been tested on a variety of operating systems, including most Unices, Windows under Cygwin and Mac OS X.

Acknowledgements. The development of Classdesc and ClassdescMP was made possible with a grant from the Australian Partnership of Advanced Computing. Benchmark results were obtained on both APAC and ac3 facilities.

References

1. Duraid Madina. Topology and complexity: From automata to agents. In Namatame et al., editors, *Complex Systems '02*, pages 141–147. Chuo University, September 2002. also to appear in *Complexity International*.
2. Duraid Madina and Russell K. Standish. A system for reflection in C++. In *Proceedings of AUUG2001: Always on and Everywhere*. Australian Unix Users Group, 2001.
3. R. K. Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 2003. (accepted).
4. Russell K. Standish. Some techniques for the measurement of complexity in Tierra. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *Advances in Artificial Life: 5th European Conference, ECAL 99*, volume 1674 of *Lecture Notes in Computer Science*, page 104, Berlin, 1999. Springer.
5. Russell K. Standish. Ecolab4. *Complexity International*, 8, 2001.

A Novel Evolutionary Approach to Linear Time-Series Forecasting Model

Prakash Vijayan[1] and S. Suresh[2]

[1]Department of Mechanical Engineering, Indian Institute Of Technology, Madras
Chennai-600036, India, v_prakash81@yahoo.com
[2]Department of Computer Science and Engineering, Indian Institute Of Technology, Madras
Chennai-600036, India, suresh_12may@yahoo.com

Abstract. This paper presents a handshake between the concepts of genetic algorithms and the forecasting problem to present a novel search based multi-phase genetic algorithm to the forecasting problem based on the time series model. The backbone concept of the paper lies in utilizing the genetic approach effectively for implementing the Autoregressive process, a linear stochastic model where a time series is supposed to be a linear aggregation of random shocks. We propose to utilize the concept of genetic algorithms to transform an initial population of random suggested solutions to a population that contains solutions approximating the optimal one. A carefully chosen fitness function acts in the capacity of a yardstick to appraise the quality of each "chromosome" to aid the selection phase. We simulated the presented approach on a Pentium IV processor and obtained results that were very encouraging.

1 Introduction and Previous Work

Quantitative methods for generating forecasts of future outcomes using statistical procedures involve the examination of current and historically seasonally unadjusted data. This knowledge can be used to extrapolate the variable of interest. It is assumed that the process is stable over the forecast time horizon, but valid only for short time forecasts. Two types of quantitative forecasting models used are time series models and causal econometric models. Time series models involve a statistical analysis, which uses only historical data of the variable. Causal models are based on statistical analysis of data for other related variables and the use of these variables to forecast the variable of interest. There are several approaches for time series modeling. Some of them include Moving average, exponential smoothing and ARIMA, which are linear in nature. These numerous forecasting methods and the empirical findings, which are often in conflict, have given no clear guidelines as to the most appropriate methods for forecasting.

Box-Jenkins method [1] is popular because of its generality as it can handle any stationary or non-stationary time series, with or without seasonal elements. The Box-Jenkins method for selecting an appropriate ARIMA model for estimating and forecasting a univariate time series consist of identification and testing and application. Although performance is primary concern, the choice of the Box-Jenkins

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 903–910, 2003.
© Springer-Verlag Berlin Heidelberg 2003

model is due to its sound theoretical basis and the numerous research publications available. Box-Jenkins' ARIMA models are a class of linear models that were designed for linear time series. The model has been successfully applied in not only economic time series forecasting, but also for modeling the empirical dependencies between successive times between failures as discussed by Walls and Bendell [2]. The model gives satisfactory predictive performances as shown by Ho and Xie [3]. The model has been successfully used and tested for International tourist demand forecasting. This has been discussed in large and compared with different methods by Guerts and Ibrahim [4], Martin and Witt [5]. A typical problem in the ARIMA model for a time series is to estimate the coefficients of the Auto Regressive (AR) and the Moving Averages (MA) processes. Though, it is possible to fit a variety of linear models for a time series, parsimony is one important factor, which need to be considered. The number of terms used in the linear model should be a minimum. In the existing method, as demonstrated by Box Jenkins [1] in the identification stage, sample correlograms, the autocorrelation and partial autocorrelation coefficients are plotted to identify the order of the AR model. However, for higher order models, identification becomes tedious and difficult. An empirical investigation on the accuracies of the different forecasting methods has been discussed by Makridakis, S and Hibon, M[6]. The accuracy of a forecasting method is determined by analyzing the forecast error, which is defined as the actual value minus the forecast (or fitted) value of the variable for time period t:

$$e_t = A_t - F_t$$

where, A_t the actual value at time t; F_t the forecast value at time t + 1.

To the best of our knowledge, this is the first reported genetic approach to solve this problem. The field of genetic algorithms has a large amount of prior literature. Basics of genetic algorithms are presented in [9].

2 Auto Regressive Process

In the Autoregressive process, a linear stochastic model, a time series is supposed to be a linear aggregation of random shocks. However, parameter parsimony is an important criterion in choosing the right model. The stochastic process, as a linear model can be represented as

$$\overline{Z}_t = e_t + \phi_1 a_{t-1} + \phi_2 a_{t-2} + \phi_3 a_{t-3} + \phi_4 a_{t-4} + \ldots\ldots\ldots$$

The linear process represents $\overline{Z}_t$ as a weighted sum of present and past values of the white noise process a_t. The value $\overline{Z}_t$ is the deviation from the mean as the process is stationary and is represented by $\overline{Z}_t = Z_t - \mu$.

In the case, where the first 'k' weights are non zero, the process becomes an Autoregressive process of order k, represented by AR(k) or ARIMA (k,0,0). The process can be written as,

$$\overline{a}_t = e_t + \phi_1 a_{t-1} + \phi_2 a_{t-2} + \phi_3 a_{t-3} + \ldots\ldots\ldots\phi_k a_{t-k}$$

In the above process, the finite sets of weights $\phi_1, \phi_2, \phi_3 ..... \phi_k$ are non-zero.

The first order Auto Regressive process or the Markov process can be represented as

$$a_t = \mu + e_t + \phi_1 a_{t-1}$$

For stationarity of the process,

$$-1 < \phi_1 < 1$$

Similarly, for a second order Auto regressive process, represented as

$$a_t = \mu + e_t + \phi_1 a_{t-1} + \phi_2 a_{t-2}$$

For stationarity of the process,

$$-2 < \phi_1 < 2 \text{ and } -1 < \phi_2 < 1$$

The permissible values of ϕ_1 and ϕ_2 are for a stationary Autoregressive process as shown by Makridakis, Wheelwright and McGee[7], must fall in the triangle as shown in the figure 1.

3 The Genetic Approach

The algorithm presented in this paper apart from using only the genetic algorithm uses a search based multi-phase genetic algorithm that proceeds towards the optimal solution in an excellent manner. We start with a genetic algorithm by setting the ranges of the coefficients and operate in that range.

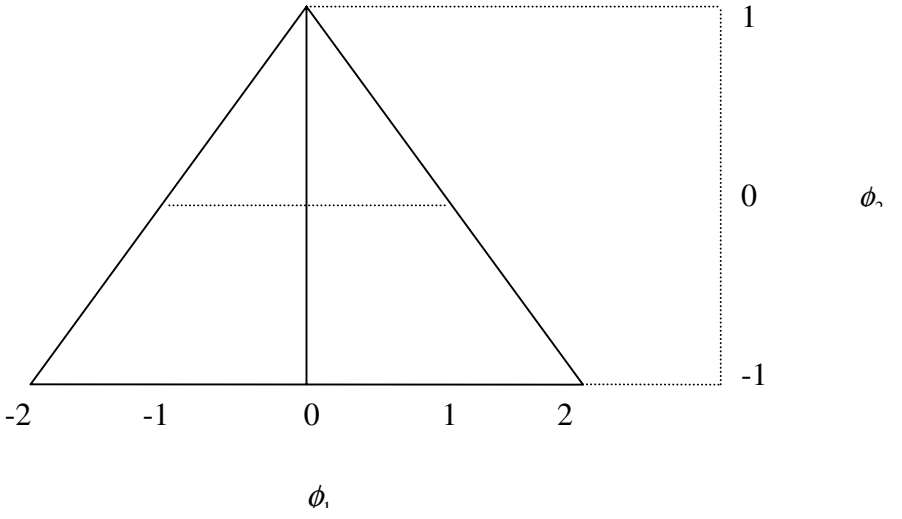


Fig. 1. The triangle shows the range of values that ϕ_1 and ϕ_2 for a stationary Autoregressive process.

After simulating this genetic algorithm for a sufficiently large number of iterations we change the range of values that can be taken by the coefficients in the following manner:

- Observe the best ten solutions based on their fitness values.
- Obtain the maximum and minimum values taken by each of the coefficients and set them as the maximum and minimum values that can be taken by the respective coefficients during the next execution of the genetic algorithm.

This approach utilizes an intelligent and random methodology and therefore proceeds towards the optimal solution in an intelligent manner.

It is however essential to note that it is very important that we execute the first few genetic algorithms for sufficiently large number of iterations. The reason for such a claim is attributed to the fact that an improper prediction of range at the end of the first few executions may result in vast deviations of the solution from the optimal solution and therefore affecting the performance badly.

3.1 Encoding

The encoding phase essentially establishes a mapping over of the solution into a form that allows the genetic algorithm to analyze it. The encoding provided is an array named *Coeff*[N], where N represents the number of past values used for forecasting. Each element of the array consists of the coefficient taken by each of the past values in the forecasting equation. For example:
An array *Coeff[3]* having the values {1.25, 0.35, 0.11) represents the equation of the form:

$$a_t = 1.25a_{t-1} + 0.35a_{t-2} + 0.11a_{t-3}$$

3.2 Formation of the Initial Population

The formation of the initial population determines the portion of the solution space explored by the genetic algorithm. Though the size of the initial population is up to the user, it should not be too small as it would greatly hinder the performance of the algorithm, while at the same time, a population too large in size would essentially increment the runtime as well since we would have to take care of quite a lot of chromosomes. Through extreme experimentation we found that an initial population of size equal to 10N, where N depicts the number of previous values used for forecasting gave us the best solution. We generate the initial population in a random manner but ensuring that the coefficients lie in the appropriate range. We generate 10*N-3 chromosomes using the random approach. The other three chromosomes are generated by setting all the coefficients in the array Coeff[N] to the maximum, minimum and the mid value of the ranges of the respective coefficients.

3.3 Genetic Operators

The genetic operators are applied to the population at hand to produce offspring that are, with a high probability, better than their parents. These operators ensure proper evolution of the current population towards the desired solution.

3.3.1 The Mating Operator

For the mating process to take place, we would require the two parents to interact. The parents selected for mating are randomly chosen from the set of given population at hand. However the probability of selection of any chromosome is directly proportional to its fitness value. We adopt a one-point crossover strategy and the crossover point is selected randomly. The following example shows how mating is performed on two chromosomes:

Parent1: {1.5 | 0.5,0.12}
Parent2: {2.3 | 0.88,0.01}

When the above two parents are crossed over at the point shown by the '|' they produce two offspring:

Child1: {1.5, 0.88, 0.01}
Child2: {2.3, 0.5, 0.12}

3.3.2 The Mutation Operator

The second genetic operator, the mutation operator, gives the genetic algorithm a greater touch of randomness. The randomness makes sure that no specific kind of pattern is lost at any point of time in the population. As a result of the mutation operator, we have the power to redirect an evolution that is converging to a local maximum, and place it on a path to a better solution. We assign a probability of mutation to each candidate of the population in a manner that ensures that a higher probability of mutation would be associated with those chromosomes that are of a low quality as far as the fitness value is concerned.

3.3.3 Quality Enhancement Operator (QEO)

This is the last genetic operator that we employed in an unconventional manner. This operator is primarily concerned with improving upon the quality of the chromosome on which it is operated. The QEO is employed on ¼*(Population size) number of randomly chosen chromosomes, the probability being same as the mutation probability. The operator functions in the following manner:

Calculate the value of the function:

$$a_t = Coeff[0]a_{t-1} + Coeff[1]a_{t-2} .... + Coeff[N]a_{t-N-1}$$

If the value is more than the expected value for most of the given data, then randomly select any positive coefficient and decrement its value by 25% of its original value. It is important to note that by doing this, we ensure with a higher probability that the offspring produced is of a superior quality than its parent. This operator greatly enhances the efficiency of our algorithm.

3.4 Fitness Value

The sole purpose of the fitness function is to implement the "survival of the fittest" strategy. The fitness value in our case is solely concentrated on the minimization of deviation of the solution at hand from the actual solution. This optimization criterion is framed aptly in the form of a function that evaluates the chromosomes. The

function *Calculate_fitness(Chromosome C)* calculates the fitness value of the chromosome C in the following manner:

```
Calculate_fitness(Chromosome C)

Float MaxDev, CurrDev, Fitval
MaxDev = Maxdeviation()
For each series of data available
begin
      Calculate  a_t = Coeff[0]a_{t-1} + Coeff[1]a_{t-2} .... + Coeff[N]a_{t-N-1}
      CurrDev = CurrDev + (a_t - a_t^*)^2 , where a_t^* depicts the
desired value of a_t .
End for

Fitval = 1 - CurrDev / (Maxdev* Number of series of data)

Return FitVal
End Function
```

The function *Maxdeviation()* calculates the square of maximum possible deviation of any chromosome which is calculated by substituting the upper bound values of each of the coefficients. Note that the series of data given can be formed by splitting the given series into smaller data series of requires length. For example:

In case of N=3, a data series 1,2,3,4,5,6,7,8,9,10 can be split into the following data series: {1,2,3,4}, {2,3,4,5}, {3,4,5,6}, {4,5,6,7}, {5,6,7,8}, {6,7,8,9} and {7,8,9,10}.

3.5 The Main Algorithm

The main algorithm of any phase of the genetic algorithm is as follows:

```
GA(Ranges of Coefficients)
Create initial population
while solution with the required fitness value has not been found do
        Choose Parents for mating process
        Pair up Parents Randomly for the mating process
        Carry out Mating
        Add Offspring to population
        Carry out Mutation
        Apply the QEO
End While
End Function
```

Note that, if at any point of time the population size exceeded the limit of 2000 chromosomes we filtered the population in the following manner:

- Retain the top 50% of the chromosomes.
- From the bottom 50% of the chromosomes select 50% of them in random manner.

It was observed that such a policy adopted for controlling the population was highly lucrative.

4 Experimental Section

We implemented our algorithm in C language on a Pentium IV processor. We tested the working of our algorithm on the IBM common stock closing prices presented in [11]. We allowed our algorithm to proceed till the best solution presented by our algorithm remained stable for a significantly long time.

Before executing our algorithm we had to decide upon the various parameters of our genetic algorithm like the rate of mating, rate of mutation, size of initial population etc. By varying these parameters we were able to decide upon the best values for these parameters, which struck an optimal balance between the run-times, and the fitness values of the best solutions obtained.

The forecast equation obtained by our algorithm is:

$$\nabla z_t = a_t + 0.125(\pm 0.035)a_{t-1} - 0.003(\pm 0.001)a_{t-2}$$

The equation as predicted by the methodology proposed in [1] was:

$$\nabla z_t = a_t + 0.09(\pm 0.05)a_{t-1}$$

It is very clear that the prediction of any value is mainly based on the previous two values as the coefficients associated with the remaining time lags are very close to zero. The residual variance of our algorithm is 43.45, which is a significant improvement over the variance of 52.2 presented in [1]. Figure 2 shows the first differences obtained by our algorithm and the actual first differences. It is significant to note that our forecasts were very close to the desired values. The complexity of executing the algorithm was very less and the run-times were also impressive. Thereby our algorithm stood well on both the criteria as mentioned in section 1, as far as the forecasting accuracy and the ease of interpreting results is concerned.

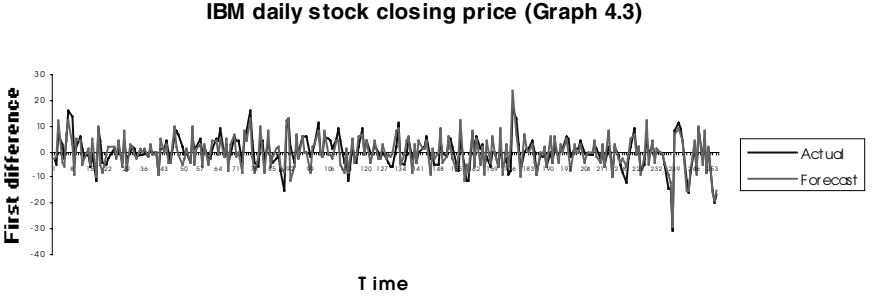


Fig. 2. The first differences obtained by our algorithm and the actual first differences

5 Conclusion

The search based multi-phase genetic algorithm proposed in this paper has proved to be efficient in estimating the coefficients to very high accuracy. The results obtained were highly encouraging and the complexity associated with the execution was also very low. This paper opens up a whole new methodology for fixing the coefficients in the ARIMA model. Further research in this field may be oriented towards developing evolutionary approaches for identification and fixing the coefficients for the ARIMA model. There is also a tremendous scope of improvement in developing new crossover, mating and the fitness operators.

References

[1] Box GEP, & Jenkins GM. Time series analysis: Forecasting and control, Revised ed. Holden-Day, San Francisco,1976.

[2] Walls, L. A., & Bendall, A. (1987). Time series methods in reliability. Reliability Engineering, 18, 239–265.

[3] Ho, S. L., & Xie, M. (1998). The use of ARIMA models for reliability forecasting and analysis. *Computers and Industrial Engineering, 35*

[4] Geurts, M.,& Ibrahim, I. (1975). Comparing the Box–Jenkins approach with the exponentially smoothed forecasting model application to Hawaii tourists. Journal of Marketing Research,12, 182–188 .

[5] Martin, C. A., & Witt, S. F. (1989). Forecasting tourism demand: A comparison of the accuracy of several quantitative methods. International Journal of Forecasting, 5,7–19.

[6] Makridakis, S., & Hibon, M. (1979). Accuracy of forecasting: An empirical investigation. Journal of the Royal Statistical Society A, 142,97–145.

[7] Makridakis. S, Wheelwright SC, McGee VE, Forecasting: Methods and applications, John Wiley & sons, 1983

[8] Kenneth A. De Jong and William M. Spears , "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," In *International Workshop Parallel Problem Solving from Nature*, University of Dortmund, pages 38–47(1990).

[9] Kenneth A. De Jong and William M. Spears (1992), "A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms," In *Annals of Mathematics and Artificial Intelligence Journal*, v5, n1, pages 1–26.

[10] Pinaki Mazumder, Elizabeth Rudnick ,"Genetic Algorithms for VLSI Design, Layout and Test Automation ," Prentice Hall PTR, (2002).

[11] Box GEP, & Jenkins GM. Time series analysis: Forecasting and control, 3rd ed., Prentice Hall, pp. 542–543, IBM Common stock closing prices.

Design and Implementation of XML DBMS Based on Generic Data Model*

JongSeon Lim[1], Sung yoon Bang[2], and Kyung-Soo Joo[1]

[1]Dept.of Computer Science and Engineering, College of Engineering SoonChunHyang Uni., P.O Box 97, Asan, ChungNam, Korea, 336-745
ronmer@chol.com, gsoojoo@sch.ac.kr
[2]Hanseo University Computer Center, SeoSan, ChungNam, Korea, 356-706
sybang@hanseo.ac.kr

Abstract. Nowadays XML is used for exchanging information in e-Commerce, especially B2B. Necessity of XML DBMS has being increased to efficiently process XML data. So a lots of database products for supporting XML are rapidly appeared in the market. In this paper, we made an XML DBMS system based on Generic Data Model. First we developed XML Adaptor based on Generic Data Model and added it on relational DBMS for developing XML DBMS. XML Adaptor is composed of Query Converter and XML Repository System. The Query Converter parse commands that are for XML data manipulation and then call the relevant component of XML Repository System for relational database operation. The XML Repository System handles relational database operations such as create, delete, store, and etc. In this way we can use a relational DBMS for manipulation XML data. Therefore we can build more economically XML DBMS.

1 Introduction

In order to support XML data, current mainstream relational DBMS uses tools to convert other data formats to and from XML, which causes slow-down in DBMS processing speed and other problems. Thus we are likely to see increased demands for XML DBMS that can store or load XML data in its original form. Also, since there are many customers who want single DBMS platform, relational DBMS with XML support is likely to become general trend rather than pure XML DBMS. Relational DBMS currently takes up more than half the current market and they have significant difficulties in transforming to the new XML DBMS. In order to overcome these limitations, enterprises can employ two kinds of method of either converting the relational DBMS to XML DBMS or combining relational DBMS and XML together. For the former, DBMS itself is transformed inside and out and costs a great amount. But all the powerful features of XML DBMS in XML document processing. For the latter, relational DBMS adapts support for XML, allowing for easy storing and

* This work was supported by the University IT Research Supporting Program under the Ministry of Information Communication of Korea.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 911–919, 2003.
© Springer-Verlag Berlin Heidelberg 2003

managing of XML documents. However, it will not be able to utilize all the powerful XML document processing features the former has.

In this research an XML DBMS is designed on top of a relational DBMS as an XML adaptor. Causing no modifications on the existing relational DBMS, it will also be able to perform as an XML DBMS. This can bring significant benefits in system implementation with least expenses. Section 2 describes relevant researches and technologies, section 3 XML DBMS design, section 4 XML DBMS implementation and lastly in section 5, we will draw conclusion for this research.

2 Related Works

2.1 EJB

J2EE (The Java 2 Platform, Enterprise Edition) is the standard for developing multi-tier enterprise applications. In other words, J2EE simplifies the enterprise application developing process by building on top of standardized and modular components, a multi-tier enterprise application, providing full service to the respective component and automatic application processing without complicated codes. J2EE is a platform for enterprise solutions that features simplicity, migration, extension, integration and such. Further, J2EE can be said to be a set consisting of JSP, Servlet, Java Bean, EJB, and so forth, of which EJB is the core technology[3].

EJB is a component based distributed object technology that possesses the crucial core technology responsible for abstract data and business logic in enterprise level application development. Moreover, EJB container helps the developer by providing built-in services such as automatic enterprise bean life-cycle management, state information management, security, transaction processing, and continuous processing, so that the developer only needs to be concerned with the business logic.

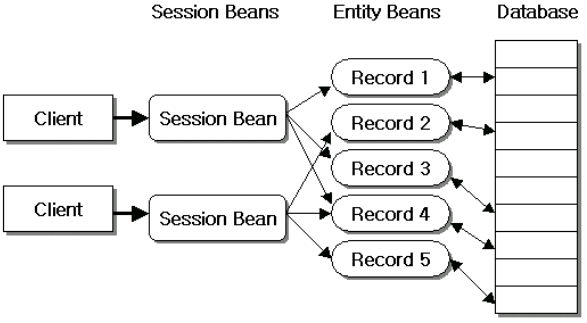


Fig. 1. Session beans and Entity Beans.

Enterprise Bean is an EJB component that clients can call and use. Enterprise Bean can be developed either as Session Bean and Entity Bean. Fig. 1. shows the relationship between the Session Bean and Entity Bean[4][5].

2.2 XML Database

An XML database is a collection of XML documents that persist and can be manipulated. Historically, documents were developed form communication between humans. With the advent of computers, documents may also be used for computer-computer communication, human-computer communication, or computer-human communication. Each of these communication paradigms has is own requirements, such as aesthetics, tolerance for ambiguity, preciseness, and flexibility[1][2].

XML documents tend to be either document-processing-oriented or data-processing-oriented. Document-processing-oriented documents are those in which XML is used form its ability to capture natural(human) languages, such as in user's manuals, static Web pages, and marketing brochures. They are characterized by complex or irregular structure and mixed content and their physical structure is important. The processing of the document is focused on the final presentation of the information to the user; thus, they may be called presentation-oriented documents. Data-processing-oriented documents are those where XML is primarily for data transport. These include sales orders, patient records, and scientific data. The physical structure of data processing-oriented documents, such as the order of elements or whether data is stored in attributes or sub elements, is often unimportant. They are characterized by highly-regular structure with many repetitions of those data structures. The processing of the document is usually focused on its use and exchange by applications; thus, they may be called message-oriented documents. Document-processing-oriented and data-processing-oriented documents are sometimes called document-oriented documents versus data-oriented documents, respectively, but those terms are ambiguous as every document always contains data[2].

2.3 Generic Data Model

The Generic Data Model is derived from the previous data models and is used form the storage mechanisms. It provides a basic data model form a variety of applications that can also be extended using more of the ideas in the previous data models [2].

2.3.1 Operations

1) Database

The operations on the Database type are made explicit in the Generic Data Model. Table 1. are representation to database operations in Generic Data Models, and abbreviated to arguments and return values[2].

2) Document

Two operations are added to the other Data Model. "GenerateDocument", generates the document next as suggested in the W3C Data Model and a new operation "selectElement", operation is more general than the "getElementsByTagmane" of the Node-centric Data Model.

Table 1. Database operations.

Database Operation	Description
newDatabase()	Create a new database.
newDocument()	Create a new document.
storeDocument()	Store a new document, return the new document.
retrieveDocumentName()	Retrieve from database.
deleteDocument()	Delete from database the document, return true if the document was originally there.
getAllDocuments()	Retrieve from database a list of all its documents.
selectDocument()	Select from database all documents, which satisfy constraint.

3) Element

The twelve operations for Element are the same as the Simple Data Model. The operations are taken from the Simple Data Model rather than Node-centric Data Model, because the DOM Node operations are not likely to be efficient for large databases.

Operations in general data model for XML elements are shown above. There are lots more operations other than the ones shown, but operations for database, attributes, and elements are introduced here. The ones proposed in this research also applies database operations from general data model.

3 XML DBMS Design

3.1 XML DBMS Structure

XML DBMS is created using the existing relational DBMS as its infrastructure and adding XML adaptor. Fig. 2. shows the system-wide structure. Users can make queries through general data model and XML adaptor without regards of the DBMS. XML adaptor can then store, search, and delete XML documents stored in the database through SQL.

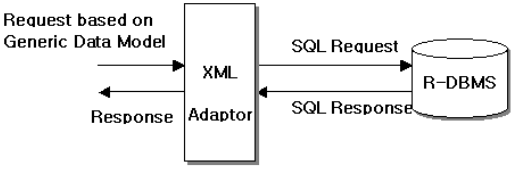


Fig. 2. XML Adaptor.

Fig. 3. is a more detailed look at the XML adaptor in Fig. 2. Queries are basically processed through general data model query language and query converter converts the general data model based queries into SQL. Further, these converted queries can use the storage management system that are built on EJB to perform store, search, delete and other tasks. In order to perform various operations in general data model, a few additional components have been added.

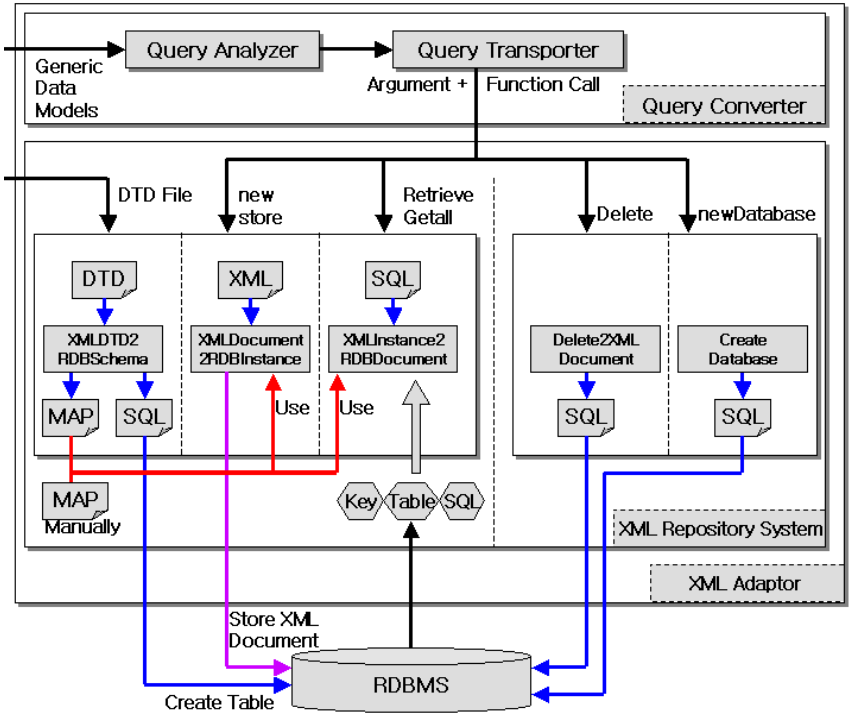


Fig. 3. Detail of XML Adaptor.

Query analyzer analyzes the query and extracts the operation part and argument, return value parts. These 3 extracted parts are then handed over to the query converter. Query converter in turn applies the operation and argument parts to convert them to the most appropriate commands or SQL and turns it over to the repository system. Repository system consists of components that embody every database operation for generic data model, and queries the database using the component called by the query analyzer.

3.2 Query Converter

Query converter consists of query analyzer and query converter. Query analyzer performs the task of analyzing generic data model based query and transporting the

result values to the query converter. General generic data model query consists of operation, argument, and return value as shown in Fig. 5.

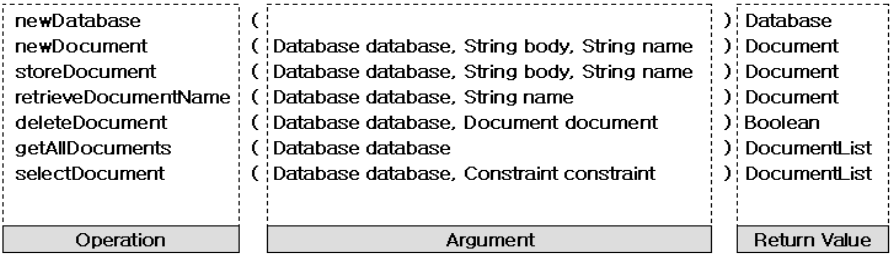


Fig. 4. Query structure of Generic Data Models.

Query converter is responsible for converting the query from the analyzer into XML repository system commands or SQL. Fig. 5. describes what format the query from Fig. 4. is converted into. Fig. 5. shows the decision process on how generic data model database operations are converted. Component to call is decided by the operation, argument, and return value. Then database name and document name are determined appropriately for the respective component.

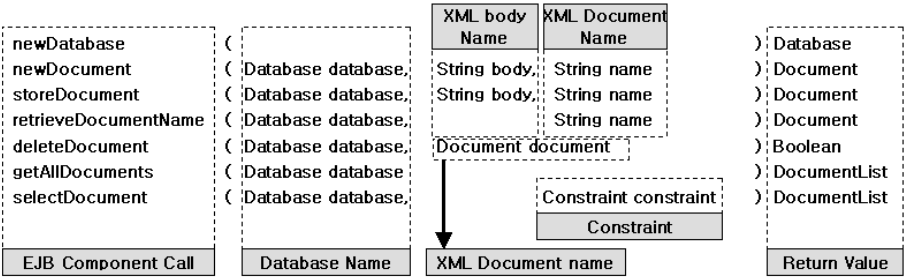


Fig. 5. Arguments of Generic Data Models.

Each operation for generic data model takes different arguments, so by separating each respective argument and calling the appropriate operation, we can store, search, and delete data through the repository system as shown in Fig. 5.

3.3 XML Repository System

XML repository system is divided into conversion, storage, search, deletion and database generation modules. Conversion module takes XML DTD as the input and converts them to relational database schemes. Storage module stores the XML documents in the relational database. Search module searches the relational database with key, table, and SQL, then documents the search results into XML. Deletion module looks up the table for a single XML document and deletes it. Database generation module generates a database. The structure of XML repository system is

designed by making the XML DBMS features into EJB components and implementing the system by assembling each of these EJB components.

3.3.1 XML Repository System Functions

XML repository system functions are listed below:
① Conversion: Provides two methods for converting XML DTD into relational database scheme. First method takes the XML DTD as its input and automatically extracts the relational database scheme. Second method is a manual conversion method where designer provides a map that displays how XML DTD and relational database scheme interrelates and based on this mapping, relational database scheme is extracted from the XML DTD.
② Storage: Stores an XML document in the relational database according to the supplied mapping.
③ Search: 3 different methods can be used. First method uses the data in the root table to search every table and generates an XML document in the mapping order. Second method searches data in a single table to generate an XML document in the mapping order. Last method generates an XML document in the mapping order by accepting SELECT command from the user for data search.
④ Deletion : Deletes an XML document in the database.
⑤ Database Generation : Generates a database.

3.3.2 Relationship between XML Repository System and EJB Components

Components used for implementing XML repository system are shown in Fig. 6. An EJB component, 'XMLDTD2RDBSchema' is used for conversion, 'XMLDocument2RDBInstance' for storage, 'RDBInstance2XMLDocument' for deletion, and 'CreateDatabase' for creating a database. All these components are assembled together to implement XML repository system.

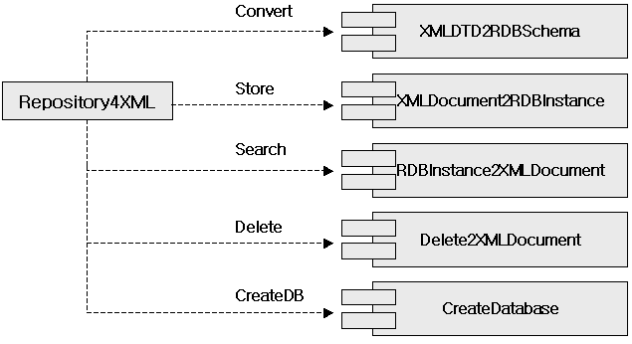


Fig. 6. EJB components of XML Repository System

4 Implement of XML DBMS

Windows 2000 Server, JBuilder 4.0, and Oracle 8i were used in implementing our system. Fig. 7. is a query for storing a new XML document in the database by using generic data model. The called component uses a storage component shown in Fig. 9. to store the XML document in the database. Similarly, Fig. 8. and 10 shows the conversion and search components that are called upon.

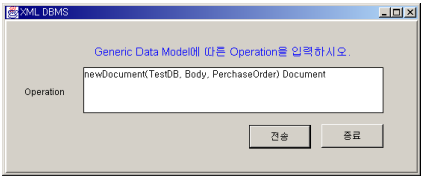


Fig. 7. "Newdocument" operation

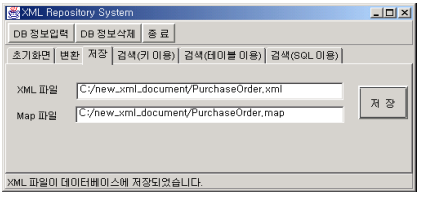


Fig. 8. Store XML document

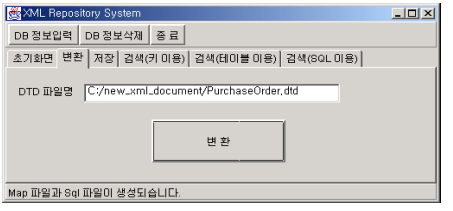


Fig. 9. Convert XML document

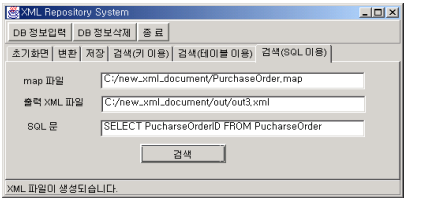


Fig. 10. Search XML document.

5 Conclusion

This research used existing relational DBMS as the infrastructure and XML DBMS was developed by adding XML adaptor for handling general data model based XML document. XML adaptor is composed of i) a converter that parses generic data model type command and calls appropriate component, and ii) XML repository system that performs the actual tasks on the relational database. XML repository system is built by assembling 5 components (i.e. conversion, storage, search, deletion, and database generation) that are based on EJB.

Using XML adaptor introduced in this research, XML data handling is made possible without having to replace the existing relational DBMS, ultimately lowering the cost for implementing an XML DBMS and enabling the usage of low-cost relational DBMS.

References

1. Harvey Deitel, XML How to Program, Prentice Hall, 2002
2. Mark Graves, Designing XML Databases, Prentice Hall PTR, 2002
3. Vlada Matena; Beth Stearns, Applying Enterprise JavaBeansTM Component-Based Development for the J2EETM Platform, Sun, 2001
4. Lee Jung-soo, Jung Sang-hyuk, Joo Kyung-soo, "A Design and Implementation of XML Repository System based on EJB component ", Korean Society for Internet Information
5. Lee Jung-soo, Joo Kyung-soo, "Designing Components for mapping from XML DTD to ORDB Schemas", Journal of Information Technology Applications & management, Vol.9 No.1 March 2002, pp72–83

Jong-Seon Lim received his B.S. degree from Dept. of Computer Science, Korea Uni. in 1997 and obtained M.S. degree from Dept. of Computer Science, Soonchunhyang Uni. in Korea.

Dept. of Computer Science, Graduate school Soonchunhyang Uni.
E-mail : ronmer@chol.com

Sung yoon Bang received his B.S. degree from Dept. of Computer Science, Korea National Open Uni. In 1996 and obtained M.S. degree from Dept. of Computer Science, Hoseo Uni. In 1998 and obtained Ph.D degree from Dept. of Computer Science, College of Engineering Soonchunhyang Uni. In 2003.

Hanseo University Computer Center manager
E-mail : sybang@hanseo.ac.kr

Joo Kyung-Soo received his B.S. degree from Dept. of Mathematics, Korea Uni. In 1980 and obtained M.S. degree from Dept. of Computer Science, Korea Uni. In 1985 and obtained Ph.D degree from Dept. of Computer Science, Korea Uni. In 1993

Dept. of Computer Science, College of Engineering Soochunhyang Uni. Prof.
E-mail : gsoojoo@sch.ac.kr

Developing a Unified Design Methodology Based on Extended Entity-Relationship Model for XML

Mun-Young Choi [1], JongSeon Lim[2], and Kyung-Soo Joo [3]

Dept.of Computer Science and Engineering, College of Engineering SoonChunHyang Uni.,
P.O. Box 97, Asan, ChungNam, Korea, 336-745
[1] griffin@hyejeon.ac.kr, [2] ronmer@chol.com, [3] gsoojoo@sch.ac.kr

Abstract. Nowadays an information exchange on XML such as B2B electronic commerce is spreading. Therefore the systematic and stable management mechanism for storing the exchanged in formation is needed. For this goal there are many research activities for connection between XML application and relational database. A unified modeling methodology need to store the XML data on the variety database. In this paper an unified design methodology based on EER data model for XML applications is proposed. For this goal, first a XML modeling guideline for XML DTD based on ER data model is introduced. Second a database modeling methodology for storing on object relational database is proposed.

1 Introduction

XML is a text form planned to utilize structured document in web. XML is embossed highly as the standard language about data exchange. XML has many advantages for compatibility and extensity. One of the important tasks of XML is the easy exchange of different data and information in B2B electronic commerce. The important exchanged data or information are stored on existent relational database system, and it is important that the data or information is automatically processed to make XML file.

In this paper a XML modeling guideline for XML DTD based on ER data model is introduced. A database modeling methodology for storing on relational database is proposed. XML document structure is described by using DTD, which is defined by standard, and data should be preserving states as much as possible in conversion by XML document[1], [2].

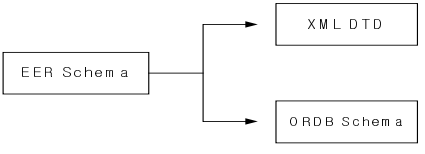


Fig. 1. XML modeling and data modeling that use EER schema

The references described about XML modeling and data modeling way in EER schema are[3], [4] But, there is no way that integrate both of two. Therefore, XML modeling and data modeling way in EER schema with Fig. 1 are described in this paper.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 920–929, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 XML DTD Design Method of ER Base

In this section we will unite the proposed translations from conceptual schema to DTD into an guideline to be applied to any given conceptual schema. The guideline may be validated by applying it to the sample ER schema in section 3 to generate a DTD. Moreover we will comment on the quality of the ER to DTD transformation by analyzing how much information is lost if a retransformation to ER is performed.

The complete guideline in pseudo-code is depicted in Fig. 2. As root element we use an element corresponding to the whole database represented by the schema name. Elements for entities at higher levels are added as direct subelements of the root with the objective to preserve as many constraints from the schema as possible. These elements include elements for entities that are not functionally dependent on other entities as well as relationships of higher cardinality and arity.

In step 3 we start with the detailed contents definition of the elements from the previous step. Whenever we need new elements in the process they are appended to the list of elements in need of detailed definition. The process for each element includes a straight-forward mapping of entity attributes as described in (a), (b) and (e). Subelements are included for relationships to weak entity types (step (c)) and other functionally dependent entities (step (d)) such that a validating XML parser will be able to verify these constraints on a particular document.

In step (f) 1:1-relationships with total participation of the current entity and partial participation of the opposite side are realized by adding all the relationship attributes to the current element as well as an IDREF attribute for the opposite entity type.

Finally in step 4 elements for n:m- or k-ary relation-ships ($k > 2$) are defined as direct subelements of the root element of the document. The references to the participating entities are realized by IDREFs which can only enforce the existence of matching XML elements but not correctness of the desired element type.

A major restriction is that participation of entities in relationships modeled by IDREF attributes cannot be reconstructed. This is a major restriction since several relationships could not be set up properly in a reconstruction process, the relationship Works_On can be reconstructed as binary n:m-relationship but in general it cannot be determined between which entities it holds. To overcome this for a particular XML document one may either obtain the entity information from the key attribute of the XML element pointed to by the reference or use special annotations maintaining the name of the entity participating in a relationship.

Also relationship attributes of 1:1-relationships that were included in the entity element of the total participation side cannot be distinguished from regular attributes of that entity without additional annotations. Weak entities cannot be reconstructed as weak entities since they were modeled in the same way. Role names from the ER schema were not included in the DTD and can therefore not be obtained from DTD or document.

3 Example Create XML DTD from ER Schema

In this section we start with a motivating example which illustrates the translation of a well-known ER schema to a DTD. In the second part translation rules from this example are derived and ideas for translating additional modeling constructs are developed..

1. define <SchemaName> to be the root element
2. subelements with cardinality * of root element <SchemaName> are:
   (a) elements for all entities not occuring on the n-side of any 1:n-relationship
   (b) elements for all binary n:m-relationships and all k-ary relationships with k > 2
   (c) elements for all entities occuring only on n sides and only partially in relationships
3. while not all entity elements used are defined in the DTD define the next element by (favor entities having identifying relationships to weak entities):
   (a) introducing a subelement for each composite attribute
   (b) introducing a subelement with cardinality * or + for each multivalued attribute
   (c) defining a subelement for each relationship with weak entity types (cardinality as in the ER schema)
   (d) introducing a subelement for each 1:n-relationship where the current element is on the 1-side with cardinality * or +
   (e) defining XML attributes for all simple-type attributes of the current element (key attributes as ID, others as CDATA)
   (f) defining XML attributes for all attributes of 1:1-relationships where the current element participates totally and the other side is partial; define an IDREF attribute for the opposite side of such a relationship; following the same process, if the opposite side entity is a previously seen entity, even if participation is total
   (g) merging the opposite side entity and all relationship attributes into this element for all 1:1-relationships with total participation on both sides, if the opposite side entity is a new entity
   (h) for each relationship element obtained from (c) or (d):
   h1. defining a subelement for the opposite side of the relationship, if it is a new entity

   h2. defining an IDREF attribute for the opposite side of the relationship, if it is a previously seen entity
   h3. defining attributes for all relationship attributes as above with entity attributes
4. for all relationship elements from n:m- or k-ary relationships with k > 2:
   (a) introducing an IDREF attribute for all entities participating in this relationship
   (b) defining attributes for all relationship attributes as above with entity attributes

Fig. 2. Guideline to generate a DTD from an ER schema

In this section the translation is illustrated by applying it to a well-known ER schema of a company database which is shown in Fig. 3. The DTD generated by our guideline can be seen in Fig. 4.

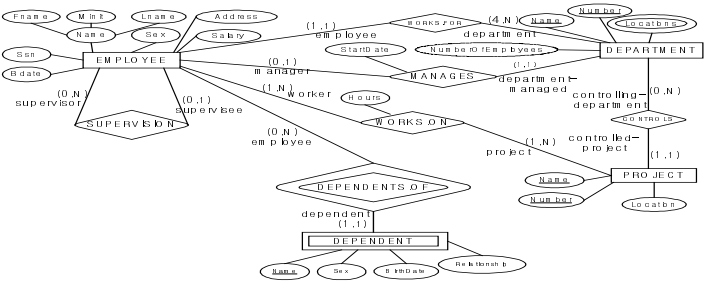


Fig. 3. Example ER schema of a company database, schema name Company

At first, there is a root element Company that represents the entire company database. Since entity department is the only entity that is not functionally dependent on another entity it is defined as subelement Department of the root element. Also the only n:m-relationship in the schema appears as direct subelement Works_On of the root element of the document. The structure of Department is given by subelement Location for the multivalued attribute locations and by subelements for the two functional relationships. Their XML cardinality (either * or +) can be derived from the participation of the opposite side in that relationship.

The 1:1-relationship manages is modeled by an IDREF attribute Manager for the opposite side and an attribute for the relationship attribute Start-Date. Subelements of the relationship elements Controls and Works_For introduce the elements for the dependent entities project and employee.

```
<!ELEMENT Company (Department*, WorksOn*)>
<!ELEMENT Department (Location*, Controls*, WorksFor, WorksFor, WorksFor, WorksFor+) >
   <!ATTLIST Department NameNumber ID #REQUIRED>
   <!ATTLIST Department ManagesStartDate CDATA #IMPLIED>
   <!ATTLIST Department Manager IDREF #REQUIRED>
<!ELEMENT Controls (Project) >
<!ELEMENT WorksFor (Employee) >
<!ELEMENT Location EMPTY>
   <!ATTLIST Location Name CDATA #IMPLIED>
<!ELEMENT Project EMPTY>
   <!ATTLIST Project NameNumber ID #REQUIRED>
   <!ATTLIST Project Location CDATA #IMPLIED>
<!ELEMENT Employee (Name, DependentsOf*, Supervision*) >
   <!ATTLIST Employee Ssn ID #REQUIRED>
   <!ATTLIST Employee Bdate CDATA #IMPLIED>
   <!ATTLIST Employee Adress CDATA #IMPLIED>
   <!ATTLIST Employee Sex CDATA #IMPLIED>
   <!ATTLIST Employee Salary CDATA #IMPLIED>
<!ELEMENT DependentsOf (Dependent) >
<!ELEMENT Supervision EMPTY>
   <!ATTLIST Supervision Supervisee IDREF #REQUIRED>
<!ELEMENT Dependent EMPTY>
   <!ATTLIST Dependent Name CDATA #REQUIRED>
   <!ATTLIST Dependent Sex CDATA #IMPLIED>
   <!ATTLIST Dependent BirthDate CDATA #IMPLIED>
   <!ATTLIST Dependent Relationship CDATA #IMPLIED>
<!ELEMENT WorksOn EMPTY>
   <!ATTLIST WorksOn Project IDREF #REQUIRED>
   <!ATTLIST WorksOn Worker IDREF #REQUIRED>
   <!ATTLIST WorksOn Hours CDATA #IMPLIED>
```

Fig. 4. DTD for the company schema

Features illustrated by the definition of Employee are the subelement Name for a composite attribute, the modeling of weak entity types similar to regular 1:n-relationships and modeling relationships to the entity itself. According to the type of supervision we use a subelement as before. The difference is in modeling the structure of subelement Supervision: we do not use any further subelements in order to avoid redundant information in the XML file since entity employee is already taken care of. Thus we use an IDREF attribute to point to the particular supervisees. Since we have subelements of Supervision for each supervisee we need only one such attribute.

Each strong entity type E of the ER schema is translated into an XML element of the same name E in the DTD. Each of the simple attributes Ai of E, which are usually assumed to be of a simple data type in ER schemas, is modeled as an XML attribute Ai belonging to element E. Key attributes in the schema should be translated to ID type attributes with the REQUIRED specification. Where as other attributes should be declared as CDATA with either IMPLIED or REQUIRED specification depending on whether they may be null or not. The problem that keys are locally unique whereas IDs are globally unique can be overcome by using the entity name as prefix to the key value in the document instance. Similarly we can use composite strings for composite keys. Of course this is still not an equivalent translation since key datatypes are not preserved and a proper retransformation from DTD to conceptual schema will have to rely on such additional conventions but it is probably the best we can do in the presence of DTDs. Each composite attribute A in the ER schema will be translated into a subelement A of E. The subelement A itself can be defined in much the same way as we developed E.

Each weak entity type W of the ER schema with owner entity type E is modeled by a subelement W of the identifying relationship between E and W. This relationship is itself modeled as subelement of E with cardinality. The identifying relationship itself is modeled as regular 1:n-relationships which are described below. All attributes of

A 1:1-relationship between entity types S and T where both participations are total should be translated by merging the corresponding XML elements S and T into one element. Attributes of the relationship are added to the XML element type.

Each regular binary 1:n-relationship R with entity S on the 1 side and T on the n side of R can be represented by a subelement R of S with cardinality * or + depending on the participation of T in R. The case where S and T are the same entity type has to be treated differently. Since all instances of S will already be included we should not use a subelement of R for the n side but rather an attribute of type IDREF.

Binary n:m-relationships R are mapped to top-level elements R in the DTD. The element R itself consists of two attributes which are defined as IDREF. These references will point to a pair of elements of R in the XML document later. By using IDREF XML documents with minimal redundancy are obtained since each tuple occurs only once and is only referenced at other places.

4 Extending the Generation to EER Schemas

Since the constructs of the traditional ER model were not sufficient for advanced application modeling numerous extensions have been proposed. The enhanced ER model (or EER model for short) as used in this article has been described in detail in [5] and uses extensions to the ER model similar to the ones presented in [6] and [7]. The additional constructs available are specialization and generalization, both with partial or total participation and disjoint or overlapping, as well as categories or union types. For our purposes we need not differentiate between specialization and generalization since we work on the finished schema where both have the same meaning.

The first and probably most common form of specialization is a disjoint specialization with total participation. We use XML elements for both super- and subclass entities where the elements of the subclasses contain a subelement representing the superclass. This

mapping is possible since every object belongs to exactly one subclass and the superclass is abstract. Only subclass elements are directly included in the DTD.

For a disjoint, partial specialization we need to include superclass elements in the DTD directly as well since it may be instantiated. Thus we need elements for all entities and include the superclass element directly in the DTD with an optional subelement to be chosen from all possible subclasses. That means, if we have a superclass G with possible subclasses S1,. . . ,Sn in the current specialization we include the term <!ELEMENT G (S1|. . . |SN)?> as description of G.

5 Relational Database Design of EER Data Model Base

Relational tables could use to actuality database system from EER model can be planed. But, EER model can see that model is model situated on midstream of relation design with object intention design, therefore partial difficult items are in conversion in relational table [8].

Relation type can be made in individual table with EER model's each Entity type. If the second relation type of Mapping is 1:1 or 1:m's case, relation type using (foreign key) to come form abroad in Entity type instead of is existed in individual table the relation mark can. And individual attribute exists in relation type or it is general that relation type of n:m case has individual table. In the case of generalization, according to restriction condition, various kinds selection is available.

ER schema model's EMPLOYEE has belonged to DEPARTMENT in Fig. 3 and EMPLOYEE large number of relation between each other has belonged by 1:n relational to one DEPARTMENT. Create schema and 1 side composes schema by own attribute including basis height of 1 side in n when change this to relational schema. 1:1 relational can lift relation that chief wealth of one person manages to a DEPARTMENT by relation that only other one individual corresponds to one individual. If change this by relational schema, include basis height in one side and make schema. In the case of n:m relational compose each by relational schema when several PROJECTs that EMPLOYEE plans belong to several PROJECT list and relation between each other composes by schema too. Fig. 5 makes out relational database about the EER schema example.

6 Object Relation Database Design of EER Data Model Base

In this chapters Explain about changing EER schema to object relation database. Objective-comparison of relation data type mechanism is same as following.

① Built-in types: They have maturity and high performance because they are compiled into the ORDBMS. They are good as building blocks for other types. They are inflexible, and because they are byte-level in nature, built-in types contribute little semantic value to the data model.

② DISTINCT TYPE: This is the easiest to use of the UDT mechanisms. Each DISTINCT TYPE has the same performance as its underlying type. It is the least flexible of UDT mechanisms, but it is useful if the new type is similar to an existing one (which is surprisingly often).

③ ROW TYPE: ROW TYPE objects are reasonably easy to use and mandatory for some object-relational features (inheritance). It supports compound objects (multi-element types well). In some situations ROW TYPE limits the range of OR-SQL features that can be used in queries involving them. They are not as fast as an OPAQUE TYPE equivalent.
④ OPAQUE TYPE: It is the fastest UDT mechanism, and the most flexible, and it provides good runtime performance on the broadest range of data types.

6.1 Strong Entities

For each strong entity in the EER model, create a table in which the table's name is the same as the entity name, each entity attribute has a corresponding column in the table, and the kind of data corresponding to each attribute is represented with the corresponding data type.

Each of the entity's attributes has a corresponding table attribute of the appropriate data type. Mandatory elements are established with NOT NULL column constraints. Strong entities should have at least one candidate key: either a subset of its attributes or a system-generated value to identify the row uniquely. Of course, any other candidate keys should be constrained with a UNIQUE constraint.

Earlier in this chapter we investigated how certain entities may be analyzed in more detail using object-oriented methods. If the entity is found to possess read methods that can be used in a query, the entity is best implemented as follows:
① Use a ROW TYPE to define the table/entity structure.
② Create a table using the ROW TYPE.
③ Add whatever constraints have been identified.

Once these steps are complete, the entity read methods can be implemented using UDFs. If these read methods are sufficiently common, such extensions should be implemented in C so that a functional index can be created later.

Again, we emphasize that using a ROW TYPE to define tables complicates the task of altering a table's structure. Consequently, use this approach only when necessary: either when the table is part of an inheritance hierarchy, or when some value can be computed from the table's rows with a user-defined function.

6.2 Weak Entities

You rarely encounter a strong entity on its own. More typically, the conceptual model consists of a network of interrelated weak and strong entities. Weak entities are also represented using table; the table's name is the same as the entity type's name, and it has a set of columns that correspond to the entity type's attributes in much the same way that a strong entity's table does.

When a weak entity is transformed into a table, the columns defining the primary key of the related strong entity are added to the columns defining the weak entity's attributes. Sometimes, these extra columns are called join columns.

We use the name of the table, Strong_Entity, as the name for the new column. Using table names to represent relationships is a convenience for developers who write queries; the new column could have any name that makes sense. Sometimes the role label from the

relationship is used instead. The constraints added after the tables are created enforce the referental integrity.

6.3 Modeling Complex Relationships

All examples we have seen involve converting entities into tables, but certain kinds of relationships are also represented this way. For instance, entity relationships that are N:M in their cardinality, or n-ary relationships involving more than one or two entities, are stored as rows in a table. The general idea is to create a table that has columns that are the same as the primary-key columns of the tables involved in the relationship and rules to ensure the correctness of its data.

You might want to enforce rules over the entire relationship in the same way you enforce rules over entity tables. For example, in circumstances in which an Employee may work for multiple Departments, and a single Department will consequently have many Employees, it may be desirable to ensure that each relationship between an Employee and a Department is recorded only once. In other words, the ORDBMS can ensure that a row of values is not repeated in the relationship table. Tables recording relationships are often called resolution tables. One advantage of this approach is that it is relatively straightforward to represent more complex kinds of relationships, such as n-ary relationships and relationships with associated attributes. Handling n-ary relationships also calls for creating a table containing key columns of all related tables, and possibly extending it with additional columns to contain any attributes of the relationship.

6.4 General Rules for Table Definitions

There are several "rules of thumb" concerning how attributes are handled. These rules are common to all tables created from EER diagrams.

① NULL values in columns complicate query processing and should be avoided. Some of the most notorious and long-running "wrong answer" problems reported to technical support organizations can be traced to an OR-SQL programmer misunderstanding one of the (many) subtle rules governing three-value logic. They should be avoided wherever possible.

② Columns that are include in a table's keys cannot contain NULL values. Columns where there is an obvious default - a type-specific token that means "Unknown" or "Did not respond," a system-generated value such as TODAY, or a generally acceptable value - should insert that value automatically. The only place NULL values are absolutely required is in columns for a FOREIGN KEY that is not mandatory.

③ Use CHECK constraints to enforce rules on entity attributes. Some rules on the data in a column will be enforced as part of the data type's implementation, but not all.

④ Name all constraints (with the possible exception of the NOT NULL constraints, which should be the de facto standard in the schema). User-defined types provide some control over the values columns can contain. However, when you have identified additional constraints as part of your analysis, it is useful to enforce them. This can lead to a large number of constraint rules per table. Naming constraints simplifies the task of managing them and determines which of them was violated by a particular operation. System-generated constraint names are totally impenetrable.

```
CREATE ROW TYPE DEPARTMENT_Structure (
  Name string NOT NULL, Number string NOT NULL, Locations string NOT NULL );
CREATE FUNCTION Read_Interface_Method
    (Argument DEPARTMENT_Structure)
RETURNS Result_Type
    RETURN Argument.name::CHAR || ' ' ||
    Argument.Number;
END FUNCTION;

CREATE TABLE DEPARTMENT
OF TYPE DEPARTMENT_Structure;

ALTER TABLE DEPARTMENT
ADD CONSTRAINT
  PRIMARY KEY ( Name )
  CONSTRAINT DEPARTMENT_PK;

CREATE TABLE EMPLOYEE (
  Name string NOT NULL, Sex string NOT NULL, Address string NOT NULL, Ssn string NOT
NULL, Bdate string NOT NULL, Salary string NOT NULL );
ALTER TABLE EMPLOYEE
ADD CONSTRAINT
  PRIMARY KEY ( Ssn )
  CONSTRAINT EMPLOYEE_PK;

CREATE TABLE DEPENDENT (
  Name string NOT NULL, Sex string NOT NULL, BirthDate string NOT NULL, Relationship
string NOT NULL   );

ALTER TABLE DEPENDENT
ADD CONSTRAINT
  PRIMARY KEY ( Name )
  CONSTRAINT DEPENDENT_PK;
CREATE TABLE PROJECT (
  NameNumber string NOT NULL, Locations string NOT NULL   );
ALTER TABLE PROJECT
ADD CONSTRAINT
  PRIMARY KEY ( Name )
  CONSTRAINT PROJECT_PK;
```

Fig. 5. Object relation database of EER schema example

7 Conclusion

The current information on the web is realized into the important property, the information is moving fast in webs, and the importance of information is increasing day after day. ER model is an information modeling tool that use most and show the relation between Entity and objects that display basic object that is required to system. Conversion methods are proposed by object relational database schema with the method changing from ER model are by an information modeling tool to XML DTD structure.

Integration design methodology that proposed in this research can be systematic, and plan efficiently XML application. The embodiment for DTD conversion is gone automatically in relational database schema using design guide given in this paper. The

more studies are necessary about the use of database techniques by the other example such as object intention or native XML finally.

References

1. Carey, M., Florescu, D., Ives, Z., Lu, Y., Shanmugasundaram, J., Shekita, E.,Subramanian, S.,: XPERANTO: Publishing Object Relational Data as XML. In Suciu, D., Vossen(eds.), G., Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18–19, 2000, 105-110.
2. Kim Chea-mi, Choi Hak-yeal, Kim Sim-seok, XML Camp with Professional , Mighty Pr Inc, 2001.
3. Carsten Kleiner, Udo W. Lipeck, Automatic Generation of XML DTDs from Conceptual Database Schemas. GI Jahrestagung(1) 2001, 396–405
4. Paul Geoffrey Brown, Object-Relational Database Development: A Plumber's Guide, Prentice Hall PTR, December 22, 2000
5. R. Elmasri, S. B. Navathe: Fundamentals of Database Systems (Third Edition), 3rd edition. World Student Series, Addison-Wesley, Reading, MA, 2000.
6. T. J. Teorey, D. Yang, J. P. Fry: A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. ACM Computing Surveys 18:2 (1986), 197–22.
7. M. Gogolla, U. Hohenstein: Towards a Semantic View of an Extended Entity-Relationship Model. ACM Transactions on Database Systems 16:3 (1991), 369–416.
8. Robert J. Muller, Database Design for Smarties: Using UML for Data Modeling, Morgan Kaufmann Publishers, Inc, 1999.

Choi Mun-Young received his BS degree from Dept. of Computer Science, Chungwoon Uni. in 1998 and obtained MS degree from Dept. of Computer Science, Chungwoon Uni. in 2000.

Dept. of Computer Science, Graduate School Soonchunhyang Uni.
E-mail : griffin@hyejeon.ac.kr

Jong-Seon Lim received his B.S. degree from Dept. of Computer Science, Korea Uni. in 1997 and obtained M.S. degree from Dept. of Computer Science, Soonchunhyang Uni. in Korea.

Dept. of Computer Science, Graduate school Soonchunhyang Uni.
E-mail : ronmer@chol.com

Joo Kyung-Soo received his BS degree from Dept. of Mathematics, Korea Uni. In 1980 and obtained MS degree from Dept. of Computer Science, Korea Uni. In 1985 and obtained Ph.D degree from Dept. of Computer Science, Korea Uni. In 1993

Dept. of Computer Science, College of Engineering Soochunhyang Uni. Prof.
E-mail : gsoojoo@sch.ac.kr

Investigating Neural Network Modeling Decisions for the Australian All-Ordinaries Index

Andrew Flitman[1], Mark Barnes[2], and Deniss Teng Tai Kiat[1]

[1]Monash University, Business Systems, Clayton, Victoria, Australia
andrew.flitman@infotech.monash.edu.au
[2]Monash University, Computing and IT, Churchill, Victoria, Australia
mbbar1@student.monash.edu.au

Abstract. Estimating stock market output depends mainly on identifying non-linear relationships of input variables. To forecast such systems a non-linear modeling tool is required. This paper describes the experimental approaches for developing an Artificial Neural Network for the purpose of modeling the Australian All Ordinaries Index movement over a prediction horizon of 1 year. Network parameters such as network architectures, input data sizing and periodicity are considered in the development of the network. The evaluation criterion for the Neural Network output is the R Square Statistic.

1 Introduction

Previously, [3] questioned the validity of Artificial Neural Networks (ANN) as a forecasting tool. He pointed out that few comparisons with standard statistical techniques had been made. However, in recent years, Neural Networks have perhaps become the most significant forecasting tools to be applied to time series data and also to model financial markets.

The interest in the ANN capabilities to forecast has resulted in a significant amount of research in comparing how well it performs compared to previous statistical techniques. Flitman [4] modeled the Australian All Ordinaries Index using a Back Propagation Neural Network and compared those results with statistical methods. He used combinations of five explanatory variables to predict the index twelve months in advance. Often the Neural Network performed as well, if not better than traditional methods. Barnes [2] also modeled the Australian All Ordinaries Index, however, in that paper the prediction forecasts were 1, 3 and 6 months. This model included a 'sliding window', which allowed the models to be retrained with the most recently available data while maintaining the training size. It was found that the explanatory variables improved the predictive power of the Neural Network when predicting 3 and 6 months into the future, but they didn't assist in predicting one-month ahead. At this stage, not all factors that influence the forecasting performance of Neural Networks are known. As a result, issues of methodology for development of any ANN for specific purposes become increasingly important. This paper extends previous work of Flitman [4] and Barnes [2] focusing on iterating data sizing, periodicity and

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 930–939, 2003.
© Springer-Verlag Berlin Heidelberg 2003

network architectures, for development of an ANN in trying to predict the movement of the Australian All Ordinaries Index. Development factors were evaluated by comparing the performance of different Neural Network architectures using the same input data set and the performance of a single Neural Network by varying and scaling the input data.

This paper is organized into four main sections. In section 2 a discussion of the All Ordinaries Index is presented. This is followed by a discussion on the selection of variables, section 3. Then a description of Neural Networks is section 4. In section 5 we describe the network architecture and its results. The final section 6 concludes the paper and discusses some further research.

2 All Ordinaries Index

The All Ordinaries Index is a weighted average of more than 250 share prices. It is an example of a price index, in that no account is taken of dividend payouts. As such it only measures the capital gain or loss resulting from investing in the stock market. In a wider context the All Ordinaries Index can be thought of as a reflection of the current and future health and wealth of the economy.

The All Ordinaries works by following the changes in the aggregate market values (AMVs) of companies covered by the index. For each company, the number of listed shares is multiplied by the market price. The AMV for the index portfolio is then calculated with the index calculated as:

$$\text{Todays Closing Index} = \frac{(\text{Yesterday's closing}) \times (\text{End - of - day AMV})}{\text{Start - of - day AMV}} \tag{1}$$

The number of companies on the index is not fixed, and alters as company eligibility for inclusion changes. It is comparable to Standard and Poors 500, The Toronto 300 Composite and the Topix Index (Japan).

Peters [9] showed that a historical plot of the S&P 500 looked very similar irrespective of whether plotted on a monthly, weekly or daily basis. This shows that non-linear dynamic (NLD) systems demonstrate self-similarity, which means that the variability of the time series would look similar when viewed with different time scales. The AOI is assumed to belong to the same family as the S&P 500 and hence exhibit non-linear dynamic and self-similarity characteristics. Theoretical work by [9] indicated that three inputs were required to predict the pattern of the S&P index. Whilst this result may imply a relatively simple time series, the reality is that no attempt was made to identify what the variables were. Thus, given the imperfection of the input variables that are actually available to us, we will need somewhat more than the theoretical minimum. As discussed below our models utilised five input variables.

3 Selection of Variables

Flitman [4], used ten different variables in combinations of five to locate the five variables that had the most influence in improving his predictions. This was guided by texts such as [7] where variables demonstrated the statistical relevance of the data. We compared the predictive power of the models using lagged AOI values and the inclusion of the extra four explanatory variables as suggested by [4]. We also looked at all combinations of the four explanatory variables using the Neural Network and Linear Regression techniques to observe the importance, individually and collectively that they had on the models. The constant variable used in all models was the historic movements in the all ordinaries.

Historic Movements in the All Ordinaries (AOI): is included since the past history of the data often can give a guide to the future.
The four explanatory variables are as follows:

Producer Prices (Manufactoring)(PP): are used in preference to Consumer Prices. Producer prices provide a leading indicator to consumer prices, and also reflect more accurately the changing prices of doing business.

Money Supply (M3): is regulated by the Government to provide sustainable economic growth. If the supply is shrinking very quickly there is a possible deflationary cycle with consequent effect on the All Ordinaries. When the rate of change is slowing down very quickly this implies a slowdown in economic activity with a consequent effect on stocks. Similarly, too rapid growth implies inflationary pressures with natural knock-on effects for stocks.

Price Earning Ratio (PE): is an excellent indicator of how cheap or expensive the stock market is. Of particular interest are real P/Es. In general the higher the market's average real P/E, the worse stocks have performed over the next twelve months [7].

Treasury 10 Year Bonds (TB): is used as an indicator of interest rates. When used independently, interest rates are only of use in predicting the stock market when expressed in real rather than nominal terms [7].

3.1 The Data Sets

Data on the above indicators were readily obtained from http://www.finance.yahoo, Australian Bureau of Statistics and www.economagic.com. Figures at the start of the month were obtained from May 1987 to May 2001. This is an extension of the data set [4] used previously and a subset of the data set [2] used. The data set consists of a total of 169 observations and divided into three categories for the Neural Network:

The data that was being used as input to develop the Neural Network model consisted of the training and test sets. Thus the network was trained using training data, with the network being saved each time an improvement was registered on the test data. Extraction of the test data from the training data was undertaken using sampling. The

remainder of the data that was not used to train the network was used for validating the output (production set). The production data was not exposed to the network prior to this. The production set was simply used to compare the actual index to the results produced by the model. It is the result against this validation set which are of particular interest. The Neural Network modeling approach can be applied to fit historic data with good R-square.

4 Neural Networks

Neural Networks are built from elements that behave somewhat like individual nerve cells (or neurons). Much research has been conducted looking at the application of this technology for classifying and forecasting for various time series such as options pricing and demand forecasts and prediction of thrift failures [10]. This paper investigates the development of a Neural Network system to predict the Australian All Ordinaries Index from the explanatory variables (input drivers).

To ensure the non-linear aspect of the problem is catered for the networks have a minimum of three levels. The reader is referred to [8] for an overview of Neural Network theory.

In designing a neural network, there are a number of variable parameters that need to be selected. These include Learning Rate, Momentum, Initial Interconnection Weights (all to do with learning), architecture selection, number of internal neurons in the hidden layer, input data manipulations and data smoothing (We shall explore some of these design parameters later in the next section). The choice of these parameters can greatly influence the network's performance.

5 Network Architectures

As indicated above, there are many parameters that have to be decided upon when developing a network model to predict the All Ordinaries Index. Parameters such as interconnection rate, learning rate, momentum, neurons interconnections, number of hidden neurons, activation functions, architectures, input data and so forth have to be considered.

It is not intended here to discuss all of the above influencing factors. The reader is referred to texts such as [5] and [11] for details. Specifically, two of the most superficial but most important parameters, architecture selection (number of hidden neurons and neurons interconnection will be considered as part of the architecture) and input data will be developed as a basis for the development of the other influencing factors.

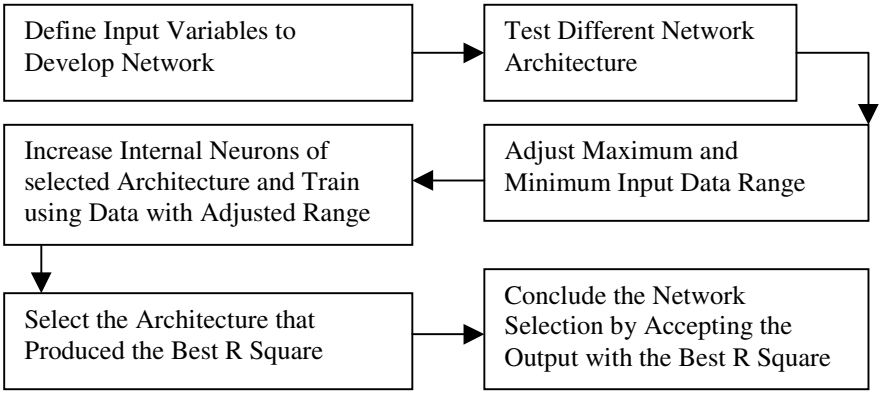
Guided by authors such as [6] and the preliminary results of Flitman [4] and Barnes [2] financial modeling applications seemed to be facilitated using tanh activation function, thus we have used that as our benchmark.
Interconnection weights, learning rate and momentum were set to defaulted values and will be kept a constant throughout the development process.

Table 1. Constant Network Parameters

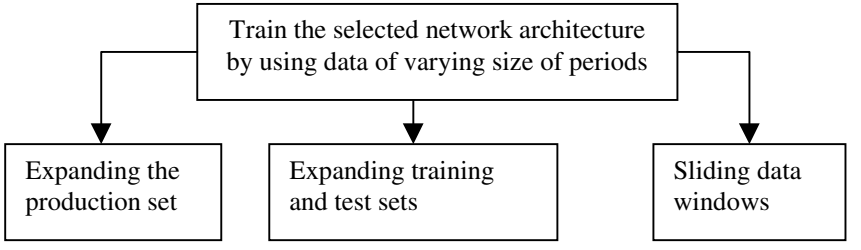
Network Parameters	Value
Interconnection Weights	0.3
Learning Rate	0.1
Momentum	0.1

The main architectures that were considered consisted of between 3 and 5 layers. Layer connection was either standard backpropagation or a modified version with each layer connected to each other (Jump Connection networks).

We conducted a number of structured experiments to investigate Network Architectures and Data Set Size. This can be summarized into a simple flow diagram.



Architecture Selection



Data Size and Period Selection

Fig. 1.

5.1 Determining Network Architectures

The data set used to train the network was kept constant in order to investigate network architectures as well as the number of internal neurons. Determining the size of the network (the number of neurons) has important consequences for its performance [11]. Too small a network may not reach an acceptable level of accuracy. Too many neurons may result in an inability for the network to generalise; it may be rote learning the patterns. The training set consisting of 5 inputs and a single output (109 observations) were applied to all the different network architectures. The data was normalised in the range of [-1,1]. This required knowledge of the expected maximums and minimums for the entire data set including the test and variation set. The scaling was to keep the inputs and outputs to figures relative to its maximum and minimum. This will minimise the effects of input magnitudes as well as aid network learning.

We decided to present the training sets to the network in a random order rather than rotate through the training set, so that bias could be minimized (the network could end up learning based on the position of the training data rather than the data itself).

Each time a number (200) of complete passes (epochs) of the training data passed through the network, we applied the partially trained network to the test data set and recorded the average forecast error. The network parameters that recorded the lowest error would be saved, overwriting previous saved parameters. This, however, lends itself to the risk of saving a network that has memorized the training set without the ability to generate new data.

This process will continue until a number of successive tests using the test data yielded no further improvement. We allowed 300 tests on the test data without any improvements before concluding the results.

Once the training was completed, the network was applied to the production set for validation. The production set was represented by 60 consecutive months of data. We relied entirely on the R square statistic when comparing the network quality.

We decided that the data normalization based on the minimum and maximum values of the data set was far too broad. We therefore decided to normalize based on standard deviations from the mean. Initially the maximum and minimum values for the data set were reset to 2 standard deviations from the mean and the network was retrained using the same architectures as above.

The results of the R square of all networks except for network Run 11, trained by setting the maximum and minimum of the input variables to 2 standard deviation of their mean, have recorded improved results. The best result was produced by the 3-layer jump connection network with 10 internal neurons with an R square of 44.38%. Further exploration found that increasing the number of neurons didn't improve the Neural Networks predictive power.

Table 2. Varying architectures by adjusting the input data (Min/Max)

	Network Architecture	Internal Neurons	R Square Default Min/Max	R Square Min/Max 2 SD
1	3 layer BPN	5	0.0871	0.2051
2	3 layer BPN	10	0.089	0.4382
3	4 layer BPN	5	0.0758	0.1984
4	4 layer BPN	10	0.0332	0.2718
5	5 layer BPN	5	0.066	0.2741
6	5 layer BPN	10	0.0012	0.0744
7	3 layer jump connection	5	0.0557	0.1076
8	3 layer jump connection	10	**0.1527**	**0.4438**
9	4 layer jump connection	5	0.0223	0.2798
10	4 layer jump connection	10	**0.1048**	**0.2864**
11	5 layer jump connection	5	**0.1726**	0.1328
12	5 layer jump connection	10	0.0156	0.1134

Results have shown that the maximum and minimum values of the input variables clearly affect the Neural Network's output. We therefore next varied the maximum and minimum values of the inputs in multiples of 2 +/- 0.25 standard deviations from the mean. The 3-layer jump connection with 10 neurons (best R Square) was then trained using the varied data:

Table 3. Maximum and Minimum of input variables

	Standard Deviations from the Mean	R Square
3 layer Jump Connection (10)	1.25	0.1794
	1.5	0.4256
	1.75	0.3950
	2	**0.4438**
	2.25	0.2313
	2.5	0.1753
	2.75	0.1788

The best R Squared still remained with the network trained using input data with their maximum and minimum values set to 2 standard deviations from the mean.

At this stage in our experiment we narrowed down the vast parameters in the development of the neural network model for predicting the All Ordinaries Index to the following (based on the above results):

- *Network Architecture*: 3-layer Jump Connection with 10 Neurons
- *Input Variables Max and Min*: 2 Standard Deviations from the Mean

5.2 Determining the Input Data Set Size and Period

The input data set were figures at the start of the month obtained from May 1997 to May 2001 ([4] used data from May 1987 to October 1993). The data set consists of a total of 169 observations. The data was divided into 3 categories for the Neural Network (training, testing and production sets).

5.2.1 ***Expanding the Production Set*****:** We trained the data set using an expanding production set. The training and test set were kept constant while the production set was being expanded from a period of 12 months (12 observations) up to 60 months in multiples of 12.

Table 4. Expanding the Production Set

	Production Set	**R Square**
1	12	0.3930
2	24	**0.6724**
3	36	0.2423
4	48	0.2484
5	60	0.4438

The best result in this exploration is produced by training the 3 layer jump connection with 10 neurons, with a training and test set of 109 observations and a production set of 24 observations. The R Square is 67.24%. The data set was from May 1987 to May 1998.

5.2.2 ***Expanding Training and Test Sets*****:** We applied the former approach to the training and test data. The production set was kept constant at 12 observations while the training set was expanded from 109 observations up till 157 observations in multiples of 12.

Table 5. Expanding Training and Test Sets

	Training Set	**Test Set**	**R Square**
1	87	22	0.3930
2	96	25	**0.5036**
3	106	27	0.2423
4	115	30	0.0290
5	125	32	0.2685

The best result was again obtained using input data from May 1987 to May 1998 with an R Square of 50.36. Therefore with a constant production set size it was found that this technique did not improve the predictive power of the Neural Network. It was noted that the results were significantly worse after the Run 2. This tends to suggest that the increased training and test set size doesn't generally improve the model.

To confirm this point a different architecture was tried. The best result again were obtained by training the network with data from May 1987 to May 1998 and the larger training and test sets did not improve its predictive power. This is shown below.

Table 6. Expanding Training and Test Sets(Cyclical)

	Training Set	Test Set	R Square
1	87	22	0.0068
2	96	25	**0.6709**
3	106	27	0.6069
4	115	30	0.0414
5	125	32	0.3743

5.2.3 ***Sliding Data Windows*****:** Finally, we applied the 'sliding window' approach to the input data. The data set was fixed to a period of 10 years (121 observations in all). All the training set, test set and production sets were constant in size. Only the time periods were varied.

Table 7. Sliding Data Windows

	Data Range	R Square
1	May 1987 – May 1997	0.3930
2	May 1988 – May 1998	**0.7569**
3	May 1989 – May 1999	0.5250
4	May 1990 – May 2000	0.0038
5	May 1991 – May 2001	0.6140

It was found the sliding window approach gave the best results of the techniques tried in this paper. The best result was found to have an R Square of 75.69% for the forward 12-month forecast of the All Ordinaries Index movement produced by training input data from May 1988 to May 1997 and validating the forecast with data from June 1997 to May 1998. As with each of the previous techniques, Run 4 gave very poor results, so it can be concluded that predicting this period (i.e. the 12 months leading up to May 2000) was very volatile and difficult to make accurate predictions. The model was developed using a 3-layer jump connection with 10 internal neurons with the input data range set to 2 standard deviations from the mean.

6 Conclusions and Further Research

From the investigations conducted, it is clear that the iterative development greatly aided the production of the 'best' Neural Network for the purpose of modeling the Australian All-Ordinaries Index.

It was found that the 3 Layer Jump Connection Neural Network with 10 internal neurons produces the best result in terms of R Square. Results also show that the input

data range of 2 standard deviations (maximum and minimum) along with the sliding window technique produced the best R Square output.

With data manipulation, it was found that different periods of the same data that was used to train the network produced significantly different results. The best R Square (trained using 3 layer feed forward neural network) for the entire experiment was obtained using data from May 1988 to May 1998.

There are of course many other factors that affect the learning ability of a neural network. With the consideration of 'Network Architectures' and 'Data Sizing and Periodicity' already considered, it would be useful to broaden this research to encompass other parameters such as momentum, learning rate and interconnection rate.

Further work is also being undertaken to improve sensitivity by looking at input data of shorter time periods (weekly or daily).

References

1. Barnes, M.B., Rimmer, R.J., Ting, K.M.. (1999) "A study of techniques for data mining from the Australian Stock Exchange" Proceedings of the Forth World Conference on Systemics, Cybernetics and Informatics, Vol. 8, Part 2, 2000, pp52–57.
2. Barnes, M.B., Flitman, A., Ting, K.M., "Australian All Ordinaries Index: Re-examine the Utilities of the Explanatory Variables Using Three Different Measures", Proceedings of the Ninth International Conference on Neural Information Processing, Vol. 5, 2002, pp 2335–2339.
3. Chatfield, C. "Neural Networks: Forecasting Breakthrough of Passing Fad?" International Journal of Forecasting, Vol. 9, 1–3.(1993).
4. Flitman, A., "A neural network forecaster of the Australian All-Ordinaries", NEORAP'95, pp 189–195.
5. Hertz, J., Krogh, A., Palmer, R.G. "Introduction to the Theory of Neural Computation", Redwood City CA, Addison-Wesley (1991)
6. Klimasauskas, C.C. "Neural Network Techniques" in Trading on the Edge, New York: John Wiley (1994)
7. Leeb, S. "Market Timing for the Nineties", New York: Harper Business (1994).
8. Lipperman, R. "An Introduction to Computing with Neural Nets", IEEE. (1987).
9. Peters, E. "Chaos and Order in the Capital Markets", New York: John Wiley (1991).
10. Salchenberger, L.M., Cinar, E.M., Lash, N.A. "Neural Networks: A New Tool for Predicting Thrift Failures", Decision Sciences Vol. 23 (1992).
11. Wasserman, P.D., "Advanced Methods in Neural Computing", Van Nostrand, New York, (1993).

Rapid Prototyping Methodology and Environments for Fuzzy Applications

Chantana Chantrapornchai

Faculty of Science, Silpakorn University, Nakorn Pathom 73000, Thailand
ctana@su.ac.th

Abstract. In this paper, we present an integrated design environment and methodology for fuzzy applications. The framework integrates the concept of rapid system design where the given system specification is broken into two components: *conceptual* and *parameter* specifications. The *conceptual specification* contains the system core, which is unchanged throughout the simulation phase and can be implemented in hardware at an early stage to create partial prototype. The *parameter specification* includes variables that are adjustable for system performance tuning. We develop a design environment to facilitate fuzzy system designers to verify the correctness of the system behavior by varying their parameters. A design example using our framework is demonstrated.

1 Introduction

In the design of application specific systems, the iterative process of detailing specifications, designing a possible solution, and testing and verifying the solution is repeatedly performed until the solution meets user requirements. However, such a repeated cycle prolongs the system development time. The system life cycle is likely to be short due to rapid pace of improvements in current technology. It is likely that the system will be out of date by the time it is ready to be shipped unless the development time is improved. The need for early prototyping, therefore, becomes critical to implement a system specification and provide customers with feedback during the design process.

Embedding all system design components into hardware is not a good approach due to lack of flexibility during reconfiguration. In fuzzy systems, by partitioning the system components properly, an early prototype can be established [1]. In this paper, our methodology and design environment integrate hardware and software capability and provides flexibility in exploring several design solutions. Given a set of user requirements, a system specification which describes what system functions are needed to satisfy the requirements is developed. Such a specification can be abstractly divided into two parts: a *conceptual specification* and a *parameter specification.* The conceptual specification contains a specification core that is rarely changed during various system adjustment while the parameter specification includes specific details which can be easily modified during the design phase. We are developing an integrated design environment which facilitate designers to design fuzzy systems based on the above concept. The framework enables designers to test many possibilities of selecting parameters. The built-in simulation feature allows designers to verify the system behavior for the tested parameters.

Considerable work has been done on developing hardware implementations for general purposed fuzzy control systems. In [2], special hardware chips have been invented to speedup the fuzzy inference engine. Other work has concentrated on design and implementation of fuzzy

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 940–949, 2003.
© Springer-Verlag Berlin Heidelberg 2003

architectures and processors [3, 10]. Software fuzzy systems are flexible but fail to provide high-speed result [7]. Although general-purposed fuzzy hardware can yield high-speed output, they may be able to neither meet stringent real-time requirement of specific applications nor specific needs of the applications. Kan and Shragowitz presented presented a generic development tool for fuzzy systems [4]. Their tools provide flexibility in defining membership functions, rules, and fuzzy logic operations. Nevertheless, they do not focus on constructing hardware modules from the derived system. In [5], a computer-aided design tool for implementation of fuzzy controllers on FPGAs was proposed. However, they do not provide flexibility during a prototyping phase,.e.g., membership functions are implemented as a boolean circuit. Also, none of these works provide a flexible framework and tools which support designing systems with many input variables and various shapes of membership functions as well as testing environments for both crisp and fuzzy inputs. Unlike the others, our research provides rapid prototyping framework for application specific fuzzy systems as well as flexible implementation.

Since one of the goals of using fuzzy logic is to reduce computational complexity of designing systems, e.g., fuzzy control systems, fuzzy logic rule bases consist of human knowledge expressed in rule format. In our approach, the conceptual specification consists of the rule base portion since it defines system functionality and is seldom changed or required only a few modification. In some classical fuzzy systems which have practical applications such as temperature controller, and inverted pendulum, the rules can commonly be found [9]. For detailed fuzzy system tuning, varying parameters are considered such as the range of system operation, the shape of membership functions, type of fuzzy logic operations, defuzzification methods, etc. We collectively include these portions in the parameter specification. Such a specification can be implemented in software for flexible tuning.

The details of our method will be discussed in the remainder of this paper. The model and synthesis methodology are discussed in Section 3. Section 4 presents an example of prototyping a fuzzy control system. Finally, Section 5 draws conclusion from this work.

2 Backgrounds

In this section, we provide some background knowledge related to fuzzy system characteristics. Fuzzy set originated by Zadeh [11] differs from the traditional crisp set. For crisp sets, the set boundary is crisp. For example, let A be the set of real numbers ranges in $[0, 10]$. We can determine whether or not a given number is a member of the set or not. We may define a membership function called $\mu_A(x)$ which maps from a real number x to a value in $[0, 1]$. Given a set $Y \subset X$, where $Y = [5, 8]$. The membership value of $y \in Y$ for $\mu_X(y)$ is 1 for all y.

For fuzzy sets, the boundary of the set is uncertain. Thus the fuzzy set is used to represent another kind of uncertainty. For example, let the universe of the age of people be $[1..20]$. Let B be a set of "young" people. Since young is a linguistic variable, the meaning of "young" varies depending on the interpretation of experts. Once people is 20 years old, will he definitely not a young people anymore? This cannot be determined precisely. Therefore, a set of young people may represent a set with fuzzy boundary. For example, for a people whose age is 25 may be considered as a young people with the degree of 25.

For simplicity, we assume that inputs are fuzzy sets and a given set of rules is in a canonical form and each rule yields only a single consequence.

Given any two inputs i and j, applying the Max-Min inference, we compute the cut of the output membership degree as shown in Figure 2(a) and Figure 2(b). After a cut is computed for each output membership function, all of them are unioned together as shown in Figure 2(c). Then, a defuzzification method is employed to change the output to crisp a value.

A general form of a rule comprising two or more linguistic variables conjunctive ($\wedge$) to each other is in the following form:

$$R_1\text{: IF } x_1 \text{ is } X_{11} \wedge \ldots \wedge x_m \text{ is } X_{1m}, \text{ THEN } y \text{ is } Y_1$$
$$R_2\text{: IF } x_1 \text{ is } X_{21} \wedge \ldots \wedge x_m \text{ is } X_{2m}, \text{ THEN } y \text{ is } Y_2$$
$$\vdots$$
$$R_1\text{: IF } x_1 \text{ is } X_{n1} \wedge \ldots \wedge x_m \text{ is } X_{nm}, \text{ THEN } y \text{ is } Y_n$$

The above system will require m non-interactive fuzzy set inputs, $\{X_1', X_2', \ldots X_m'\}$ and produce a single fuzzy set output Y'. X_{ij} is a fuzzy subset of corresponding universe of discourse U_j and Y_j is a fuzzy subset of the universe of discourse V. An element x_j denotes a fuzzy element in the related set X_{ij} and $\mu_{X_{ij}}(x_j)$ represents its corresponding membership value.

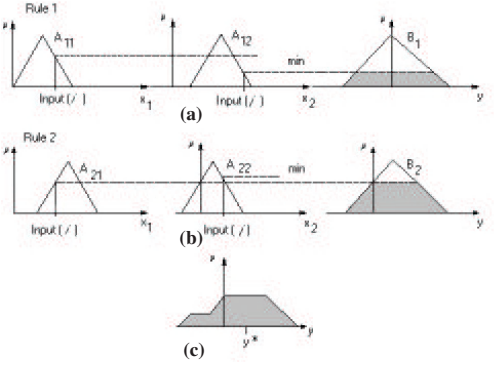


Fig. 1. Graphical Max-Min inference for two rules.

3 Design Methodology

In our methodology, we divide the specification into two parts: Conceptual and Parameter specifications.

3.1 Conceptual Specification

For the conceptual specification, it is the core component which represent the main behavior of the system. In fuzzy systems, we regard a rule base as a conceptual specification. The rule-base specification is modeled as a *Conceptual State Graph* which presents state-oriented behaviors of the rules.

Definition 1. *A Conceptual State Graph (CSG)* $G = (\mathbf{S}, \mathbf{I}, \mathbf{O}, \mathbf{E}, \omega)$ *is a connected directed edge-weighted graph where* $\mathbf{S}$ *is a set of nodes* $s_i \in S, 1 \leq i \leq p$*, representing states of the graph,* $\mathbf{I}$ *is the set of inputs,* $\mathbf{O}$ *is the set of outputs including* ϕ*,* $\mathbf{E} \subseteq S \times S$ *is a set of edges, denoted by* $s_i \rightarrow s_j$ *or* e_{s_i,s_j}*,* $s_i \in \mathbf{S}$*,* $s_j \in \mathbf{S}$*, and* ω *is a function from* $\mathbf{E}$ *to* $\mathbf{I} \times \mathbf{O}$*, representing the set of edge weights, denoted by* x/y*, where* $x \in \mathbf{I}$ *and* $y \in \mathbf{O}$*.*

For the edge weight x/y of an edge $e_{u,v}$ or $u \rightarrow v$, if $y = \phi$, then no output is given for the edge from u to v with the input x. As an example, $\mathbf{I}$ contains all combinations of values for x and y according to the rules, that is $\mathbf{I} = \{(x, L), (x, M), (x, H), (y, H)\}$. Similarly, $\mathbf{O} = \{(z, L), (z, H), \phi\}$, $\mathbf{S} = \{s_0, s_1, s_2, s_3\}$ and $\mathbf{E} = \{e_{s_0,s_1}, e_{s_0,s_2}, e_{s_0,s_3}, e_{s_1,s_3}\}$ Edges to states s_2 and s_3 have output labels (y, L) and (y, H) respectively. The output label for the intermediate edge (s_0, s_1) is ϕ. The edge weights are: $\omega(s_0, s_1) = (x, M)/\phi, \omega(s_0, s_2) = (x, L)/(z, L), \omega(s_0, s_3) = (x, H)/(z, H), \omega(s_1, s_3) = (y, H)/(z, H)$

Using this model, one can easily implement a system with multiple input variables. This can be advantageous since many current fuzzy processors allow only limited number of input variables. The CSG model can be transformed to an incompletely specified FSM and therefore, transformed to a completely specified FSM to which traditional FSM synthesis can be applied.The CSG is similar to the incompletely specified FSM, based on an assumption that no unspecified next state is encountered. All unspecified outputs, ϕ, convey "don't care" outputs and the set of final states are the set of states where outputs are given.

Based on the model, assume that the rule is in the canonical form, the number of bits used to represent output variables are $\lceil \log_2(Q + 1) \rceil$ where Q is the number of distinct output linguistic variables. The number of bits used to represent input variables is $\lceil \log_2(\sum_{j=1}^{N} M_j + 1) \rceil$ where M_j is the number of distinct input linguistic variables of input j and N is the number of inputs. Further the number of bits used to represent states equals to $\lceil \log_2(1 + Q + \sum_{j=1}^{N-1} M_j^j) \rceil$.

3.2 Parameter Specification

A parameter specification includes membership functions for input and output linguistic variables as well as defuzzification methods. For a membership function, a designer may vary the shape of membership functions and boundary. Figure 2 shows possible membership function shapes for a linguistic variable. A designer may specify different defuzzification methods such as centroid, weighted-average, Mean-Max etc. After a membership function shape and boundary are selected for each linguistic variable, a membership value is converted to discrete values ranged [0,255]. It is then mapped to a memory mapped table. The number of entries of this lookup table for each linguistic variable is P where P is the number of possible input values of each membership function that is greater than 0. Hence, totally we require the space $O(\sum_{j=1}^{N} \sum_{i=1}^{M_j} k_i + \sum_{i=1}^{Q} q_i)$ where k_i is the number of distinct values for each linguistic variable for input j and q_i is the number of distinct values for each output linguistic variable, Q is the number of output linguistic variables.

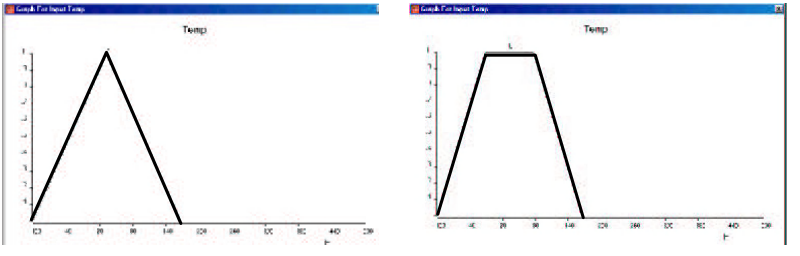


Fig. 2. Different shapes of membership functions (a) Triangular (b) Trapezoid

3.3 Component Integration

According to the fuzzy inference process, a special calculation is required to determine the output strength of each rule given the associated input strengths. We assume min operation is used for computing fuzzy rule strength. Let $\mu_{X_{ij}}(x_j)$ denote a membership function corresponding to the linguistic variable X_{ij} and the current input instance x_j for rule i. Let $\mu_{Y_i}(y)$ be a membership function corresponding to the linguistic variable Y and output instance y for rule i. Suppose a rule requires m inputs. Given input instance $x_1, \ldots, x_n$, for each rule i, we determine a modified output function $\mu'_{Y_i}(y)$ that is used in the defuzzification process, where $\mu'_{Y_i}(y) = \mu_{Y_i}(y)$ if $\mu_{Y_i}(y) < r_i$ and $\mu'_{Y_i}(y) = r_i$, otherwise. where the rule strength r_i, calculated by $r_i = \min_{j=1}^{n} \mu_{X_{ij}}(x_j)$ which is the limit of the output strength Y_i. If more than one rules give the same linguistic variable output Y_i and therefore generates a new function $\mu'_{Y_i}(y)$, each $\mu'_{Y_i}(y)$ is combined by using the max operation. $\mu''_{Y_i}(y) = \max_{\forall Y_k = Y_i} \quad '_{Y_k}(y))$ Based on these and the given CSG, the following algorithm presents the overall inference process.

Algorithm 1 (InferenceCSG)
Input: *CrispInput* $\{I_j\}$
Output: *CrispOutput*

```
1  begin
2    set 1 to all rows of Cut
3    foreach row i of A  do
4      clear Reg to 0
5      foreach column j of A do
6        read CrispInput I_j and compute Degree using InputLinguistic
7        send A[i, j] to Rule CSG and get OutIndex
8        Reg = Compute MIN(Reg,Degree) od
9      Cut[OutIndex] = MAX(Cut[OutIndex],Reg)  od
10   CrispOutput = DEFUZZIFY(Cut,OutputLinguistic);
11 end
```

Algorithm 1 is based on the max-min inference. In particular, it presents the function of the control unit in Figure 3. When the inputs to the system are fuzzy rather crisp values, we add the component to compute the minimum cut between each fuzzy input and linguistic variable before computing the min operator. Memory A temporarily stores all combinations of input sequence to be fed to the Rule CSG. The size of memory A is $b\Pi_{i=1}^{N} M_i$ bits where b is the number of bits used to represent all states and N is the number of input variables and M_i is the number of input linguistic variables for each input i.

For each row i of A, each input variable j is fed to CSG. Reg is used to keep minimum values for each rule (Line 8). This is then used to be input to MAX component to compute the maximum value of each output linguistic variable (Line 9). Note that Line 6 and Line 7 can be done at the same time, i.e, computing input degree and computing degree of output linguistic variable. At Line 10, the defuzzification method is then invoked using the stored Cut and output membership function for each output linguistic variable. The following summarizes the size needed for memory used to store each part.

InputLinguistic	OutputLinguistic	InputGenerator
$c\sum_{j=1}^{N}\sum_{i=1}^{M_j} k_i$	$c\sum_{i=1}^{Q} k_i$	$b\Pi_{i=1}^{N} M_i$

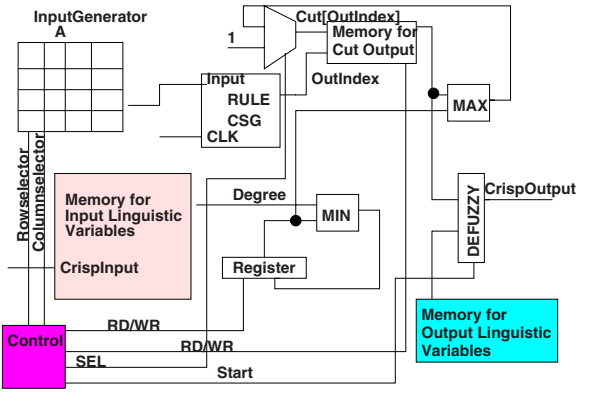


Fig. 3. Abstract connection of conceptual and parameter specification integration.

4 Design Example

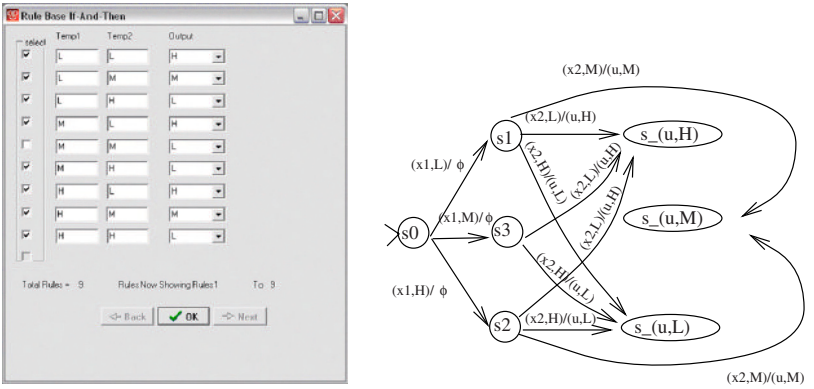
In this section, we show our design framework mapped to an example of a temperature control system for the oven found in [8]. We first create an initial design for a common rule base for the controller easily. In a meantime, we can test various parameters for the system based on our design environment.

4.1 Conceptual Specification

In the electric oven, the interior temperature is controlled by varying the heat input u to the jacket. Here two input variables are defined as the excess of temperature over the exterior, x_1, and x_2. Thus, the control system has two temperature input variables x_1 and x_2 and outputs a new temperature u so that the oven temperature is adjusted properly. Figure 4(a) uses our IDE to represent the following set of rules. In the figure, Temp1 is variable x_1 and Temp2 is variable x_2 while Output is u.

1. IF $x_1 = low\ (x_1, L)$ **and** $x_2 = low\ (x_2, L)$ THEN $u = high\ (u, H)$
2. IF $x_1 = low\ (x_1, L)$ **and** $x_2 = medium\ (x_2, M)$ THEN $u = medium\ (u, M)$
3. IF $x_1 = low\ (x_1, L)$ **and** $x_2 = high\ (x_2, H)$ THEN $u = low\ (u, L)$
4. IF $x_1 = medium\ (x_1, M)$ **and** $x_2 = low\ (x_2, L)$ THEN $u = high\ (u, H)$
5. IF $x_1 = medium\ (x_1, M)$ **and** $x_2 = high\ (x_2, H)$ THEN $u = low\ (u, L)$
6. IF $x_1 = high\ (x_1, H)$ **and** $x_2 = low\ (x_2, L)$ THEN $u = high\ (u, H)$
7. IF $x_1 = high\ (x_1, H)$ **and** $x_2 = medium$ d (x_2, M) THEN $u = medium\ (u, M)$
8. IF $x_1 = high\ (x_1, H)$ **and** $x_2 = high\ (x_2, H)$ THEN $u = low\ (u, L)$

The tuple next to each clause is a shorthand notation of the clause's linguistic variable. Figure 4(b) presents the CSG for this set of rules where the system inputs are temperatures x_1 and x_2 and the temperature output is u. An edge label is a tuple (x, z) where x is either variable x_1, x_2, or u and z is a linguistic variables with respect to $domain(x_1)$, $domain(x_2)$, or $domain(u)$ respectively.



(a) (b)

Fig. 4. (a) Rule specification based on our IDE. (b) CSG.

```
package input_output_type is
type data_3 is array (2 downto 0) of bit;
type data_2 is array (1 downto 0) of bit;
constant zero_2 : data_2 := "00";
constant zero_3 : data_3 := "000";
end input_output_type;
use work.input_output_type.all;
entity csg is
    port (input: in data_3;clk : in bit; s : out data_2; output : out data_2);
end csg;
architecture behavior of csg is
    SIGNAL state : data_2 := zero_2;
    SIGNAL statenext : data_2 := zero_2;
begin
rule: PROCESS (input,state, clk)
BEGIN
WAIT UNTIL (clk'event and clk = '1');
IF state = "00" and input = "000" THEN
    statenext <= "00";
    output <= "11";
elsif state = "00" and input = "001" THEN
    statenext <= "10";
    output <= "11";
elsif state = "00" and input = "010" THEN
    statenext <= "01";
    output <= "11";
elsif state = "01" and input = "011" THEN
    statenext <= "00";
    output <= "00";
.
.
.
else
    statenext <= "00";
    output <= "11";
end if;
state <= statenext;
END process rule;
end behavior;
```

Fig. 5. CSG VHDL implementation.

Figure 4(b) shows the CSG where S is $\{s_0, s_1, s_2, s_3, s_{(u,M)}, s_{(u,L)}, s_{(u,H)}\}$. The transitions to states $s_{(u,M)}$, $s_{(u,L)}$, and $s_{(u,H)}$ output temperature values *medium*, *low*, and *high* respectively. Figure 5 shows the VHDL representation of this CSG. In this implementation, linguistic variables (x_1, L), (x_1, M), (x_1, H), (x_2, L), (x_2, M), (x_2, H) are represented by 000, 001, 010, 011, 100 and 101 while output variables (u, H), (u, M), (u, L), ϕ are encoded as 00, 01, 10 and 11 respectively. The initial state s_0 is encoded as 0. States s_1, s_2, and s_3 are encoded as 1, 2, and 3 respectively. After optimization, states $s_{(u,H)}$, $s_{(u,M)}$ and $s_{(u,L)}$ can be merged with the

initial state to save one control step and be ready to accept the next input. Thus, we only use 2 bits to encode existing states in this design. Any sequence of inputs that does not fire any rule will lead to the initial state.

4.2 Parameter Specification

The parameter specification includes membership functions and defuzzification approaches. Since all input and output variables are temperatures, the same membership functions for all variables are used. The temperature domain is restricted to between -100 and 500 degrees centigrade using our IDE. Then we may adjust the boundary of each input and output functions as shown in Figure 6.

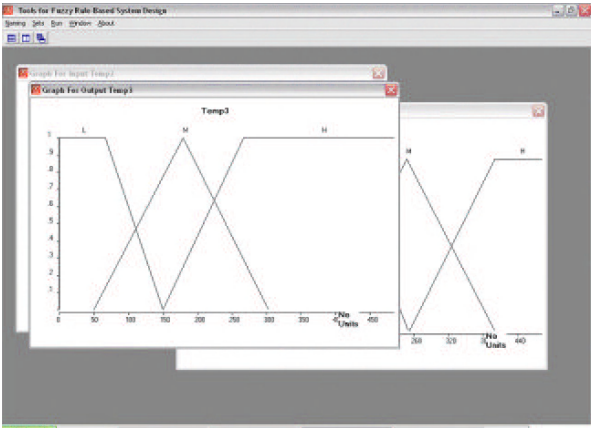


Fig. 6. Adjusting the boundary of membership functions.

For defuzzification method, we may have a choice of using centroid method, weighted average method, or Mean-Max method [8]. We may try to select each choice using our tool. In this example, we use the following equation. $x_1(t+1) = -2x_1(t)+x_2(t)+u(T), \qquad x_2(t+1) = x_1(t) - x_2(t)$ to compute the feedback function. where initial $x_1 = 90$ and $x_2 = 100$. Assume that membership function in Figure 6 is used and the centroid is used as a defuzzification method.

Using our IDE, the case when inputs are fuzzy values can also be simulated. Designers may test the system again fuzzy input cases when the shapes are varied as shown in Figure 8. Based on our framework, one can easily adjust the parameter parts to investigate design choices and re-simulate the system. Once the parameter is finalized, the prototype can be set-up rapidly.

Figure 9 shows the connection of the components and the size of each memory. For easy implementation, the membership values are scaled from real values ranging in $[0, 1]$ to integer values between $[0, 256]$. Input generator computes combinations of input linguistic variables which will be given to CSG rules (whose implementation is shown in Figure 5). The component Feedback this picture represents the feedback calculation where control action u together with inputs are used to calculate the new inputs for the next cycle. Here we use 16-bit address and data buses. The number of entries for input linguistic variables is 1806 and for output linguistic variable is 813. Input sequence generator A has 9 combinations, each of which is of three bits since we use 3 bits to represent all input linguistic variables. The size of OutIndex is 2 bits since only two bits is used to represents 4 possible output linguistic variables.

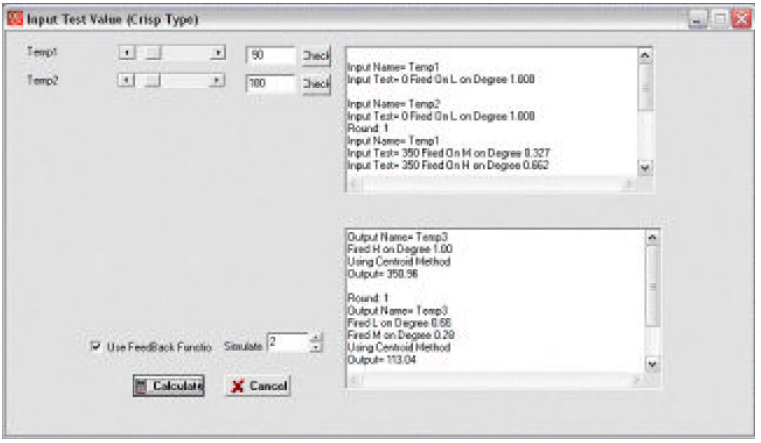


Fig. 7. Simulation for $x_1 = 90$ and $x_2 = 100$

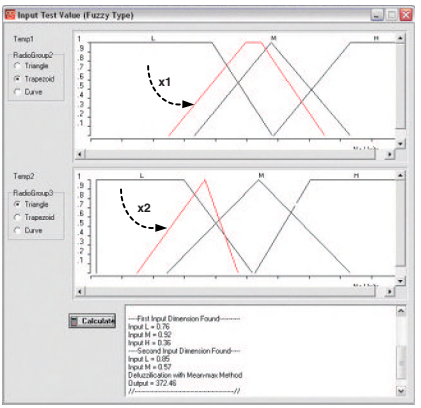


Fig. 8. Simulation for fuzzy inputs x_1 and x_2

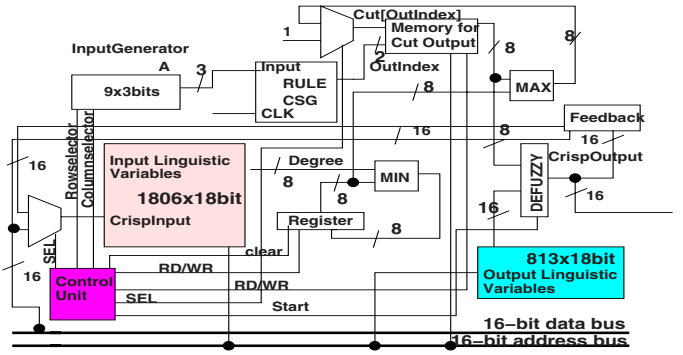


Fig. 9. Integration of all components.

5 Conclusion

We present a design methodology and environments for designing fuzzy controllers.Our methodology partitions system components into conceptual and parameterized specifications. The conceptual specification defines the core of system specification which is rarely changed and may be implemented as a hardware prototype to create a fast prototype. Parameterized components are specified in software in order to be easily edited for system modification. This method actually integrates hardware and software capability, yielding flexibility in re-specification and shorten prototyping time. We provide the IDE which gives flexibility for designers to explore fuzzy design parameters rapidly. By varying membership functions including shape and universe boundary as well as defuzzification methods, a designer performs various simulations to select the best possible parameters for the system. Altogether our framework enables rapid prototyping for fuzzy systems.

References

1. C. Chantrapornchai, S. Tongsima, and E. H. Sha. Rapid prototyping techniques for fuzzy controllers. In *Lecture Notes in Computer Science–ASIAN'99*, pages 37–49, 1999.
2. J. W. Fattaruso, S. S. Mahant-Shetti, and J. B. Barton. A fuzzy logic inference processor. In *Proc. of 3rd Intl. Conf. on Industrial Fuzzy Control and Intelligent Systems*, pages 210–214,1993.
3. A. Jaramillo-Botero and Y. Miyaka. A high-speed parallel architecture for fuzzy inference and fuzzy control of multiple processes. In *Proc. of theInternational Conference on Fuzzy Systems*, pages 1765–1770, 1994.
4. M. S. Khan and E. E. Swartzland Jr. Rapid prototyping fault-tolerant heterogeneous digital signal processing systems. In *Proc. of 6th Intl. Workshop on Rapid System Prototyping*, pages 187–193, 1995.
5. M. A. Manzoul. CAD tool for implementation of fuzzy controllers on FPGAs. *Cybernetics and Systems*, pages 599–609, 1994.
6. G. D. Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill, Inc, 1994.
7. J. Moore and M. A. Manzoul. An interactive fuzzy CAD tool. *IEEE Micro* , pages 68–74, April 1996.
8. T. J. Ross. *Fuzzy Logic with Engineering Applications*. McGrawHill, 1st edition, 1995.
9. M. H. Smith. Tuning membership functions, tuning *and* and *or* operations, tuning defuzzication: which is the best? In *Proc. of the North American Fuzzy Information Processing Society Biannual Conference*, pages 347–351, 1994.
10. H. Watanabe. Risc approach to design fuzzy processor architecture. In *Proc. of the Intl. Conf. on Fuzzy Systems*, vol. 3, pages 1809–1814, 1994.
11. L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, Part I. *Information Science*, 8:199–249, 1975.

Multiple Correlation Analysis of Fuzzy Sets

Nancy P. Lin

Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei, Taiwan, R.O.C.
nancylin@mail.tku.edu.tw

Abstract. There are so many fuzzy informations in databases waiting to be explored; methods to analyze these fuzzy data are needed. After a series of topics of correlation among fuzzy sets has been studied, namely, simple [3] and partial correlation analysis [4] of fuzzy sets, we want to study more about multiple relationships among fuzzy sets. In this paper, we will generalize the discussion of linear relationship between fuzzy attribute-pairs to linear relationship among one dependent fuzzy attribute and several independent fuzzy attributes, measured by multiple correlation coefficient of fuzzy set. We will also discuss the fact that the regression coefficients of a multiple linear regression model of one dependent fuzzy attribute on several independent fuzzy attributes can be used to obtain the multiple correlation coefficients among the particular dependent and those independent fuzzy attributes in the regression model.

1 Introduction

Data volumes get larger so rapidly, many data may be fuzzy but useful, waiting to be explored, methods to investigate these fuzzy data are needed. In previous studies [3] and [4], we have discussed the simple and partial correlation analysis on Zadeh's fuzzy sets [9,10,12,13]. In analyzing the fuzzy attributes in databases, simple correlation coefficients [3] provides us with a good sense of linear relationship of fuzzy attribute-pairs, while a partial correlation coefficient [4] shows the linear relationship between two fuzzy attributes when the influence of other fuzzy attributes are partialed out.

But, sometimes we have doubts that fuzzy attributes in a database may have contributed to the result of a specific dependent fuzzy attribute. Therefore, in this paper, the idea of correlation between fuzzy attribute-pair will be generalized to finding the linear relationship between one dependent fuzzy attribute and several independent fuzzy attributes, we will measure it by multiple correlation coefficients of fuzzy attributes.

In Section 2, a definition of multiple correlation coefficients of fuzzy attributes is given and properties of multiple correlation coefficients are noted. The same fuzzy data set from the university student activity example in [4] are used in computing the multiple correlation coefficients.

Relationship between multiple linear regression and multiple correlation coefficient are discussed in Section 3. A statistics, coefficient of multiple determinations, can be

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 950–957, 2003.
© Springer-Verlag Berlin Heidelberg 2003

obtained from the regression coefficients of a multiple regression model, which measures the combined effect of independent fuzzy attributes on the dependent fuzzy attribute in the regression model. In fact, this coefficient of multiple determinations is the square of multiple correlation coefficient, we have a study on this matter.

In the last paragraph, we will talk about the related area of further studies. Methods proposed in this paper and papers [3] and [4], may be applied to discover characteristics and the relationships among the fuzzy data in order to generate useful rules in data mining of fuzzy data.

2 Multiple Correlation Analysis of Fuzzy Sets

Here, multiple correlation coefficients are defined for considering p independent fuzzy attributes $A_1, A_2, \cdots, A_p$ to correlate one dependent fuzzy attribute O.

Suppose we have a random sample $(x_1, x_2, \cdots, x_n)$, drawn from a crisp set **X**. Membership grades, $\{\mu_O(x_i)$ and $\mu_{Aj}(x_i)$, for j = 1,…, $p\}$, of dependent fuzzy attribute O and independent fuzzy attributes $A_1, A_2, \cdots, A_p$ are attached to each observation x_i. Now we want to examine the linear relationship between dependent fuzzy attribute O and independent fuzzy attributes $A_1, A_2, \cdots, A_p$ by multiple correlation coefficient, $R_{O \bullet 1,2,\cdots,p}$, defined as follows:

Definitation 1. Multiple correlation coefficients for fuzzy sets (i). In order to find the multiple correlation between a dependent attribute O and some independent fuzzy attributes $A_1, A_2, \cdots, A_p$, the sample covariance matrix needs to be computed first as in [4], it is then partitioned into four parts $S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$ where $S_{11} = s_{OO} = s_O^2$ is the sample variance of fuzzy attribute O. S_{12} is the transpose of S_{21}, and represents the vector of covariance, s_{Oj}, of fuzzy attribute-pair (A_O, A_j), where j=1,…,p. S_{22} is a symmetric $p \times p$ matrix, represents the covariance s_{ij} of fuzzy attribute-pair (A_i, A_j), where i=1,…p and j=1,…,p.

The multiple correlation coefficient between fuzzy attribute O and fuzzy attributes $A_1, A_2, \cdots, A_p$ can then be obtained by

$$R_{O \bullet 1,\cdots,p} = \sqrt{\frac{S_{12} S_{22}^{-1} S_{21}}{S_{11}}} \tag{2.1}$$

Let us illustrate this result in a simple case when p=2. Assume that we have three fuzzy attributes O, A_1 and A_2, we are interested in finding the multiple correlation

coefficient of dependent fuzzy attribute $\boldsymbol{O}$ on independent fuzzy attributes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. The first thing is to compute the sample covariance matrix

$$\boldsymbol{S} = \begin{bmatrix} s_{OO} & s_{1O} & s_{2O} \\ s_{O1} & s_{11} & s_{12} \\ s_{O2} & s_{21} & s_{22} \end{bmatrix},$$

which can be partitioned into four parts, that is:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} \\ \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} s_{OO} \end{bmatrix} & \begin{bmatrix} s_{1O} & s_{2O} \end{bmatrix} \\ \begin{bmatrix} s_{O1} \\ s_{O2} \end{bmatrix} & \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \end{bmatrix} \tag{2.2}$$

Where $\boldsymbol{S}_{11}$ is the sample variance of fuzzy attribute $\boldsymbol{O}$. $\boldsymbol{S}_{12}$, a 2×1 vector and the transpose of $\boldsymbol{S}_{21}$, represents the covariance-vector of fuzzy attribute $\boldsymbol{O}$ and fuzzy attributes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. $\boldsymbol{S}_{22}$ is the symmetric covariance-matrix of fuzzy attributes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. Then by using equation (2.1), multiple correlation coefficient of fuzzy attribute $\boldsymbol{O}$ on fuzzy attributes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ can be computed

$$\mathrm{R}_{O\bullet 12} = \left[\frac{\begin{bmatrix} s_{1O} & s_{2O} \end{bmatrix} \times \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}^{-1} \times \begin{bmatrix} s_{O1} \\ s_{O2} \end{bmatrix}}{s_{OO}} \right]^{1/2}$$

$$= \left[\frac{\begin{bmatrix} s_{1O} & s_{2O} \end{bmatrix} \times \dfrac{1}{(s_{11}s_{22} - s_{12}s_{21})} \begin{bmatrix} s_{22} & -s_{12} \\ -s_{21} & s_{11} \end{bmatrix} \times \begin{bmatrix} s_{O1} \\ s_{O2} \end{bmatrix}}{s_{OO}} \right]^{1/2}$$

$$= \left[\frac{\begin{bmatrix} s_{1O}s_{22} - s_{2O}s_{21} & -s_{1O}s_{12} + s_{2O}s_{11} \end{bmatrix} \times \begin{bmatrix} s_{O1} \\ s_{O2} \end{bmatrix}}{(s_{11}s_{22} - s_{12}s_{21}) \times s_{OO}} \right]^{1/2}$$

$$= \left[\frac{(s_{22}s_{O1} - s_{O2}s_{12})s_{O1} + (s_{11}s_{O2} - s_{O1}s_{12})s_{O2}}{(s_{11}s_{22} - s_{12}^2) \times s_{OO}} \right]^{1/2}$$

$$= \left[\frac{(s_{22}s_{O1} - s_{O2}s_{12})}{(s_{11}s_{22} - s_{12}^2)} \cdot \frac{s_{O1}}{s_{OO}} + \frac{(s_{11}s_{O2} - s_{O1}s_{12})}{(s_{11}s_{22} - s_{12}^2)} \cdot \frac{s_{O2}}{s_{OO}} \right]^{1/2}.$$

Property 1. Properties of Multiple correlation coefficients. In the case of p=2, for any two independent fuzzy attributes A_1 and A_2, the properties that should be satisfied by multiple correlation coefficient $R_{O\bullet 1,2}$ are:

(i) $R_{O\bullet 1,2} = R_{O\bullet 2,1}$.

(ii) $R_{O\bullet 1,2}$ reaches its minimum when $r_{12} = 0$, where r_{12} is the simple correlation coefficient between fuzzy attributes A_1 and A_2.

$$\text{Since } R_{O\bullet 1,2} = \left[\frac{\begin{bmatrix} s_{10} & s_{20} \end{bmatrix} \times \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}^{-1} \times \begin{bmatrix} s_{01} \\ s_{02} \end{bmatrix}}{s_{00}} \right]^{1/2}$$

$$= \left[\frac{(s_{22}s_{01} - s_{02}s_{12})s_{01} + (s_{11}s_{02} - s_{01}s_{12})s_{02}}{(s_{11}s_{22} - s_{12}^2) \times s_{00}} \right]^{1/2}$$

Divide both the numerator and the denominator by $s_{00}s_{11}s_{22}$, we have

$$R_{O\bullet 1,2} = \left[\frac{\frac{s_{01}^2}{s_{00}s_{11}} - \frac{s_{01}s_{02}s_{12}}{s_{00}s_{11}s_{22}} + \frac{s_{02}^2}{s_{00}s_{22}} - \frac{s_{01}s_{02}s_{12}}{s_{00}s_{11}s_{22}}}{1 - \frac{s_{12}^2}{s_{11}s_{22}}} \right]^{1/2}$$

$$= \left[\frac{r_{01}^2 - 2\frac{s_{01}s_{02}s_{12}}{s_{00}s_{11}s_{22}} + r_{02}^2}{1 - r_{12}^2} \right]^{1/2}$$

(iii) $R_{O\bullet 1,2}$ is not defined, if $r_{12} = 1$, we can see that from the last equation.

(iv) $0 \le \mathrm{R}_{O\bullet I,2} \le 1$. Since $\mathrm{R}^2_{O\bullet I,2}$ is a ratio of partial variance to total variance, which is positive and always ≤ 1, therefore, the positive square root of it is $0 \le \mathrm{R}_{O\bullet A,C} \le 1$.

An example, in previous discussion of partial correlation analysis [4], we have used a university student activity sample and noticed that fuzzy attributes ***A***, ***C***, and ***E*** may have contributed to the result of fuzzy attribute ***O***. We refer the fuzzy attribute ***O***, the objective accomplishment, as a dependent attribute, and fuzzy attributes ***A***, ***C***, and ***E*** as independent attributes, representing group atmosphere, group leader, and group environment respectively. The result of multiple correlation analysis ought to show how the independent fuzzy attributes ***A***, ***C***, and ***E*** may be involved to what extent in the result of the dependent fuzzy attribute ***O*** in this fuzzy data set.

Multiple correlation coefficients between dependent fuzzy attribute ***O*** and pairs of combinations of independent fuzzy attributes ***A***, ***C***, and ***E*** can be computed from equation (2.1): $\mathrm{R}_{O\bullet AC} = 0.4998$, $\mathrm{R}_{O\bullet AE} = 0.4850$, $\mathrm{R}_{O\bullet CE} = 0.2144$.

Similarly, multiple correlation coefficient of the dependent fuzzy attribute ***O*** regarding to three independent fuzzy attributes ***A***, ***C***, and ***E***, can be computed:

$\mathrm{R}_{O\bullet ACE} = 0.5024$.

3 Multiple Linear Regression vs. Multiple Correlation Coefficient of Fuzzy Sets

In a database including an interesting dependent fuzzy attribute ***O*** and some related independent fuzzy attributes A_1 ,A_2 and A_3, attached with their membership grades, { $\mu_O(x_i)$ and $\mu_{A_1}(x_i)$, $\mu_{A_2}(x_i)$, $\mu_{A_3}(x_i)$ }, now, let us consider a simple fuzzy linear regression line of fuzzy attribute ***O*** on fuzzy attribute A_1, $\hat{\mu}_O = b_0 + b_1\mu_{A1}$, (μ_O and μ_{A1} are as defined in previous section). The effect of independent fuzzy attribute A_1 on the variance of ***O***, can be measured by a coefficient of simple determination r^2_{O1}. This means that independent fuzzy attribute A_1 accounts for 100(r^2_{O1})% of the total variance of dependent fuzzy attribute ***O***. The coefficient of simple determination can be found from the regression coefficient,

$$\mathrm{r}^2_{O1} = \left[\frac{s_{O1}}{s_O s_1}\right]^2 = \frac{s_{O1}}{s_{11}} \cdot \frac{s_{O1}}{s_{OO}} = b_1 \cdot \frac{s_{O1}}{s_{OO}} = \mathrm{R}^2_{O\bullet 1} \quad \because b_1 = \frac{s_{O1}}{s_{11}}.$$

Here, s_{OO}, s_{O1}, s_{11} are the same as in equation (2.2).

Where $\mathrm{R}_{O\bullet 1}$ is the multiple correlation coefficient, as from previous section, of dependent fuzzy attribute ***O*** with only one independent fuzzy attribute A_1. And in this

special case of only one independent fuzzy attribute, $R_{O\bullet 1}$ is actually the simple correlation coefficient [3], r_{O1}, between O and A_1.

Next, let us examine the combined effect of independent fuzzy attributes A_1 and A_2 on the dependent fuzzy attribute O in the fuzzy linear regression model, $\hat{\mu}_O = b_0 + b_1\mu_{A1} + b_2\mu_{A2}$. This effect can be measured by the coefficient of multiple determination, $R^2_{O\bullet 12}$, that means that $100(R^2_{O\bullet 12})\%$ of the total variance of dependent fuzzy attribute O is accounted for by independent fuzzy attributes A_1 and A_2. Where $R_{O\bullet 12}$ is the multiple correlation coefficient of A_1 and A_2 on O.

The coefficient of multiple determinations can also be obtained from the regression coefficients of multiple regression line by

$$R^2_{O\bullet 12} = b_1 \cdot \frac{s_{O1}}{s_{OO}} + b_2 \cdot \frac{s_{O2}}{s_{OO}}.$$

Here, s_{OO}, s_{O1}, s_{O2} are the same as in equation (2.2).

Now, a third independent fuzzy attribute A_3 is introduced into the regression model, $\hat{\mu}_O = b_0 + b_1\mu_{A1} + b_2\mu_{A2} + b_3\mu_{A3}$, the effect of combining all three independent fuzzy attributes, A_1 ,A_2 and A_3 on the dependent fuzzy attribute O, is measured by the coefficient of multiple determination, $R^2_{O\bullet 123}$, from which we know that $100(R^2_{O\bullet 123})\%$ of the total variance in dependent fuzzy attribute O is accounted for by attributes A_1 ,A_2 and A_3.

Also, the coefficient of multiple determination can be computed from the regression coefficients as:

$$R^2_{O\bullet 123} = b_1 \cdot \frac{s_{O1}}{s_{OO}} + b_2 \cdot \frac{s_{O2}}{s_{OO}} + b_3 \cdot \frac{s_{O3}}{s_{OO}}.$$

Here, $s_{OO}, s_{O1}, s_{O2}, s_{O3}$ are the same as in equation (2.2), and $R_{O\bullet 123}$ is the multiple correlation coefficient of A_1 ,A_2 and A_3 on O. From the discussion of this section, we have another way of determine multiple correlation coefficients.

Definitation 2. Multiple correlation coefficients for fuzzy sets (ii). If the multiple linear regression equation of independent fuzzy attributes $A_1, A_2, \cdots, A_p$ on the dependent fuzzy attribute O is known,

$$\hat{\mu}_O = b_0 + b_1\mu_{A1} + b_2\mu_{A2} + \cdots + b_p\mu_{Ap}. \quad (3.1)$$

Then the multiple correlation coefficient between fuzzy attribute O and independent fuzzy attributes $A_1, A_2, \cdots, A_p$ can be found by the formula:

$$R_{O\bullet 1,2,\cdots,p} = \sqrt{\frac{\sum_{i=1}^{p} b_i s_{Oi}}{s_{OO}}}. \tag{3.2}$$

Where s_{OO} and $s_{Oi}, i = 1, \cdots, p$ are the same as in equation (2.2).

An example, from the result of the university student activity sample [3,4], we obtain the simple linear regression line of dependent fuzzy attribute O on independent fuzzy attributes A, $\hat{\mu}_O = 0.3542 + 0.5771\,\mu_A$. Then, coefficient of simple determination can be obtained by equation (3.2):

$$r_{OA}^2 = 0.2352 = (0.4850)^2 = R_{O\bullet A}^2.$$

Which means, 23.52% of the total variation of dependent fuzzy attribute O (degree of the objective accomplishment), is explained by fuzzy attribute A (group atmosphere). Or, the simple correlation coefficient between fuzzy attribute O and fuzzy attribute A is 0.4850, as that in [3].

When two independent fuzzy attributes A, C are introduced into the regression model, we have $\hat{\mu}_O = 0.1514 + 0.5500\mu_A + 0.2420\mu_C$. The coefficient of multiple determination can be obtained from equation (3.1):

$$\begin{aligned} R_{O\bullet AC}^2 &= b_1 \cdot \frac{s_{OA}}{s_{OO}} + b_2 \cdot \frac{s_{OC}}{s_{OO}} \\ &= 0.5500 \cdot \frac{0.005}{0.012} + 0.2420 \cdot \frac{0.001}{0.012} \\ &= 0.2498 = (0.4998)^2 \end{aligned}$$

That means, 24.98% of the total variance of dependent fuzzy attribute O is explained by the combined effect of A (group atmosphere) and C (characteristics of group leader). It also leads to that the multiple correlation coefficient of A and C on O is 0.4998, which corresponds to the result in previous paragraph.

Now, independent fuzzy attribute E (group environment) is added to the regression model, we have $\hat{\mu}_O = 0.1415 + 0.5638\mu_A + 0.2918\mu_C - 0.0579\mu_E$. Then the coefficient of multiple determination computed from equation (3.1)

$$\begin{aligned} R_{O\bullet ACE}^2 &= b_1 \cdot \frac{s_{OA}}{s_{OO}} + b_2 \cdot \frac{s_{OC}}{s_{OO}} + b_3 \cdot \frac{s_{OE}}{s_{OO}} \\ &= 0.5638 \cdot \frac{0.005}{0.012} + 0.2918 \cdot \frac{0.001}{0.012} - 0.0579 \cdot \frac{0.002}{0.012} \end{aligned}$$

$$= 0.2524$$

$$= (0.5024)^2.$$

This figure shows that 25.24% of the total variance of dependent fuzzy attribute ***O*** is explained by the combined effect of all three independent fuzzy attributes ***A***, ***C*** and ***E***. Also, we can see that the multiple correlation coefficient of ***A***, ***C*** and ***E*** on dependent fuzzy attribute ***O*** is 0.5024 as obtained from equation (2.1) in previous section.

4 Conclusions

Correlation analysis among attributes in a database usually interests us; the correlation analysis for ordinary crisp data has been discussed for a long time in conventional statistics and related applications [1,2,5]. In the mean time, many fuzzy attributes existed in databases are getting more and more interested to the researchers; our objective is to extend the correlation analysis to fuzzy attributes in analyzing fuzzy data. By adopting the methods from mathematical statistics, definitions has been proposed for simple correlation coefficient on fuzzy sets [3], partial correlation coefficient on fuzzy sets [4], and, in this paper, multiple correlation coefficient on fuzzy sets.

As data volumes grow exponentially, massive fuzzy information in databases have not yet been discovered, which may represent important facts or past experiences. To mine knowledge and generate useful rules from fuzzy data is our goal to pursue. Methods proposed in this paper and papers [3] and [4], can be applied to discover characteristics and the relationships among the fuzzy data in order to generate useful rules in data mining of fuzzy data.

References

1. T. W. Anderson: An Introduction to Multivariate Statistical Analysis, 2nd Ed. John Wiley & Sons, 1984.
2. S. F. Arnold: Mathematical Statistics, Prentice-Hall, New Jersey, 1990.
3. D. A. Chiang and N. P. Lin: Correlation of Fuzzy Sets, Fuzzy Sets and Systems, Vol. 102 (1999) 221–226.
4. D. A. Chiang and N. P. Lin: Partial Correlation of Fuzzy Sets, Fuzzy Sets and Systems, Vol. 110 (2000) 209–215.
5. S. Dowdy and S. Wearden: Statistics for Research, John Wiley & Sons, 1983.
6. G. J. Klir and T. A. Folger: Fuzzy Sets, Uncertainty, and Information, Prentice-Hall International, Inc. 1988.
7. G. J. Klir and B. Yuan: Fuzzy Sets and Fuzzy Logic, Theory and Applications, Prentice-Hall International, Inc. 1995.
8. L. A. Zadeh: Fuzzy sets, Information and Control, Vol. 8 (1965) 338–353.
9. H.-J. Zimmermann: Fuzzy Set Theory and Its Applications, 2nd Ed., Kluwer Academic Publishers, 1991.

An Efficient Dispersal and Encryption Scheme for Secure Distributed Information Storage*

Sung Jin Choi [1], Hee Yong Youn[1**], and Bo Kyoung Lee[2]

[1] School of Information and Communications Engineering
Sungkyunkwan University
440-746, Suwon, Korea +82-31-290-7949
{choisj, youn}@ece.skku.ac.kr
[2] Information Security Technology Division
Korea Information Security Agency
138-803, Seoul, Korea +82-2-405-5263
bklee@kisa.or.kr

Abstract. Digital information is a critical resource, creating a need for distributed storage systems that provide sufficient data availability and security in the face of failures and malicious attacks. However, the requirements for achieving high availability and security with distributed storage systems usually conflict with each other. This paper proposes a novel data dispersal/encryption scheme improving both the availability and security of distributed storage system by using some important properties of matrices. It also allows complete recovery of original data even though they are partially damaged. Analysis shows it improves the availability about 15% compared with an efficient existing scheme for both read and write operation, while it allows secure storage simultaneously.

1 Introduction

Digital information has become a critical resource, creating a need for distributed storage systems that provide sufficient data availability and data security in the face of failures and malicious attacks. Many research groups have been exploring the design and implementation of such survivable storage systems. These systems are built on various decentralized storage system technologies sharing a same high level architecture. The challenge in the development of survivable storage is to achieve acceptable levels of performance and manageability. Moreover, a means for evaluating survivable storage systems is required to facilitate the design of the systems [1].

A data distribution scheme needs data encoding and partitioning algorithm. There exist many such algorithms applicable to survivable storage, including encryption, replication, striping, erasure-resilient coding, secret sharing, and various combinations of them. They offer different levels of performance (throughput), availability (prob-

* This work was supported by Korea Research Foundation Grant (KRF-2002-041-D00421)
** Corresponding author

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 958–967, 2003.
© Springer-Verlag Berlin Heidelberg 2003

ability that data can be accessed), and security (effort required to compromise the confidentiality or integrity of stored data).

For example, replication provides high availability at the sacrifice of network bandwidth and storage space, whereas short secret sharing provides high availability and security with low storage and network bandwidth overhead but at high computation overhead. Likewise, selecting the number of shares required to reconstruct a secret-shared value involves a trade-off between availability and confidentiality: if more machines must be compromised to steal the secret, then more machines must be operational to provide it legitimately. Availability of storage system needs to be improved through data distribution, and development of new efficient data distribution and protection scheme is inevitable.

In distributed storage system availability and security of data are greatly influenced by the policy how the data are dispersed. For example, complete recovery is possible even though most data are lost in one scheme while one node failure causes loss of entire data in another scheme, vice versa. Usually, the requirements for achieving high availability and security with distributed storage system conflict with each other, and there exist some tradeoff between them. This paper proposes a novel data dispersal/encryption scheme to improve both the availability and security of distributed storage system by using some important properties of matrices. It also allows complete recovery of original data even though they are partially damaged. Analysis shows that the availability is improved about 15% compared with an efficient existing scheme, while it allows secure storage.

The rest of the paper is organized as follows. The next section explains representative data distribution schemes. Section 3 presents the proposed dispersal/encryption scheme using eigenvalues and eigenvectors of matrices. In Section 4 we evaluate and compare the performance of the proposed scheme. Finally, Section 5 concludes the paper with future work.

2 Related Work

Secure distributed information storage system uses general threshold schemes to encode data before it is stored. Specifically, a *p-m-n* threshold scheme breaks data into n shares such that any m of the shares can reconstruct the original data and fewer than p shares reveal absolutely no information about the original data. Although encryption makes it difficult to ascertain the original data, it does not change the value of p since information is still available for theft. Different parameter selections of *p-m-n* expose a large space of encoding mechanisms for storage [2].

An example of a specific threshold scheme is N-way replication. It is a *1-1-N* threshold scheme. That is, each replica reveals some information on the encoded data (hence $p = 1$). A single replica is required to reconstruct the original data ($m = 1$), and there are N replicas to select from when attempting to reconstruct the original data ($n = N$). Table 1 lists some specific examples of general threshold schemes.

The threshold schemes can be implemented in different manners. Blakley's secret sharing scheme works in an m-dimensional space. Secrets (data) are points in the

Table 1. Specific threshold schemes.

Parameters	Description
1-1-n	Replication
1-n-n	Decimation (Striping)
n-n-n	Splitting (XORing)
1-m-n	Information Dispersal
m-m-n	Secret Sharing
p-m-n	Ramp Scheme

space, and the shares are multidimensional planes. Fewer than *m* shares represent a multidimensional plane that contains the secret. However, since all points in the field being considered are part of the plane, no information is revealed. With *m* shares, a single point of intersection - the secret - is determined. Figure 1 illustrates the Blakley's secret sharing scheme. Shamir's secret-sharing scheme is based on interpolating the coefficients of a polynomial by evaluating the polynomial at certain points [3].

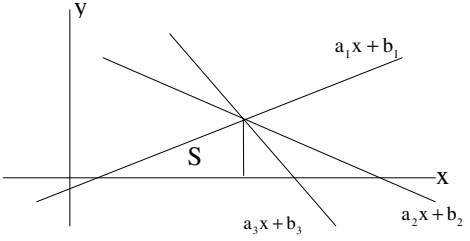


Fig. 1. Blakley's secret sharing scheme with m=2, n=3, and original data S.

All implementations of secret sharing use random numbers to guarantee that collecting fewer than *p* shares reveals no information on the original data. Rabin's information dispersal algorithm, a *1-m-n* threshold scheme, does not use random numbers. Indeed, the parameter *p* indicates the number of random numbers required per encoding (i.e. *p*-1random numbers are required to encode a secret for any threshold scheme). Ramp schemes implement the full range of general *p-m-n* threshold schemes, while they are like secret-sharing schemes up to *p* shares and information dispersal schemes with *p* or more shares [4-5].

Figure 2 shows some examples of threshold schemes. (a) Replication (*1-1-n*) increases availability by increasing the storage required by a factor of *n*. It provides no confidentiality because each share contains an entire copy of the original information object. (b) Splitting (*n-n-n*) increases confidentiality by increasing the storage required by a factor of *n* but decreases availability because all *n* shares must be available. (c) Decimation (*1-n-n*) divides information objects into *n* pieces and stores each piece separately. It decreases availability because all shares must be available while it offers no confidentiality because each share exposes 1/*n* of the original information. (d) Rabin's information dispersal algorithm (*1-m-n*) offers a range of storage solutions that show trade-off between availability and required storage space. Like decimation, it does not offer confidentiality [6].

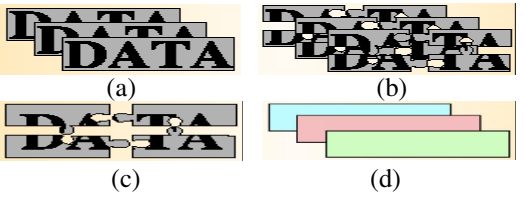


Fig. 2. Examples of simple threshold schemes.

The threshold schemes can be used instead of cryptographic techniques to guarantee the confidentiality, and multiple schemes can be combined. For example, short secret sharing encrypts the original information with a random key, stores the encryption key using secret sharing, and stores the encrypted information using secret sharing or information dispersal. Short secret sharing offers a different set of trade-offs between confidentiality and storage requirements than general threshold schemes. The confidentiality of the data stored by short secret sharing hinges on the difficulty of analyzing the information gained by collecting the shares because the information gained pertains to the encrypted data.

An extension to threshold schemes is cheater detection. Here the shares are constructed in such a fashion that a client reconstructing the original information object can tell, with high probability, whether some shares have been modified or not. This technique allows strong information-integrity guarantees. Cheater detection can also be implemented using cryptographic techniques such as adding digests to data before encoding or to the shares themselves after the data have been encoded [7].

Secret-sharing scheme is *m-m-n* threshold scheme that shows trade-off between confidentiality and availability; the higher the confidentiality guarantee, the more shares required to reconstruct the original information object. It can be thought of as a combination of splitting and replication techniques.

3 The Proposed Scheme

We first present some definitions and theorems required to explain the proposed dispersal/encryption scheme.

3.1 Eigenvalues and Eigenvector's Definition

• Definition 1.
Let A be an $n \times n$ matrix. A nonzero vector v is an eigenvector of A if Eq. (1) holds for some scalar λ. λ is called an eigenvalue of A corresponding to the eigenvector v. Eigenvalues are also known as characteristic, or proper, values or even as latent roots.

$$Av = \lambda v \,. \tag{1}$$

Let us now discuss how to compute eigenvalues and eigenvector in general. Because

$$\begin{aligned} Av=\lambda v &\Rightarrow Av=\lambda Iv \\ &\Rightarrow Av-\lambda Iv=0 \\ &\Rightarrow (A-\lambda I)v=0 \end{aligned} \quad . \tag{2}$$

We see that v is an eigenvector if and only if it is a nontrivial solution of the homogeneous system $(A-\lambda I)v=0$. In this case v is a nonzero vector of the null space of $A-\lambda I$. The system has a nontrivial solution if and only if the determinant of the coefficient matrix is zero. Thus, λ is an eigenvalues of A if and only if $\det(A-\lambda I)=0$. The following is a well known theorem [8].

• Theorem 1.
Let A be a square matrix.
1. A scalar λ is an eigenvalue of A if and only if det(A- λ I) = 0.
2. A vector v is an eigenvector of A corresponding to an eigenvalue λ if and only if v is a nontrivial solution of the system (A- λ I)v = 0.

3.2 Dispersal/Encryption Scheme

We need a general method for finding eigenvalue and eigenvector by using Definition 1 and Theorem 1, and it is the Power Method. It computes the dominant eigenvalue and an eigenvector corresponding to the dominant eigenvalue. Without loss of generality, it is necessary to assume that A has the following two properties:

i. There is a single eigenvalue of maximum modulus.
ii. There is a linearly independent set of n eigenvectors.

According to the first assumption, the eigenvalues can be labeled such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| . \tag{3}$$

According to the second assumption, there is a basis $\{u^{(1)}, u^{(2)}, ..., u^{(n)}\}$ for C^n such that

$$Au^{(j)} = \lambda_j u^{(j)} \ (1 \leq j \leq n) . \tag{4}$$

Let $x^{(0)}$ be an element of C^n such that when $x^{(0)}$ is expressed as a linear combination of the basis elements $u^{(1)}, u^{(2)}, ..., u^{(n)}$, the coefficient of $u^{(1)}$ is not 0. Thus,

$$x^{(0)} = a_1 u^{(1)} + a_2 u^{(2)} + \cdots + a_n u^{(n)} \ (a_1 \neq 0) . \tag{5}$$

We form then $x^{(1)} = Ax^{(0)}$ $x^{(2)} = Ax^{(1)}$ $\cdots$ $x^{(k)} = Ax^{(k-1)}$ to have

$$x^{(k)} = A^k x^{(0)} . \tag{6}$$

In the following analysis there is no loss of generality in absorbing all the coefficients a_j in the vectors $u^{(j)}$ that they multiply. Hence, we may rewrite Eq. (5) as

$$x^{(0)} = u^{(1)} + u^{(2)} + \cdots + u^{(n)}. \tag{7}$$

By this equation and (6), we have

$$x^{(k)} = A^k u^{(1)} + A^k u^{(2)} + \cdots + A^k u^{(n)}. \tag{8}$$

Using Eq. (4), we arrive at

$$x^{(k)} = \lambda_1^k u^{(1)} + \lambda_2^k u^{(2)} + \cdots \lambda_n^k u^{(n)}. \tag{9}$$

$$x^{(k)} = \lambda_1^k \left[u^{(1)} + \left(\frac{\lambda_2}{\lambda_1} \right)^k u^{(2)} + \cdots + \left(\frac{\lambda_n}{\lambda_1} \right)^k u^{(n)} \right]. \tag{10}$$

Since $|\lambda_1| > |\lambda_j|$ for $2 \le j \le n$, we see that the coefficients $\left(\frac{\lambda_j}{\lambda_1} \right)^k$ tend to 0 and the vector within the brackets converges to $u^{(1)}$ as $k \to \infty$.

To simplify the notation, we write $x^{(k)}$ in the form

$$x^{(k)} = \lambda_1^k \left[u^{(1)} + \varepsilon^{(k)} \right]. \tag{11}$$

where $\varepsilon^{(k)} \to 0$ as $k \to \infty$. In order to be able to take ratios, let φ be any linear functional on C^n for which $\varphi(u^{(1)}) \neq 0$. Recall that a linear functional φ satisfies $\varphi(\alpha x + \beta y) = \alpha\varphi(x) + \beta\varphi(y)$, for scalars α and β and vectors x and y. (For example, φ could simply evaluate the jth component of any given vector.) Then

$$\varphi(x^{(k)}) = \lambda_1^k \left[\varphi(u^{(1)}) + \varphi(\varepsilon^{(k)}) \right]. \tag{12}$$

Consequently, the following ratios converges to λ_1 as $k \to \infty$:

$$r_k \equiv \frac{\varphi(x^{(k+1)})}{\varphi(x^{(k)})} = \lambda_1 \left[\frac{\varphi(u^{(1)}) + \varphi(\varepsilon^{(k+1)})}{\varphi(u^{(1)}) + \varphi(\varepsilon^{(k)})} \right] \to \lambda_1. \tag{13}$$

This constitutes the Power Method for computing λ_1. Since the direction of the vector $x^{(k)}$ aligns more and more with $u^{(1)}$ as $k \to \infty$, the method can also give us the eigenvector $u^{(1)}$ [9]. If eigenvectors found are

$$u^{(1)} = \begin{bmatrix} s_1 \\ \vdots \\ \vdots \\ s_n \end{bmatrix} s_n \in R,\ u^{(2)} = \begin{bmatrix} t_1 \\ \vdots \\ \vdots \\ t_n \end{bmatrix} t_n \in R, \ldots\ldots,\ u^{(n)} = \begin{bmatrix} z_1 \\ \vdots \\ \vdots \\ z_n \end{bmatrix} z_n \in R. \tag{14}$$

Here P ($P = \left[u^{(1)} u^{(2)} \cdots u^{(n)} \right]$) is actually stored data and matrix D consisting of eigenvalues becomes Decryption Key. For example, if original data is $A=\begin{bmatrix} 3 & 0 & 0 \\ -4 & 6 & 2 \\ 16 & -15 & -5 \end{bmatrix}$, $\lambda_1 = 0,\ \lambda_2 = 1,\ \lambda_3 = 3$. Eigenvectors that correspond to each λ become $\begin{bmatrix} 0 \\ s \\ -3s \end{bmatrix}, \begin{bmatrix} 0 \\ 2t \\ -5t \end{bmatrix}, \begin{bmatrix} u \\ 0 \\ 2u \end{bmatrix}$ s,t,u $\in$ R. Therefore, if $s = t = u = 1$, the actually stored data P and decryption key D are as follows.

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 2 & 0 \\ -3 & -5 & 2 \end{bmatrix},\ D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

An important property of the proposed scheme is that it allows both secret dispersal as a general threshold scheme and encryption of data at the same time. That is, one cannot extract original data even though the stored data P is available since deciding the original matrix using the eigenvector is NP-hard problem. Refer to the example above. There exist infinite ways to form P matrix by arbitrarily deciding the s, t, and u values. As a result, the proposed scheme offers high availability and security as well as complete recovery of original data even though some partial damage to the data occurs.

3.3 Data Recovery Scheme

We next show how the original data is recovered using the matrix diagonalization method.

• Definition 2.

Assume an $n \times n$ matrix A can be converted to a diagonal matrix D, which is called diagonalizable. Then there exists an invertible $n \times n$ matrix P such that

$$P^{-1}AP = D. \tag{15}$$

The process of finding matrices P and D is called diagonalization. First, it is worth to notice that if D is a diagonal matrix with diagonal entries $\lambda_1, \ldots, \lambda_n$, then for i=1,…,n

$$De_i = \lambda e_i. \tag{16}$$

Hence, the standard basis vectors $e_1, \ldots, e_n$ are eigenvectors of D. In particular, the eigenvectors of D are linearly independent. More generally, we have following Theorem 2.

• Theorem 2.

(Criterion for Diagonalization)

Let A be an $n \times n$ matrix.

1. A is diagonalizable if and only if it has n linearly independent eigenvectors.
2. If A is diagonalizable with $P^{-1}AP = D$, then the columns of P are eigenvectors of A and the diagonal entries of D are the corresponding eigenvalues.
3. If $\{v_1,\ldots,v_n\}$ are linearly independent eigenvectors of A with corresponding eigenvalues $\lambda_1,\ldots,\lambda_n$, then A can be diagonalized by

$$P = [v_1 v_2 \cdots v_n] \text{ and } D = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix}. \tag{17}$$

• PROOF.

Let P be a matrix with columns of n-vectors $v_1,\ldots,v_n$ and D be a diagonal matrix with diagonal entries $\lambda_1,\ldots,\lambda_n$, respectively. Then

$$AP = A[v_1 v_2 \cdots v_n] = [Av_1 Av_2 \cdots Av_n]. \tag{18}$$

$$[\lambda_1 v_1 \lambda_2 v_2 \cdots \lambda_n v_n] = [v_1 v_2 \cdots v_n] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix} = PD. \tag{19}$$

If A is diagonalizable, with $P^{-1}AP = D$, then $AP = PD$. Hence, $Av_i = \lambda_i v_i$, i=1,…,n. So, the λ_is are eigenvalues and the v_is are corresponding eigenvectors.

Suppose that A has n linearly independent eigenvectors, say, $v_1 v_2 \cdots v_n$ (the columns of P). If $\lambda_1,\ldots,\lambda_n$ are the corresponding eigenvalues, then $Av_i = \lambda_i v_i$, i=1,…,n. If D is a diagonal matrix with diagonal entries $\lambda_1,\ldots,\lambda_n$, then $AP = PD$ by (18) and (19). Because P is a square matrix with linearly independent columns, it is invertible. Hence, $P^{-1}AP = D$ and A is diagonalizable [10].

$$P^{-1}AP = D. \tag{20}$$

If we multiply P matrix to both sides in upside form, Eq. (20) becomes $PP^{-1}APP^{-1} = PDP^{-1}$. Since $PP^{-1} = I$, $P^{-1}P = I$ (I: identity matrix),

$$\begin{aligned} A &= PDP^{-1} \\ &= \left[u^{(1)} \cdots u^{(n)}\right] \begin{bmatrix} \lambda_1 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_n \end{bmatrix} \left[u^{(1)} \cdots u^{(n)}\right]^{-1}. \end{aligned} \tag{21}$$

P: Set of Eigenvectors, D: Decryption Key consisting of eigenvalues.

Therefore, the original data can be recovered from P that was stored in the storage nodes by $A = PDP^{-1}$. For the same example of Section 3.2, we get original data by

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 2 & 0 \\ -3 & -5 & 2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 4 & -5 & -2 \\ -2 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ -4 & 6 & 2 \\ 16 & -15 & -5 \end{bmatrix}$$

4 Performance Evaluation

The substantial body of prior work in building highly-available systems provides a clear metric for evaluating information availability: the probability that a desired piece of information can be accessed at any given point in time. Assuming uncorrelated failures, this probability can be computed from the probabilities that required system components are available. For example, a *p-m-n* threshold scheme requires at least *m* of the *n* storage nodes containing the shares to be operational to perform a read. If f_{node} is the probability that a storage node has failed or is otherwise unavailable, then the availability of the stored information is

$$\text{Availability}_{\text{read}} = \sum_{i=0}^{n-m} \binom{n}{i} (f_{node})^i (1-f_{node})^{n-i} . \tag{22}$$

For writes, the computation of availability depends upon the system's design. A system could require that all of *n* specific nodes be operational for a write to succeed. This approach clearly has poor availability. To improve write availability in a system with $N > n$ storage nodes, the write operation can attempt to write the shares at different storage nodes until it has completed *n* writes. Alternatively, a system could allow a write to complete when fewer than *n* shares have been written. This requires more work during failure recovery or reduces read availability of the stored data (because *n* is effectively lower for that data). We calculate write availability using the last approach. Thus, *m* storage nodes of the *N* storage nodes in the system must be available for a write to be performed, and write availability is [11]

$$\text{Availability}_{\text{write}} = \sum_{i=0}^{N-m} \binom{N}{i} (f_{node})^i (1-f_{node})^{N-i} . \tag{23}$$

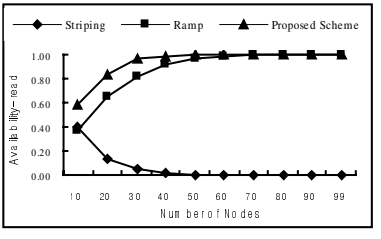


Fig. 3. Comparison of read availability

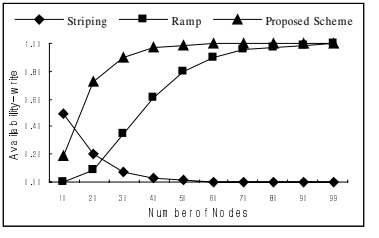


Fig. 4. Comparison of write availability

Figure 3 and 4 show the comparison of the read and write availability of the proposed scheme along with striping and ramp scheme (*p-2-n*). Here up to 100 nodes were

tested. The figures reveal that the proposed scheme display incomparably better availability than striping while it is consistently better than the ramp scheme for about 15% for both read and write operation.

5 Conclusion and Future Work

In this paper we have presented a new data dispersal/encryption scheme to improve the availability and security of distributed storage system at the same time by using some important properties of matrix. As a result, the proposed scheme offers high availability and security as well as complete recovery of original data even though partial damage occurs to the data. Analysis shows that the availability is improved about 15% compared with an existing representative scheme for both read and write operation, while it allows secure storage.

Necessity of storage system providing safe data storage is inevitable as dependency on storage device is embossed at this point. Therefore more efforts are required to develop effective data distribution and encryption techniques customized to various applications and operation environment. A formal design methodology which identifies the tradeoff between the performance measures such as availability, security, and operation speed also needs to be developed.

References

1. Yvo Desmedt: Some Recent Research Aspects of Threshold Cryptography: Information Security, First International Workshop ISW '97, Lecture Notes in Computer Science #1196, M. Mambo E. Okamoto, E. Davida, Ed., (1997) 158–173
2. A. Shamir: How to Share a Secret: Comm. ACM. (1979) 612–613
3. E. Karnin, J. Greene, M. Hellman.: On Secret Sharing Systems: IEEE Trans. Information Theory (1983) 35–41
4. G. R. Blakley, Catherine Meadows: Security of ramp schemes: Advances in Cryptology, Springer-Verlag, (1985) 242–268
5. A. De Santis and B. Masucci: Multiple Ramp Schemes: IEEE Trans. Information Theory (1999) 1720–1728
6. Jay J. Wylie, Michael W. Bigrigg, John D. Strunk, Gregory R. Ganger, Han Kiliccote, Pradeep K. Khosla: Survivable Information Storage systems: IEEE Computer (2000)
7. Y. Deswarte, L. Blain, J. Fabre: Intrusion Tolerance In Distributed Computing Systems: Proc. IEEE Symp. Security and P privacy, IEEE CS Press, Los Alamitos, Calif (1991) 110–120
8. George Nakos, David Joyner: Linear Algebra with Applications, Brooks/Cole USA (1998) 442–455
9. David Kincaid, Ward Chenney: Numerical Analysis Mathematics of Scientific Computing. 2nd edn, Brooks/Cole USA (1996) 274–276
10. Birkhauser: Linear Algebra, Birkhauser Boston (1997) 216–219
11. Mehmet Bakkaloglu, Jay J. Wylie, Chenxi Wang, Gregory R. Ganger: On Correlated Failures in Survivable Storage Systems: School of Computer Science Carnegie Mellon University, Pittsburgh, PA15213 (2002)

A DTD for an XML-Based Mathematical Modeling Language

Marcos Calle[1], S. Lozano[1], Kate Smith[2], Terence Kwok[2] and Juan J. Domínguez[3]

[1] Dpto. Organización Industrial y Gestión de Empresas. Escuela Superior de Ingenieros. Universidad de Sevilla. Camino de los Descubrimientos s.n., 41092 Seville, Spain
{mcalle, slozano}@us.es
[2] School of Business Systems. Monash University, Clayton, Victoria 3800, Australia
{kate.smith, terence.kwok}@infotech.monash.edu.au
[3] Dpto. Lenguaje y Sistemas Informáticos. Escuela Superior de Ingeniería. Universidad de Cádiz. C/Chile, nº 1, 11003 Cádiz, Spain
juanjose.dominguez@uca.es

Abstract. This paper contains a proposal for an XML-based modeling language aimed at operations research problems. It has been called Operations Research Markup Language (ORML). There are a number of mathematical modeling languages developed by different companies, but they are proprietary and each one has its own syntax. ORML tries to bring to this field the standardization and interoperability benefits of XML.

1 Origin

This proposal of modeling language appears in an environment characterized by the existence of multitude of modeling languages for operations research problems: GAMS [3], AMPL [5], AIMMS [2], MPL [8], LPL [6], etc. In order to develop this proposal, a thorough study has been made about them. We have obtained the main characteristics of each language that make it possible to model problems. The next step on the development was to obtain a list of the common characteristics of these languages. Next, some additional characteristics have been included into the ORML proposal. This proposal of modeling language is based on two existing standards: XML and MathML.

1.1 XML

XML (eXtensible Markup Language) [4] covers the necessity of creating structured documents, making possible to exchange information between distant and independent applications. The independence character means in the first place that the mentioned applications can be implemented in multitude of languages (VB, C++, Java, etc.), and refers in second place the platform in which the mentioned application is executed (UNIX, Windows, etc.). XML is a metalanguage because it is a language used to create other languages. In addition, XML documents are legible by both humans and computers.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 968–977, 2003.
© Springer-Verlag Berlin Heidelberg 2003

1.2 MathML

MathML (Mathematical Markup Language) [1] is an XML-based language. The target of this language is to express mathematical information. MathML has a double syntax. The difference of this syntax is based on the form of representing the information. The first syntax focuses on the representation format, whereas the second syntax focuses on the content of the information.

2 ORML

In this paper, we make only a summarized description of the ORML proposal, accompanied by an illustration. The complete DTDs of this proposal are available in the following URL http://io.us.es/mcs/orml_dtd.zip.

2.1 ORML Objectives

ORML has been designed to cover the following objectives:

- To express operations research problems that are not possible to express by means of MathML.
- To facilitate the standardization of a mathematical modeling languages, because at the present time each company has its own modeling language.
- To convert problems modeled with ORML to anyone of the other mentioned languages, and as far as possible to allow the inverse conversion, that is, to translate problems in any of the existing mathematical modeling languages to ORML.

2.2 ORML Structure

The structure of a problem consists, like in most of the existing mathematical modeling languages (AMPL, AIMMS, etc) of two files: problem model and problem data instance. The problem model consists of the declaration of the problem in parametric form, that is, this model contains the declarations of the objective functions, variables, restrictions, parameters, etc, without assigning numeric values. The problem data instance consists of the data of an specific case, that is, the numerical values of the parameters, bounds of the variables, initial values of the variables (seed values). The use of this structure contributes to allow for declaring the problem model only once, whereas it may be used in multiple occasions. For the later uses of this problem, the user has to update only the problem data instance, without having to modify the problem model. In our proposal, both files, problem model and problem data instance, are XML files.

2.3 Characteristics of ORML

In this section, the main elements and characteristics of ORML are described.

2.3.1 Characteristics of the Problem Model:

The problem model will contain the following sections:

- Sets
- Parameters
- Variable
- Macros
- Restrictions
- Functions objective

Set

It consists of a certain group of elements that share some important characteristic. Example of sets: "cities", "products", "warehouse", etc.

- The sets can be one-dimensional or multidimensional.
- Creation of new sets by means of operations on existing sets. These operations are union, differentiation, intersection or Cartesian product.
- The elements of the sets can be referred in the expressions.
- The main function of the sets is to act like domain index of the parameters, restrictions, variables, summation, etc.

Parameter

It is a well-known value previous to the resolution of the problem.

- They can be simple, one-dimensional or multidimensional.
- Each one-dimensional or multidimensional parameter will have an associated domain index. This parameter will have the same dimension that its index.
- The parameters can be referred in the algebraic expressions of the macros, restrictions or functions objective.

Variable

It is an unknown value previous to the resolution of the problem.

- The variables can be simple, one-dimensional or multidimensional.
- Each one-dimensional or multidimensional variable will have an associated domain index. This variable will have the same dimension that its index.
- The variables can be referred in the algebraic expressions of the macros, restrictions or functions objective.
- The variables must be one of the following basic types: continuous, binary or integer.

Macro
It consists of an algebraic expression that appears many times in the model. A macro must be declared only one time, nevertheless It can be referred in any locations where it needs to appear.

- The macros can be used in restrictions and objective functions.

Restriction
It consists of either an equality or an inequality, which it must be fulfilled in the problem.

- The restrictions can be simple, one-dimensional or multidimensional.
- Each one-dimensional or multidimensional restriction will have an associated domain index. This restriction will have the same dimension that its index.

Objective Function
It consists of an algebraic expression in which referenced variables appear. These objective functions also show whether the expression must be maximized or minimized.

Expression
- In the expressions, a particular set element can be referred, e.g.:

$$production(t) + stock(t-1) = venta(t) + stock(t) \tag{1}$$

- The use of mathematical functions and operators is allowed in the expressions. ORML contains trigonometric, hyperbolic, inverse trigonometric, inverse hyperbolic functions, functions on sets and series. It also contains comparison and arithmetic operators.

Condition
It consists of either an equality or an inequality, which must be fulfilled in the problem to allow for the inclusion of a certain algebraic expression. These conditions can be included within the summation, products and conditional element "if-then-else".

- The logic operators (AND, OR and XOR) can be used to create more complex conditions.

2.3.2 Characteristics of the Problem Data Instances:

- Definition of parameters by means of tables. This defining form is useful for multidimensional parameters.

- Definition of bounds of the variables. This definition is optional. In a problem, some of the variables can have bounds. This defining form enables to reduce the number of constraints.
- Definition of the initial values of the variables. This is useful for some resolution software, which need initial values for resolving the problems.

2.4 MathML Imported Elements

Some of the imported elements of MathML have been modified. In this section appear two lists, the first list contains the non-modified imported elements, the second list contains the modified imported elements.

Table 1. Non-modified MathML imported elements

<abs>	<and>	<apply>	<arccos>	<arccosh>
<arccot>	<arccoth>	<arccsc>	<arccsch>	<arcsec>
<arcsech>	<arcsin>	<arcsinh>	<arctan>	<arctanh>
<cartesianproduct>	<ci>	<cn>	<cos>	<cosh>
<cot>	<coth>	<csc>	<csch>	<diff>
<divide>	<else>	<exp>	<if>	<intersect>
<ln>	<log>	<matrix>	<matrixrow>	<minus>
<or>	<plus>	<power>	<quotient>	<rem>
<root>	<sec>	<sech>	<sin>	<sinh>
<tan>	<tanh>	<then>	<times>	<union>
<xor>				

Table 2. Modified MathML imported elements

<condition>	<degree>	<product>	<set>	<sum>

3 Example

This section contains a complete example. First, the problem is expressed with mathematical notation. Then, the same problem is expressed with ORML. In both cases, the information is divided in two parts. The first part includes the problem model, whereas the second part includes the problem data instance.

3.1 Problem with Mathematical Annotation

The next two sections are the problem model and problem data instance.

Problem Model

SETS (2)

river

city

PARAMETERS

$supply_{river}$

$demand_{city}$

$distcost_{river,city}$

VARIABLES

$distribute_{river,city}$

CONSTRAINTS

source :

$$\sum_{city}\left(distribute_{river,city}\right) = supply_{river} \qquad ,\forall river$$

destination :

$$\sum_{river}\left(distribute_{river,city}\right) = demand_{city} \qquad ,\forall city$$

OBJECTIVES

totalcost :

$$Min \sum_{river}\sum_{city}\left(10000000 \times distcost_{river,city} \times distribute_{river,city}\right)$$

Problem Data Instance

$$
\begin{aligned}
&\text{SETS} \\
&\quad river = \{1,2,3,4\} \\
&\quad city = \{1,2,3,4,5\} \\
&\text{PARAMETERS} \\
&\quad supply_{river} = [50 \quad 60 \quad 50 \quad 50] \\
&\quad demand_{city} = [30 \quad 20 \quad 70 \quad 30 \quad 60] \\
&\quad distcost_{river,city} = \begin{bmatrix} 16 & 16 & 13 & 22 & 17 \\ 14 & 14 & 13 & 19 & 15 \\ 19 & 19 & 20 & 23 & 999 \\ 999 & 0 & 999 & 0 & 0 \end{bmatrix}
\end{aligned}
\tag{3}
$$

3.2 Problem with ORML Annotation

This example is shown in the appendix. It consists of two XML files, for this reason the appendix contains the following two figures:

- Figure 1. Problem model.
- Figure 2. Problem data instance.

4 Summary and Discussion

In this paper, XML and MathML have been used in the development of a proposal for a new mathematical modeling language. The syntax of the elements of this language is similar to those of existing languages (GAMS, AMPL, AIMMS, LPL and MPL). ORML tries to standardize such syntax. This proposal so far consists of two DTD: the first DTD describes the structure of the problem model, the second DTD describes the structure of the problem data instance. However, DTDs are known to have limitations:

- They cannot define the datatypes.
- They do not contain namespaces that allow to import others schemas.
- They do not specify either the maximum or minimum number of times the elements can appear.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE model SYSTEM "orml_model.dtd">
<model name="metrowater">
 <sets>
  <set name="river"/>
  <set name="city"/>
  <set name="rivercity">
   <apply>
    <cartesianproduct/>
    <ci type="set" name="river"/>
    <ci type="set" name="city"/>
   </apply>
  </set>
 </sets>
 <parameters>
  <parameter name="distcost" index="rivercity"/>
  <parameter name="supply" index="river"/>
  <parameter name="demand" index="city"/>
 </parameters>
 <variables>
  <variable name="distribute" index="rivercity" type="continuous"/>
 </variables>
 <constraints>
  <constraint name="source" index="river" relation="eq">
   <lhs>
    <apply>
     <sum/>
     <indexdomain index="city"/>
     <bvar>
      <ci type="variable" name="distribute"/>
     </bvar>
    </apply>
   </lhs>
   <rhs>
    <apply>
     <ci type="parameter" name="supply"/>
    </apply>
   </rhs>
  </constraint>
  <constraint name="destination" index="city" relation="eq">
   <lhs>
    <apply>
     <sum/>
     <indexdomain index="river"/>
     <bvar>
      <ci type="variable" name="distribute"/>
     </bvar>
     <apply>
      <ci type="variable" name="distribute"/>
     </apply>
    </apply>
   </lhs>
   <rhs>
    <apply>
     <ci type="parameter" name="demand"/>
    </apply>
   </rhs>
  </constraint>
 </constraints>
 <objectives>
  <objective name="totalcost" type="minimize">
   <apply>
    <sum/>
    <indexdomain index="river"/>
    <bvar>
     <ci type="variable" name="distribute"/>
    </bvar>
    <apply>
     <sum/>
     <indexdomain index="city"/>
     <bvar>
      <ci type="variable" name="distribute"/>
     </bvar>
     <apply>
      <times/>
      <cn>10000000</cn>
      <ci type="parameter" name="distcost"/>
      <ci type="variable" name="distribute"/>
     </apply>
    </apply>
   </apply>
  </objective>
 </objectives>
</model>
```

Fig. 1. Problem model.

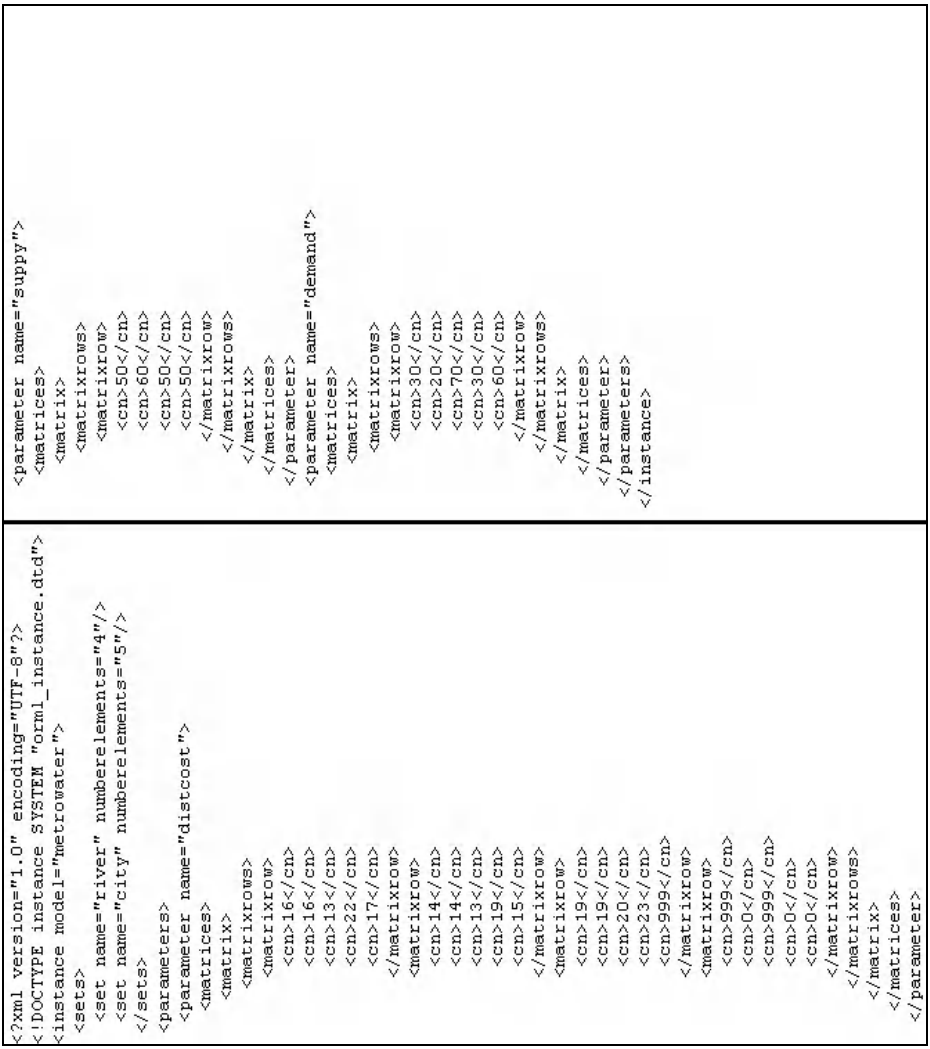


Fig. 2. Problem data instance.

Thus, the authors are working on defining appropriate XML Schemas [9]. In a later step, XSLT [10] will be used to transform ORML models to the syntax used by existing languages as well as to other formats such as MPS [7].

References

1. Ausbrooks, R., Buswell, S., Dalmas, S., Devitt, S., Diaz, A., Hunter, R., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical Markup Language (MathML). Version 2.0. World Wide Web Consortium.
http://www.w3.org/TR/2001/REC-MathML2-20010221 [21st February 2001]

2. Bisschop, J., Roelofs, M.: AIMMS: The Language Reference. Paragon Decision Technology B.V. (2001)
3. Brooke, A., Kendrick, D., Meeraus, A.: GAMS: a user's guide. Scientific Press, San Francisco (1988)
4. Extensible Markup Language (XML). 2nd edn. Version 1.0. World Wide Web Consortium. http://www.w3.org/TR/2000/REC-xml-20001006 [6th October 2000]
5. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL : a modeling language for mathematical programming. Danvers, MA Boyd & Fraser (1993)
6. Hürlimann T.: Reference Manual for the LPL Modeling Language. Version 4.43, Departement for Informatics, Fribourg, [August 2002]
7. IBM Inc, "Passing Your Model Using Mathematical Programming System (MPS) Format", http://www6.software.ibm.com/sos/features/featur11.htm
8. MPL Manual. Maximal Software, Inc. http://www.maximal-usa.com/mplman/mplwtoc.html. [octubre 28th 2002]
9. XML Schema. Version 1.0. World Wide Web Consortium. http://www.w3.org/XML/Schema [2nd May 2001]
10. XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium. http://www.w3.org/TR/xslt [16th November 1999]

A Compress-Based Association Mining Algorithm for Large Dataset

Mafruz Zaman Ashrafi, David Taniar, and Kate Smith

School of Business Systems, Monash University, PO BOX 63B, Clayton 3800, Australia.
{Mafruz.Ashrafi,David.Taniar,Kate.Smith}@infotech.monash.edu.au

Abstract. The association mining is one of the primary sub-areas in the field of data mining. This technique had been used in numerous practical applications, including consumer market basket analysis, inferring patterns from web page access logs, network intrusion detection and pattern discovery in biological applications. Most of the traditional association-mining algorithms assume that whole dataset can be loaded in the main memory. Hence, problem arise when such algorithms is applied in large dataset. In this paper we present a new algorithm for association mining. Our algorithm is efficient when the size of dataset is huge that cannot be load in the main memory. The proposed algorithm also reduces the frequent itemsets search space, by eliminating non-frequent 1-itemsets after the first pass. Our performance evaluation shows algorithm outperforms Apriori algorithm in different datasets.

1 Introduction

Data mining is iterative and interactive processes that explore and analyze voluminous data in order to discover valid, novel and meaningful patterns, associations or rules, using computationally efficient techniques [1]. It is related to the sub area of statistics called exploratory data analysis, which has similar goals and relies on statistical measures and also closely related to the sub areas of artificial intelligence called knowledge discovery and machine learning.

Data mining has been attracted huge attention in numerous research communities due to its wide applicability in many areas such as retail industry, financial forecast, decision support and intrusion detection [2]. Data mining methods include associations clustering, classification, and prediction. One of the most important fields of the data-mining domain is the association mining. Many interesting and efficient mining of association rule algorithms have been proposed in the different mining literature [2, 3, 4, 5, 6, 8, 9, 10, 11]. In this paper we present an efficient association-mining algorithm for large dataset.

Agarwal et al first introduced discovering association rules from the market basket dataset. The Apriori algorithm is one of the most popular algorithms in the mining association rules. There are number of extensions of the Apriori algorithm such as Partition [5], DHP [11], etc.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 978–987, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Discovering all association rules from very large dataset, by using support-confidence based framework is not a trivial task. The search space is exponential in the number of database attributes and with millions of I/O minimizations become paramount [2].

In this paper we present an algorithm called Compress that has significant difference from the Apriori and all other algorithms those are extended from Apriori algorithm. The compress algorithm loads the dataset into main memory and organized them into different vertical partition table based on the first item of the each transaction and compresses them after a specific limit. The algorithm also removes items from the transactions if those particular items don't have user specific support level.

The rest of paper organized as follows: In Section 2 we describe background of association mining and its problem. We mention the problem of Apriori algorithm in Section 3. In Section 4 we describe our proposed algorithm. The performance evaluation and comparison studies are described in section 5 and we conclude at Section 6.

2 Association Mining Rules: A Background

Association mining can be described as a correlation of events which means, events those are frequently observed together or in other words an association algorithm creates rules that describe how often events occur together. The prime task of association mining is to discover a set of attributes shared among a large number of objects in a given database [1]. Two important measures for association rules are support (s) and confidence (c), the former can be defined as an associated measure of statistical significance of an *itemset*, and the later is an indication of the strength of the rule. A rule is frequent if its support is greater than the user specified support and strong if its confidence is greater than the user specified confidence. The following formula could measure the confidence of an association rule.

$$Confidence = \frac{Support\ (LHS \cup RHS)}{Support\ (LHS)}$$

Consider an example of supermarket sales database shown in the figure 1(a). There are few different items and few sample transactions of customers. In order to identify purchase/shopping pattern from such dataset, the association rule intend to established correlation among the different items and apply a specific level of support and confidence on those correlations and discover rules. Finally, supermarket can use those association rules knowledge for promotions, shelf placements, inventory control, customer service and etc. Figure 1(b) shows the number of time each item appears in the supermarket customer transaction and correlation (up to level 2) of items, those have minimum *50%* of support.

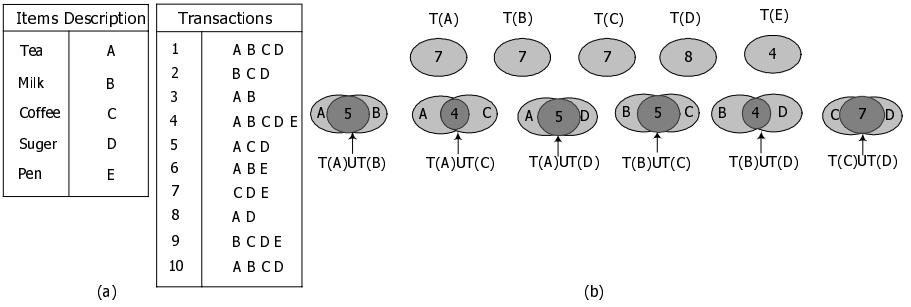


Fig. 1. (a) Supermarket dataset (b) correlation of items

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of distinct attributes, also called *literals*. Let D be the databases of transactions, where each transaction T has a set of items such that $T \subseteq I$, and unique identifier (*tid*). The set of items also known as *itemset* and the number of items in an itemset is called the length of an *itemset*. The support of an itemset X, is the number of transactions it occur as a subset. An itemset is frequent if that itemset has a user specific support. An *association rule* is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ are *itemset*, and $X \cap Y = \phi$. Here, X is called antecedent, and Y consequent. The rule $X \Rightarrow Y$ has *support s* in transaction set D, if and only if $s\%$ of transactions in D contains $X \cup Y$ and holds with *confidence c* in transaction set D, if and only if $c\%$ of transactions in D that contains X also contains Y, and can be calculated by using the formula 1.

The problem of the association rule can be divided into two subgroups. Firstly, generating all itemsets that have user specific support i.e. *frequent itemsets*. Secondly, generating *confidence rules* from those frequent itemsets, which satisfy the user specific confidence. The formal problem is nontrivial and most crucial factor that affects the performance of the association rules. Moreover, this problem affects more adversely, if numbers of distinct attributes in database are large. For example, if a particular database has m number distinct attributes there can be 2^m numbers of possible distinct large itemsets and the problem is to find out those itemsets that have user specific support. Where as, the later problem is relatively easy and straightforward. For generating *confidence rules* from *frequent itemsets* those have user specific confidence in the $X/Y \Rightarrow Z$ form, where $Y \subset X$ and X is frequent

2.1 Apriori Algorithm

To resolve above mention problem [2] present this algorithm. This algorithm is considered as one of the most popular Association Rule Mining algorithms, however this algorithm need as many database scan as to find longest itemset from the dataset. The algorithm has the following steps:

– Large itemsets are generated through iterations and each iteration database is scanned once.

- In first iteration, algorithm determines large *1-itemsets* (i.e. itemset of length 1) by simply counting each item occurrence in transactions.
- Subsequent iteration consist two phases, firstly candidate sets C_k generation by applying *Apriori-gen* function, using the large itemsets L_{k-1} found in *(k-1)th* iteration those and store in a hash tree. A node of that hash tree either contains a list of itemsets (a leaf node) or a hash table (an interior node), secondly, the database is scanned and support of the C_k is counted.
- Pruning away those candidate subsets from C_k, those have not required support.

3 Problems of Apriori

From the above discussion, it is clear that Apriori association-mining process is an iterative process, and each iteration dataset needs to be read. Reading from disk involves I/O operation and hence computationally intensive. To read disk resident data at every pass of the association mining-algorithm resultant large number of disk I/O operations. On the other hand, it is not always feasible to keep whole dataset within the main memory if dataset is large. As a result, need multiple scans, which incur some additional computational cost and degrade overall mining performance even for small size dataset.

The performances of Apriori association-mining algorithm degrade further, if it requires to read disk resident data for every pass in order to generate itemsets then it will not able to perceive those transactions that have identical itemsets and therefore will unnecessarily occupy resources for repeatedly generating itemsets from such identical transactions.

In this work, our goal is to reduce I/O operations and computational cost of association mining by organizing each database transaction in vertical partitioned-based table (i.e. based on the first item of a transaction) and dynamically compress those tables in order to utilize main memory more efficiently. Furthermore, after first pass we prune away those non-frequent *1-itemsets* from each transaction of the dataset in order to reduce unnecessary itemsets generation.

4 Proposed Algorithm

In this section we discuss different components of our association-mining algorithm to elaborate it more precisely. Before embarking on the algorithm description, we will briefly discuss two assumptions that we made in this work. Firstly, we assume each transaction in dataset is either (*TID,* $i_1\ i_2\ i_3 \ldots \ldots\ i_j$) or ($i_1\ i_2\ i_3 \ldots \ldots\ i_j$) form. Secondly, all items in transactions are sorted in a lexicographic order. It is worth to mention that similar kinds of assumptions had been made in number of previous association rule mining research [2, 5].

Our proposed algorithm works in two different phases. The *first phase* is known as *reading* phase. In this phase, all transactions are read and based on the first item of

each transaction, we construct different range of partition table as shown in the Figure 2(a) and placed each transactions to its corresponding table. Since transaction in a dataset may have thousand of items and not necessarily every transaction will begin with the same item, hence this method will create number of tables. The main benefit of this approach over the horizontal partition [5] of dataset is that, there is possibility to find transaction with same itemsets (even if we do not find transaction with the same itemsets in first pass, possibility of finding it is much higher after we prune itemsets from transaction, this method will be discussed in details in the following paragraph) in the later pass. Hence, it does not generate frequent itemsets from similar transactions (i.e. transaction with identical itemsets) for second time. Furthermore, this approach guaranteed that transactions with identical itemsets would only have single occurrence in the partition table so consumes less space in the main memory.

To understand this more precisely consider previous example at figure 1(a), where we have ten transactions consist of five different items. For each of those items we form a partition table (a hash table). Depending on the first item of a transaction we placed it to the corresponding table and set it counter into 1. Counter value will increment if any transaction with similar itemset is found later read. This phase also counts the support of each itemsets of length 1 (*1- itemset*). The size of each partition table defined in advance. When a particular partition table reached its maximum size the corresponding table is compressed and temporarily kept in the secondary memory in order to utilize the main memory more efficiently. When a partition table reaches its maximum limit a compression technique is used in order to compress that table.

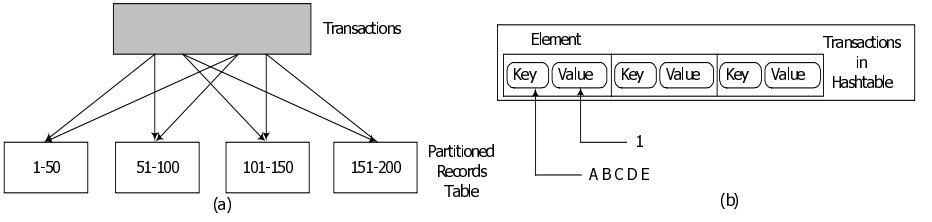


Fig. 2. (a) Different range partition table based on the first item of a transaction (b) Elements of partition table.

The *second phase* is known as *iteration* phase. At the beginning of this phase actual support of *1- itemsets* is generated. After generating all frequent *1-itemsets* each of the partition table will be iterated and corresponding itemsets (length *2,3,4, ... n*) will be generated. During iteration, elements of each partition table are examined carefully. Every non-frequent *1-itemsets* are discarded if found in any transaction because of the fact that itemset those are infrequent in initial pass cannot be able to generate frequent itemsets in the later pass and finally rehash those transactions into the corresponding partition table for a second time.

For example consider our previous example at Figure 1, after completion of the first phase all non-frequent *1-itemsets* are marked (in this case item *E* if we set *min_support = 50%*). During first pass of the *iteration* phase each transaction is ex-

amined, remove item *E* from each transaction that contains it and rehash those elements in the partition table. Figure 3 shows initially partition table has 9 elements and 4 elements contains item *E*. When we remove item *E* from each elements and rehash them for a second time we will have partition, which will have 6 elements as shown in the figure 3.

Conter	Transaction
2	A B C D
1	B C D
1	A B
1	A B C D E
1	A C D
1	A B E
1	C D E
1	A D
1	B C D E

Partition table in 1st phase

Conter	Transaction
3	A B C D
2	B C D
2	A B
1	A C D
1	C D
1	A D

Partition table in 2nd phase

Fig. 3. Elements in Partition table.

The notation used in this paper is shown in figure 4. Items of the dataset *I* will vertically be fragmented into different partitions *nP* and size *sP*. The size of a partition table *sP* specify how many transaction can be hold before compress that particular table in order to utilize main memory more efficiently. The vertical items range (i.e. which partition table will store based on the starting item if each transaction) of each partition table *vR* can be calculated by dividing the number of items *I* by the total number of partition table.

s	Minimum support;
c	Minimum confidence;
C	Candidate itemset;
N	Total number of transactions;
I	Total number of items;
nP	Number of partition tables;
sP	Size of each partition table;
vR	Vertical range of each table;
f	First item of a transaction;
F^i	Frequent itemset of length I;

Fig. 4. Notation

The algorithm is shown in the figure 5. In the first phase, algorithm will scan the whole dataset and insert each transaction in a particular partition table and count each *1-itemsets* occurrence. In the second phase each compress partition table will load into the main memory, remove every *1-itemsets* that don't have a specific support level and

rehash them into the table for a second time. The *generate_itemset()* method takes each transaction as an input and generate itemsets of corresponding length in each iteration. Since the algorithm prune away elements after the first phase, *generate_itemset()* method generate less *itemsets* of various length. The pruning operation is applied after each iteration and remove *itemsets* those don't have corresponding support.

```
/*1st phase*/
for i = 1 to N do begin
  insert_into_table(i);
  count_item(i);
end;

/*2nd phase*/
generate_frequent()
for i = 1 to nP do
begin
  t = prune(i)
  if(t! = i)
    rehash(t);
  generate_itemset(i);
end;
```

```
insert_into_table(i)
begin
  f = firstItem(i)
//find partition table for f
  pT = vR(f)
  If(t < sP)
    put(i);
  else
  {
    compress(T)
    create table(T)
    put(i);
  }
end;//insert_into_table
```

Fig. 5. Algorithm

5 Performance Evaluation

In experiments we used a *1.6GHz I*ntel P4 machine with 512MB main memory. We used different synthetic datasets that have been proposed by [2] and those generated datasets were previously used for evaluating performance in various mining algorithms. We implemented our algorithm in Java 1.3. Since we implemented our algorithm in Java, we were looking for third party Apriori algorithm in Java in order to compare the performance our proposed algorithm with Apriori. But we were not able to manage any third party Apriori algorithm implemented in Java.

Let D denote the number of transactions, T the average transaction size, I the size of the maximal potentially frequent itemset, L the number of maximal potentially frequent itemsets and N the number of items. Experiments are conducted on datasets with different values of T and I. The parameters are shown in figure 6. The nature of datasets (sparse or dense) is depending on the L and N values. In order to generate dense dataset we keep both L and N values within 100.

Name	|T|	| I |	|D|	Size in MB
T18.I8.D10M	20	8	10M	490
T20.I12.D10M	20	12	10M	500
T10.I8.D10M	10	8	10M	321
T22.I12.D10M	22	12	10M	641

Fig. 6. Dataset with different parameters

Apriori algorithm we used in the comparison study was implemented in C[1]. Java treats every thing as an object (except few primitive data type such as *int*, *float*, *char* etc.) and object creation is considered as a costliest operation. In the past, various organizations did some basic Input/Output performance comparison between Java and C and result shows C outperforms Java. To give a clear picture of the above mention argument we did some experiment to show how much time a simple and straight forward Java program will take in order to generate frequent itemsets of length-1. Figure 7 (a) shows how much time Apriori implemented in C and a sample Java program took in order to find out all itemsets of length 1. The figure 7 (b) shows how much time took to find out itemsets of length 1, Apriori algorithm implemented in C and our proposed compress association-mining algorithm. The compression algorithm took more time due to the extra overhead that required in Java object creation.

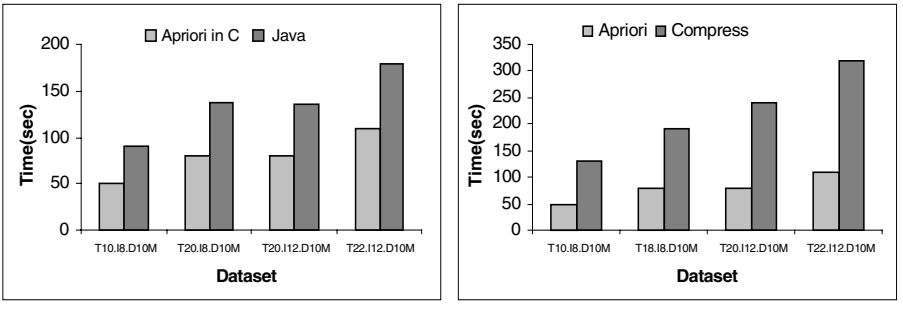


Fig. 7. Time taken for reading and generating itemsets of length 1, from different datasets.

Figure 8 (a, b, c, d) shows the performance comparison between the our proposed Compress and Apriori algorithm. The Apriori algorithm we used in this study has various options. It can store the whole dataset into the main memory during the mining task. In the beginning we tried to run Apriori with this option but we were not successful because the datasets we used in this experiment is large. Hence, we run the Apriori algorithm without loading the dataset in the main memory.

[1] The apriori algorithm was implemented by Christian Borgelt.

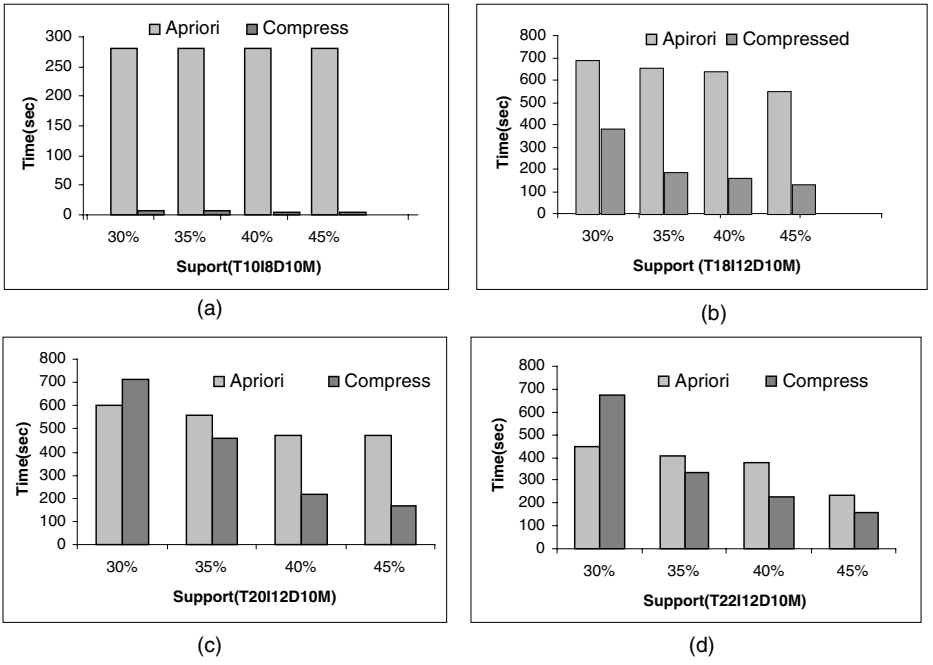


Fig. 8. Execution time.

As from the figure 8 it is clear that both Apriori and compress algorithm took more time if the size of the transaction (i.e. avg. number of items) is higher and the support level is lower. The compress algorithm prunes away those itemsets that don't have a specific support level after the first phase. It prunes away more itemsets from the transaction when support level is high. Due to this fact the proposed algorithm outperform the Apriori in most of the cases. Moreover, in our algorithm we implemented a better technique for generating candidate sets. Whereas, the Apriori algorithm uses the subset operation and hence cost of generating per itemsets is increased in the later pass [5].

Our compress algorithm performs significantly batter than Apriori algorithm in most of the cases. We found compress algorithm achieve best performance with dataset *T10.I8.D10M* in figure 8(a), because after the initial pruning transaction size become smaller. Hence the ratio of identical transactions after first phase is higher than the other datasets. Both Apriori and compress algorithm took more time as expected with the dataset *T20.I8.D10M* shown in the figure 8 (b). This is because the avg. length of size of each transaction which is larger than the previous dataset. In this dataset compress algorithm generate almost equal number of *1-itemsets* for support (35%, 40%, 45%) for this reason total execution time remain the same.

From figure 8 (c) and (d), it is clear that compress algorithm took more time as compared to Apriori algorithm when support is low and datasets have bigger transaction size. This is because, the compression algorithm finds less number of non-frequent 1-itemsets after the first phase and hence extra time required to create some

extra objects. For example, in order to remove 1-itemsets from the each transaction compress algorithm create number of objects and only able to remove small number of 1-itemsets from it. We could reduce some time if implement our compress C/C++.

6 Conclusion

In this paper we examined various issues related with association mining, we also have described an algorithm, which is effectively generate association rules by discovering frequent itemsets from large datasets. An important outcome of our algorithm is to reduce the search space by eliminating *1-itemsets* from the transactions after the first pass. This feature may prove useful for finding frequent itemset from many real life dense dataset. An extensive number of experimental evaluation over various dataset showed that our proposed compress algorithm outperforms the Apriori algorithm in most of the cases.

Reference

1. Mohammed Javeed Zaki, "Parallel and Distributed Association Mining: A Survey", *IEEE Concurrency*, October-December 1999.
2. Mohammed Javeed Zaki, "Scalable Algorithms for Association Mining" *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12 No.2 pp. 372–390 (2000).
3. Rakesh Agrawal and Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules in Large Database", *In Proceedings of the 2oth International Conference on Very Large Databases*, pp. 407–419, Santiago, Chile, 1994.
4. David Wai-Lok Cheung, Vincent T. Ng, Ada Wai-Chee Fu, and Yongjian Fu, "Efficient Mining of Association Rules in Distributed Databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 911–922, 1996.
5. Ming-Syan Chen, Jiawei Han, and Philip S. Yu "Data mining: An overview from a database perspective". *IEEE Transactions on Knowledge and Data Engineering*, Vol 8, No 6, pages 866–883, 1996.
6. Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", *In Proceedings of the 21nd International Conference on Very Large Databases*, pp. 432–444, Zurich, 1995.
7. Jiawei Han, Jian Pei, Yiwen Yin, "Mining Frequent Patterns without Candidate Generation*", In Proc. ACM SIGMOD Intl. Conference on Management of Data*, 2000.
8. Mohammed Javeed Zaki, Ya Pin, " Introduction: Recent Developments in Parallel and Distributed Data Mining" *Journal of Distributed and Parallel Databases,* Vol.11, No.2, 2002.
9. Doug Burdick, Manuel Calimlim, and Johannes Gehrke. "MAFIA: a maximal frequent itemset algorithm for transactional databases" *In Intl. Conf. on Data Engineering*, 2001.
10. Jong Soo Park, Ming-Syan Chen, and Philip S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules", *In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp. 175–186, San Jose, California, 1995.

Engineering Persistent Queue System for a Unified Stock Transaction Platform

Jingcun Wang and Carol Blum

Information Technology Group, Bear, Sterns & Co. Inc.,
383 Madison Avenue, New York, 10179 NY, USA
jcwang@bear.com

Abstract. In building a unified platform for investors to trade securities globally, we need a persistent queue system to hold all the order handling messages before they are actually delivered to the corresponding stock exchange. This paper presents underlying principles of a persistent queue system based on the concept of journal, and its implementation in Java. We contrast performance of our queue with existing products, and show that the design meets the goals of fast I/O performance, specified messages purging, and timely crash recovery.

1 Introduction

Nowadays most secondary stock markets, whether based on auction system or over-the-counter [1], trade securities and derivatives electronically; and almost every stock exchange around the world has its own interface, no matter message or API (application programming interface) oriented, for clients to access its trading system. On the other hand, it is not uncommon for stocks of a company to be listed for trading on various stock exchanges, e.g., China Telecom is listed in Hong Kong as 0728, as well as in NYSE (New York Stock Exchange) as CHA [2]. Therefore, it is desirable for investors to have a uniform platform to access stock exchanges globally. To achieve that goal, we have implemented a system, called GOR (Global Order Routing); and Fig. 1 shows architecture of it.

There are numerous challenges, excluding legal issues and regulations imposed by respective stock exchange and country, to design and implement such a platform, among which a high-performance, scalable, persistent queue is of great importance. Conceptually, that queue should fulfill several functionalities as follows:

1) Preserve every incoming message, such as placing an order, modifying an order, subscribing to price of a specific symbol, and forward to the corresponding processor, while the latter accesses trading system of the specified exchange to complete the operation;
2) Cancel pending orders in the queue as well as in the order book of the stock exchange. In case there is an error input, that order, if not filled, should be cancelled immediately when trader/clerk realizes that.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 988–993, 2003.
© Springer-Verlag Berlin Heidelberg 2003

3) Crash recovery. In case GOR crashes, all pending orders should be recovered quickly and resent to the corresponding stock exchange.

While PERSISTENT message in JMS (Java Message Service) [3] seems to fit the requirement, it lacks the interface to cancel a message in the queue and proprietary products prohibit us from further customizing. On the other hand, the queue deployed in open-source MTA (mail transfer agent) qmail [4] relies heavily on UNIX file system, and it is difficult for that queue to be scalable and portable.

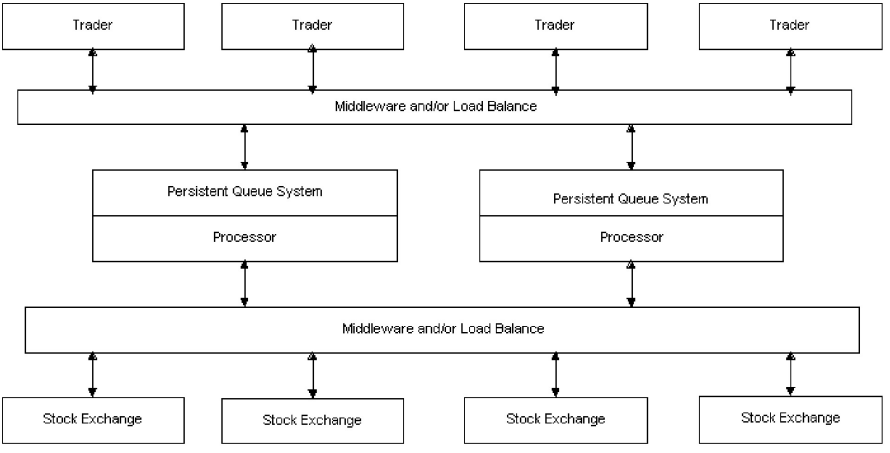


Fig. 1. N-tier Architecture of GOR

In this paper, we propose a high-performance file-system based queue to complete all the requirements. Meanwhile, a prototype has also been implemented by Java in application level for GOR.

Rest of this paper is organized as such. Section 2 presents principle and implementation details of the persistent queue system. Then in Section 3 we provide some simulation results of the system compared with SUN ONE Message Queue. The final section gives concluding remarks and some future directions.

2 Persistent Queue System

A Persistent Queue System (PQS) comprises of multiple queue blocks and a system control file, which records number of queue blocks deployed in current system as well as size (SZ) and location (PATH) of each queue block.

2.1 Queue Block Structure

Physically a queue block is a file either on local file system or mounted on an NFS file server. When PQS starts up, it generates a file with SZ Megabytes at PATH designated in the PQS control file. So, from file system perspective of view, except modifi-

cation time the queue block's directory information will not change during lifetime of PQS.

A queue block (QB) consists of four elements: header, checkpoint, data area, and, tail. Refer to Fig. 2 for detail illustration.

1) Header describes structural information for this QB. It contains positions of checkpoints and starting point of the data area among other parameters.
2) Checkpoint is used to reduce recovery time. There are two checkpoints within each queue block, and they are written alternatively. If PQS crashes when writing one checkpoint, the other can be used to recover.
   The more frequent PQS updates checkpoint, the less time PQS will spend on recovery. However, checkpoint update is a synchronous I/O, which will degrade PQS throughput. So, we need to pick an optimum interval time between each checkpoint update, and that parameter is configurable.
3) Data Area is composed of segments, and a segment contains many trading messages. In GOR, we use FIX (Financial Information Exchange Protocol) [5] to generate exchange specific message; and every segment can contain several of such messages, called item. Along with the real content, every item has a delivery status to denote if it is cancelled or delivered or pending. No trading messages can span over multiple queues.
4) Tail serves for data verification only.

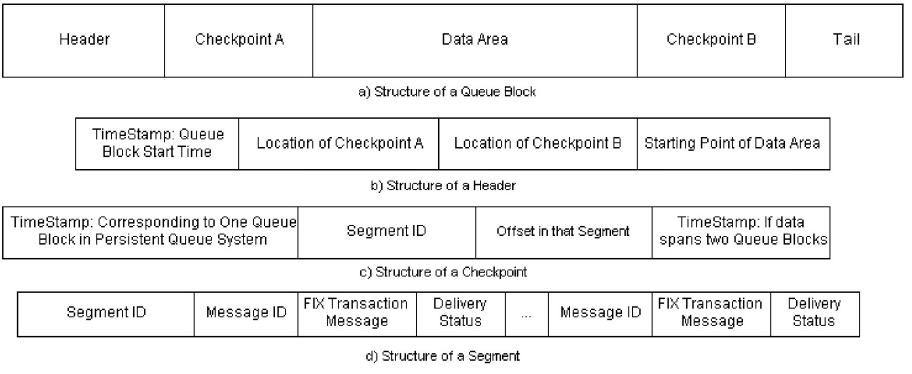


Fig. 2. Structure of a Queue Block in detail

2.2 Incoming Message Handling

When a trader triggers an action, PQS will accept the incoming message sent out from the trader client system and save it into the file system. For instance, if a trader wants to place a market order to buy 100 shares of IBM, trader client system will send out the FIX message via a function call, and the function call will not finish until PQS sends back an acknowledgement. After trader client system gets the acknowledgement, even if GOR crashes, that order is saved and will be re-placed when GOR recovers in the same day (because of high liquidity of stock markets and some SEC – Securities and Exchange Commission regulations, pending orders will only make

sense on the same day that order placed). And later when stock exchange actually processes the order routed by GOR, trader client system will receive information via registered callback functions.

In order to minimize numbers of disk operation, which is synchronous and time-consuming, PQS holds all the incoming messages until a journal (an internal buffer) filled or timeout before flushing to file system, and then acknowledges the trader client systems. The following lists pseudo-code for the saving process.

1. Start a size pre-specified buffer, called a journal;
2. Accept incoming messages, assign each with a unique message id and put it into that journal;
3. If the journal is not full and not timeout, go to Step 2;
4. Assemble contents of the journal to a segment, and flush it to PQS;
5. Acknowledge each trader client system along with the message id.

Generally there may be several queue blocks in PQS. When space of one QB is used up, the journal will be flushed to another. However, that's impossible to have infinite disk space, so we need to recycle queue blocks.

Each queue block has an active message count and range of message ids. When a message is active, it means that message has not been routed to the specified exchange yet. Thus, if active message count drops to zero, that queue block is eligible to be recycled.

2.3 Specified Message Purging

Erroneous orders can result in increased market volatility and significant financial risk and exposure to members and member organization [6], so although order cancellation is not frequent it is of great importance.

If a trader wants to cancel all orders, all queue blocks will be locked, and active message count will be set to 0; in the meantime, corresponding CancelOrder instruction will send to stock exchanges to cancel those already submitted but still remain in the order-book (not filled).

If a trader wants to cancel a specified order with the message id obtained previously, PQS will go through the following steps:

1. Find the corresponding queue block according to range of message id associated with each one;
2. Lock the queue block, and find the message in that queue block;
3. If not delivered, set the delivery status to CANCELLED, and decrement the active message count;
4. If already delivered, forward that request to the specified stock exchange.

2.4 Crash Recovery

PQS might crash because of hardware or software reasons. We want to minimize recovery time so that new incoming messages can be accepted as soon as possible. And, in fact structure of queue block is designed to achieve timely crash recovery. The recovery involves the following steps:

1. Get PATH of each queue block from PQS control file;
2. Retrieve information of each queue block from its head/tail information;
3. Find the queue block with the latest timestamp. And use checkpoint of that queue block to locate the segment being checked;
4. Roll forward from that segment until the end of that queue block.

3 Simulation Study

The Sun[tm] ONE Message Queue is an implementation of the Java[tm] Message Service (JMS) specification, an industry-standard API for handling reliable, asynchronous messaging between Java applications; and according to benchmark comparison, it is 7 times faster than IBM WebSphere Java Message Service Provider [7].

The testing environment is one SUN-Fire-280R Server with 4GB memory; and the two subjects are SUN ONE Application Server 7 integrated SUN ONE Message Queue 3.0.1 system (A) and GOR PQS (B) .

Firstly, we send persistent messages to a queue in A, and the same sized messages to B. This case measures how fast a queue can accept messages. For clarification, we normalize result of A as 1, and result of B is ratio of A. Table 1 lists message per second of B compared with A.

Table 1. Measurement results for PQS on message accepting

Message Size	1K	2K	4K	8K	10K
Messages per Second of B	1.07	1.11	1.19	1.25	1.30

Secondly, we deliver persistent messages from a queue in A, and the same sized messages in B. This case measures how fast a queue can deliver messages.

Table 2. Measurement results for PQS on message delivery

Message Size	1K	2K	4K	8K	10K
Messages per Second of B	1.00	1.00	1.01	1.02	1.02

4 Conclusions and Future Work

In this paper we describe in detail the persistent queue system deployed in one version of our unified stock exchange platform – Global Order Routing system. Similar to concept of journal in database management system, this persistent queue system holds incoming messages in an internal journal, flush it to file system while updating the checkpoint; meanwhile, it also supports the functionality to purge a specified message

in the queue. Simulation study shows that averagely throughput of the queue system can be 5 -10% better than proprietary products.

This is the first step to implement a unified stock exchange platform. In our future work, we plan to extend that platform to dynamically route orders to the best stock exchange to trade, while publish/subscribe market data in real-time.

References

1. Frank J. Fabozzi, Franco Modigliani: Capital Markets: institutions and instruments. 3rd ed. Prentice Hall, Upper Saddle River NJ (2002)
2. http://www.chinatelecom.com.cn/
3. SUN Microsystems: Java Message Service Specification, Version 1.1. SUN Microsystems, Palo Alto CA (April 12, 2002)
4. Dave Sill: The qmail Handbook. Apress, Berkeley CA (2001)
5. FIX Protocol specification, Version 4.3. http://www.fixprotocol.org (August, 2001)
6. Nicole Maestri: NYSE tells firms to guard against erroneous orders. Reuters, New York NY (December 18, 2002)
7. SUN Microsystems Web Site: http://www.sun.com.

An Efficient Algorithm for Computing Inverses in GF(2^m) Using Dual Bases

Hyeong Seon Yoo, Seok Ung Yoon, and Eui Sun Kim

Department of Computer Science, Inha University,
Incheon, 402-751, Korea
hsyoo@inha.ac.kr

Abstract. This paper propose a new multiplicative inverse algorithm for Galois field GF(2^n) whose elements are represented by optimal normal bases type II. The efficiency of the arithmetic algorithms depends on the basis and many foregoing papers use either polynomial or optimal normal basis. A normal basis element is always possible to rewrite canonical basis form. The proposed algorithm combines normal basis and canonical basis. It is shown that the suggested algorithm is suitable for implementation and reduces the computation time to 5–10 % of the normal basis algorithm.

1 Introduction

There has been an increasing attention in the design of fast arithmetic operations in the Galois field GF(2^n) which have applications in coding theory and cryptosystem. In these applications, efficient arithmetic algorithms and hardware structures are crucial factors for good performance. The efficiency of these operations is closely related in the way of presenting element bases. Polynomial basis and normal basis are the two presenting methods, and both approach have pros and cones. Basic properties of normal basis and multiplication with minimal coefficients in the finite field GF(2^n) are studied [1–3].

The most popular inversion algorithms are either based on Fermat's little theorem, or on the Euclidean algorithm or related ones [–6]. For extension fields GF(q^m), the Itoh and Tsujii inversion algorithm is very effective [4, 5]. De Win et al. studied software implementation for arithmetic operations in GF(2^n) [7]. Koç et al published papers about multiplication algorithms in finite field [8, 9]. There is also a paper addressing this issue [10]. And software impelmenting issue is covered in a book by Rosing [11].

In this paper we restrict to the extension fields that can be obtained by the optimal normal basis type II. Our idea in this paper is that if we combine the optimal normal basis and the shifted form of the canonical basis. The optimal normal basis can be very effective for squaring operations and the shifted form of the canonical basis can be easily adopted for multiplication process. The suggested inversion algorithm is shown to be efficient and easily implemented.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 994–999, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Arithmetic Operations in Optimal Normal Basis

The field GF(2^n) is viewed as an n-dimensional vector space defined over GF(2). And it is well known that a normal basis exists in every finite field [1]. A normal basis in GF(2^n) is given as equation (1).

$$M = \left\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{n-1}}\right\} \text{ for some } \beta \in GF(2^n) \quad (1)$$

An optimal normal basis of type II for this field is constructed using a normal element $\beta = \gamma + \gamma^{-1}$, where γ is a primitive $(2n+1)$th root of unity and $\gamma^i \neq 1$ for any $1 \leq i < 2n+1$. In this basis, squaring of an element is just a cyclic shift of the basis..

Using this normal element, the normal basis M can be expressed as equation (2).

$$M = \left\{\gamma + \gamma^{-1}, \gamma^2 + \gamma^{-2}, \gamma^{2^2} + \gamma^{-2^2}, \ldots, \gamma^{2^{(n-1)}} + \gamma^{-2^{(n-1)}}\right\} \quad (2)$$

It is known that a basis element of the form $\gamma^{2^i} + \gamma^{-2^i}$ for $i \in [1, m]$ can be written uniquely as $\gamma^j + \gamma^{-j}$ for $j \in [1, m]$ [8]. The normal basis M has an equivalent form N which is a shifted form of the canonical basis.

$$N = \left\{\gamma + \gamma^{-1}, \gamma^2 + \gamma^{-2}, \gamma^3 + \gamma^{-3}, \ldots, \gamma^n + \gamma^{-n}\right\} \quad (3)$$

Construction of basis N is done by a systematic transformation of M. If the exponent 2^i is bigger than n, change it by $2n+1-2^i$. The basis N can be simplified using the relation $\beta_i = \gamma^i + \gamma^{-i}$ and a number can be expressed as eq.(4). The shifted form has a good property over normal basis since it is possible to present index by a small number less than n. The two equivalent bases can be changed.

$$A = \sum_{i=1}^{n} a_i \beta_i, \quad a_i \in GF(2) \quad (4)$$

Let A and B be expresses in the canonical basis, then the product C can be written as equation (5).

$$\begin{aligned} C &= A \times B \\ &= \left(\sum_{i=1}^{n} a_i \beta_i\right) \times \left(\sum_{i=1}^{n} b_j \beta_j\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j \left(\gamma^{i-j} + \gamma^{-(i-j)}\right) + \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j \left(\gamma^{i+j} + \gamma^{-(i+j)}\right) \end{aligned} \quad (5)$$

In the above equation if the exponent $i+j$ is greater than n, it will be changed to $2n+1-(i+j)$ by using the identity $\gamma^{2n+1} = 1$.

One simple, inversion algorithm in optimal normal basis is using the Fermat's theorem and continuously factoring the exponent. Let's start with basic notation of inverse of a number A^{-1}.

$$A \cdot A^{-1} = 1 \tag{6}$$

Using the Fermat's theorem the inverse is expressed as (7).

$$A^{-1} = A^{2^n-2} \tag{7}$$

From the exponent, we get equation (8), [11].

$$-1 \equiv 2^n - 2 \mod n + 1 \tag{8}$$

The equation (8) can be factored as (9).

$$2^n - 2 = 2\left(2^{n-1} - 1\right) \tag{9}$$

Let's denote $r_0 = n - 1$. If r_0 is even, it can be factored as eq. (10).

$$2^{r_0} - 1 = \left(2^{r_0/2} - 1\right)\left(2^{r_0/2} + 1\right) \tag{10}$$

Since it is in the exponent, right side of eq.(7) means eq.(11).

$$\left(\left(A^{2^{r_0/2}-1}\right)^{2^{r_0/2}+1}\right)^2 \tag{11}$$

The above equation means a digit rotation and a multiplication. If it is odd, we can factor as eq. (12).

$$2^{r_0} - 1 = 2\left(2^{\frac{r_0-1}{2}} - 1\right)\left(2^{\frac{r_0-1}{2}} + 1\right) + 1 \tag{12}$$

Since the second-parenthesis term is easy to calculate, we concentrate on the first parenthesis in both equations (10) and (12). If we let the exponent $r_0/2$ or $(r_0-1)/2$ be r_1, we have the same problem as before. This factoring process continues till the $i-th$ exponent r_i in the first parenthesis equals 1. If we calculate elements in 155 bits, the number of factoring process will be 8. It needs 1 rotation and 1 multiplication for even exponent, and it doubles for odd exponent. So the total arithmetic operations are 11 rotations and 10 multiplications.

3 A New Dual Bases Algorithm

In this paper we suggest to combine the optimal normal basis and the shifted form of the canonical basis. The change of basis is easily done by table lookup process. The computation of inverse needs number of squarings and multiplications. Since two bases have superior properties in squarings and the canonical basis, it is suggested to change basis for two different operations.

The computation steps are followings, Algorithm 1. First check the exponent r_i, continue the process if it is greater than 1. Start with optimal normal basis type and compute a squaring process. Next step is to change the basis into the canonical basis

by table lookup, this process is carried out quickly. Compute the multiplications. If the exponent is odd, change the basis to the optimal type and follow the squaring-multiplication step.

Algorithm 1. A dual bases inversion procedure

```
   ON: optimal normal basis,
   CB: canonical basis
1. start with n − bit number A , ON type;
2. factor as eq. (10) or eq. (12);
3. if the exponent r_i > 1;
4.   compute squaring;
5.   change the basis to CB;
6.   compute multiplication;
7. .  change the basis to ON;
8.     if r_i is odd;
9.       compute squaring;
10.      change the basis to CB;
11.      compute multiplication;
12. .     change the basis to ON;
13.    endif;
14. endif;
```

4 An Example

It is common that the number of bits in elliptic curve cryptography is between 150 and 200. In this range we can construct 15 optimal normal bases and 10 cases belong to type II. We tested number of bits 173, 174, 179, 183, 186, 189 and 191 cases. And two cases of element are selected. For case 1, non-zero bit points are 0, 64 and 128. And the second case, there are 6 non-zero points such as 0, 32, 64, 96, 128 and 160.

Numerical results are obtained by a 800MHz Pentium machine. Table 1 shows for the 3 non-zero bit case, the new algorithm is faster than the conventional one in all number of bit ranges from 2.1 to 20.6 %. Table 2 is for 6 non-zero bit case and the difference between two algorithm decreases, but it shows the nearly the same pattern as in Table 1. The difference is about 2.3 ~ 17.7 %.

Table 1. Inversion results for $A = \beta^{2^{128}} + \beta^{2^{64}} + \beta^{2^0}$ (m sec).

No. of bits	Conventional algorithm	New algorithm	% decrease
173	1.85	1.75	5.4
174	1.99	1.58	20.6
179	1.88	1.84	2.1
183	2.11	1.77	16.1
186	2.11	1.85	12.3
189	2.18	1.91	12.4
191	2.40	1.93	19.6

Table 2. Inversion results for $A = \beta^{2^{160}} + \beta^{2^{128}} + \beta^{2^{96}} + \beta^{2^{64}} + \beta^{2^{32}} + \beta^{2^0}$ (m sec).

No. of bits	Conventional algorithm	New algorithm	% decrease
173	1.85	1.74	5.9
174	2.03	1.67	17.7
179	1.92	1.79	6.8
183	2.10	1.81	13.8
186	2.14	2.09	2.3
189	2.17	1.83	15.6
191	2.38	2.25	5.4

5 Conclusions

In this paper we suggested a new inversion algorithm combining two types of bases, the optimal normal basis and the canonical basis. The computation time for the new method decreased about 2.1 ~ 20.6 % less for 3 non-zero bit case. And it decreased about 2.3 ~ 17.7 % for 6 non-zero bit case. The difference between two algorithms are less for the 6 non-zero bit case, but it shows the same tendency. The suggested new algorithm is easy to implement and can be applied in any cryptosystems.

Acknowledgements. This research was supported by University IT Research Center Project.

References

1. R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press, Cambridge, 1986
2. J. Omura and J. Massey, "Computational methods and apparatus for finite field arithmetic," U.S. Patent No. 4587627, 1986
3. R. C. Mullin, I. M. Onyszchuk and S. A. Vanstone, "Optimal normal bases in GF (p^n)*, " Discrete Applied Mathematics, 22, 149–161, 1988
4. T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases," 78, 171–177, 1988
5. J. Juajardo and and C. Paar, "Itoh-Tsujii inversion in standard basis and its application in cryptography and codes," 25, 207–216, 2002
6. M. A. Hansen, M. Z. Wang and V. K. Bhargava, "A modified Massey-Omura parallel multiplier for a class of finite field," IEEE Trans. On Computers, 42, 10, 1278–1280, 1993
7. E. De Win, A. Bosselaers, S. Vandenberghe, P. D. Gersem and J. Vanderwalle, "A fast software implementation for arithmetic operations in GF(2^k)," ASIACRYPT '96, LNCS 1233, 65–76, 1996
8. C. K. Koç and T. Acar, "Montgomery multiplication in GF(2^k)," Design, Codes and Cryptography, 14, 1, 57–69, 1998
9. B. Sunar and C. K. Koç, "An efficient optimal normal basis type II multiplier," IEEE Trans. on Computers, 50, 1, 83–87, 2001
10. D. V. Bailey and C. Paar, "Optimal extension fields for fast arithmetic in public-key algorithms," CRYPTO '98, LNCS 1462, 472–485, 1998
11. M. Rosing, *Implementing elliptic curve cryptography*, Manning Publ. Co., Greenwich, CT, 1999

Object Oriented Parallel Programming Model on a Network of Workstations

B. Suresh and R. Nadarajan

Department of Mathematics and Computer Applications, PSG College of Technology, Peelamedu, Coimbatore-641 004, TamilNadu, India
{suresh_psgtech, nadarajan_psg}@yahoo.co.in

Abstract. Parallel computing in network of workstations is receiving lot of attentions from the research community. However, still it is faced with many practical difficulties. One of the major problems is the lack of high level programming model. Many programming languages provide an efficient system, but employ a relatively low level programming model and rather too complex to be used in network environment. The object oriented system paradigm in software engineering has wide acceptance and provides support for the construction of modular and reusable program components and it is attractive for the design of large and complex serial programs. To address the previous said problem, in this paper we present a flexible, easy to use, and an effective object oriented parallel programming model by amalgamating the object orientation paradigm and parallelism. It is developed on top of PVM. This model exhibits a very convenient and natural programming style and provides functionality similar to C++.

Keywords. Object oriented programming, Parallel programming, Object oriented parallel programming.

1 Introduction

The object oriented paradigm provides support for the construction of modular and easily reusable program components and has proven to be useful for the design of large and complex system. In object oriented programming a system is described as a collection of *objects.* An object is an integrated unit of data and procedures, called methods acting on such data. The data in an object are stored in variables whose contents can be changed by assignment statements, present in the methods. It means that the variables in an object are not accessible from outside. This mechanism is a clear protocol for the exchange of information between the different objects of a programming application.

The parallel programming adopts a style, which facilitates the decomposition of a problem in to several independent tasks or processes [15]. The above said aspects of

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1000–1010, 2003.
© Springer-Verlag Berlin Heidelberg 2003

object oriented paradigm are potentially suitable for the field of parallel programming by treating an object as an independent task. An object and its private data are good abstraction of a process and process's local data. Public methods activated by sending messages are similar to message passing mechanism in parallel computing [13]. That is why, *an object naturally becomes an autonomous programming unit or unit of execution in parallel programming environment*.

Many systems have been developed by researchers on object oriented parallel programming by extending C++ (*See Section 4: Related works*) and defined many new notations to include specific constructs to express the parallelism. These new notations are burden to the programmers in perspective of learning its functionalities and these systems normally fail to maintain conventional programming style.

In this paper, a new object oriented parallel programming model OP++ has been proposed. We focus more on this model to give more casualty and natural programming style as like C++ when compared to any existing system for parallel programming in network of workstations.

This paper is organized as follows: In Section 2, description about our programming model is given. In Section 3, the working principle of our model is discussed. In Section 4, some existing systems that are closely related to our model have been analyzed and in Section 5, an example application program for our model is presented.

2 Programming Model

OP++ is an object oriented parallel programming language based on C++[17]. The basic idea of OP++ is to view an object as an independent or individual process and its member functions are decomposed computational part of the problem to provide an efficient parallel execution.

2.1 The Process Class

The responsibilities assigned to the programmers are to write C++ class(es), which are decomposed part(s) to warrant parallel execution. This is accomplished by deriving the written C++ class(es) from the built-in class called "*Process*". This mechanism provides "is-a" relationship between the class *Process* (base class) and the user written class (derived class). The class *Process* is a general class whose behavior is inherited by the derived class. The instance of such a class is called "process object". The process object will become an autonomous execution unit in parallel environment. The process object can be used like any other C++ class instance.

When an object is instantiated to the class derived from the class *Process*, a new process is created and placed at any processing node, which has been added to the PVM [7, 5] network to implement the object's autonomous activities. This scenario will be held when the execution point reaches the constructor of the class *Process* at the time of process object creation. The spawned process's ID is maintained as a member data in the class *Process* for further communication to the corresponding

process while any member function is invoked at the derived class of the class *Process*. The spawned process consists of actual definition of the class(es) that is derived from the class *Process*. This is discussed in detail in *Section 3*. So, for every process object there exists a worker process in the network. Since there is no public methods in class *Process*, deriving the class from it may be public, private or protected.

2.2 Asynchronous Method Calling

Parallelism is achieved by calling the member functions of process objects in an asynchronous manner. The statement that invoked a member function can continue its execution in parallel with the invoked member function. (i.e. the caller does not wait for the method completion and continues its execution to the consecutive statements). When the result is needed from the invoked member function, the execution has to wait until the required results are available.

The member function can call asynchronously by calling it in *par* {} block. *par* {} is a key word that we extended from compositional C++ (CC++) [2]. In our system *par* {} block implies that the method invocation are executed in parallel.

```
1.  # include <parallel.h>
2.  class foo : Process
3.    { public:
4.        int function1 () {...}
5.        int function2 () {...}
6.    };
7.  void main () {
8.   foo ob1,ob2;// 2 processes are created
9.   int x,y;
10.  par {
11.         x = ob1.function1 ();
12.         y = ob2.function2 ();
13.       }
14.   }
```

Fig 1.a. Example program

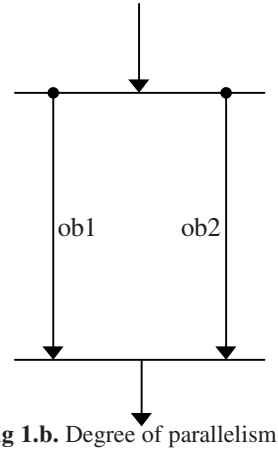


Fig 1.b. Degree of parallelism in *par* { } block

The given program (Fig 1.a) works as follows: when object *ob1* and *ob2* are instantiated, two processes will be created on processing node in network. When the line 11 calls the member function *function1( )* that belongs to the process object *ob1*, it never waits for the return value. It continues the execution and calls another member function of another process object *ob2* in line 12. The value of these two member functions is received at the end of the *par* {} block. The receiving point or synchronous point is found out depending upon the data dependency that exists in the given code inside the *par* {} block. This is discussed in detail in *Section 2.3* with an example.

The member function belongs to different process object can be invoked and executed on different nodes in parallel. As shown in Fig 1.b, *ob1.function1 ( )* and *ob2.function2 ( )* are executed in parallel. The only thing is no more than one member function of the given process objects will be executed at the same time. When more than one member function is called on same process object *in par* {} block it is executed in a sequential manner. The statements that are not available in *par* {} block will be executed sequentially.

2.3 Synchronization Point

Since the method in side the *par* {} block are invoked asynchronously, synchronization between the statements is required. The synchronization mechanism defined in our model can be described as follows. Methods invoked prior to a synchronization point can be executed in parallel with the calling methods, but only after the invoked methods have returned can the statements proceed from synchronization point.

Consider the process objects *ob1, ob2* and *ob3*. (i.e., the instance of the class derived from the class *Process*). The dependency between the statements in *par* {} block is observed at compile time and it produces suitable code to maximize the degree of parallelism. It is assumed that the methods used in the given example return some result, so that the dependency can be explained in more details.

```
par {
1.    x = ob1.fun (10,10);
2.    a = b + c;
3.    y = ob2.fun (10,a);
4.    z = ob3.fun (15,10);
5.    w = ob1.fun (1,2);
6.    c=y + z;
    }
```

Fig 2.a Example code

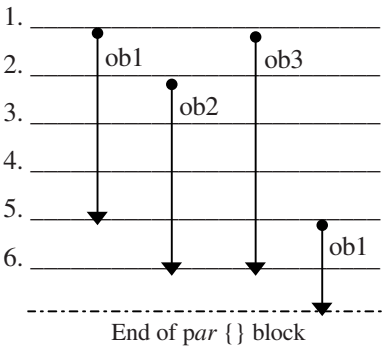


Fig 2.b Degree of parallelism

In Fig 2.a, the methods in line 1 and line 4 are invoked because both methods were called on different process objects and there doesn't exist any dependency between these statements. The method in line 3 invoked after the execution of line 2 because the method uses the variable "a" that was defined by line 2.

The value is received from the method of *ob1* before the line 5 since no more than one member function will run in parallel. Now the line 5 is executed. The value is received from the method of *ob2* and *ob3* before the line 6 and the value is received from *ob1* at end of the *par* {} block (Fig 2.b). The statements that are not associated with method invocation will not run in parallel with each other (i.e line 2 & line 6). And also if there is any statement like ,

x = ob3.fun1 (ob1.fun (10, 10), ob2.fun (15, 25))

in *par* {} block, the method of different process objects *ob1* and *ob2* in object *ob3* are invoked and executed in parallel. After the execution of the method of *ob1* and *ob2* the result is forwarded to the method of *ob3*.

2.4 Lifetime of a Process in a Network

The lifetime of a process that spawned at the time of creation of a process object is based on lifetime of the process object. When an object is created the process is created and placed in the processing node of the network. When the scope of the process object goes out and its destructor is reached, it kills the corresponding process from the network. The users need not to bother about maintaining the running processes.

2.5 Nested Parallelism

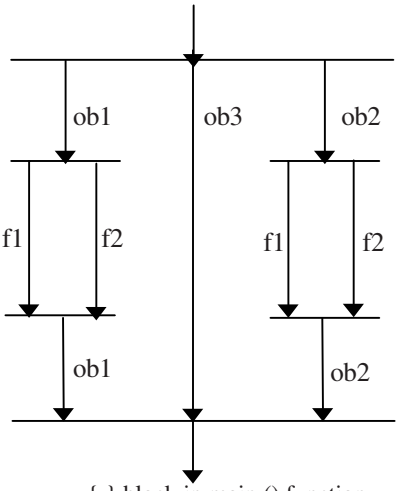
The fundamental construct of our model is based on master/worker process model. The process that runs main () function is said to be a master process. Other processes spawned at the time of process object creation are said to be worker processes. The worker process, which itself is capable of creating some other worker process(es) and so on. This kind of parallel execution is referred as a nested parallelism [11]. The nested parallelism is accomplished by containment [17]. The containment to achieve the nested parallelism is the inclusion of instance of a class derived from the class *Process* as a member data in its private area of another class derived from the class *Process*. The given example shows the nested parallelism by means of containment.

```
class foo : Process
 { public:
     int fun1 ( )
       { ....
         Statements
         ....
       }
     int fun2 ( )
       { ....
         Statements
         ....
       }
 }; // End of the class
```

```
class bar : Process
 {
  foo f1,f2; //creates 2 processes
  int x,y;
  public:
    int function ()
      {
        par{
            x = f1.fun1 ();
            y = f2.fun2 ();
           }
      }
 }; // End of the class
```

The bar class consists of two process objects that are instance of foo class. The method in a bar class calls two methods of process objects f1 & f2 in *par* {} block to run it in parallel. When an object (instance of bar class) invokes this method in some other par block, this gives nested parallelism.

```
void main ( )
  {
   bar ob1,ob2; // creates 2 processes
   foo ob3;   // creates 1 process
   int x,y,z;
   par {
        x = ob1.function ();
        y = ob3.fun1 ();
        z = ob2.function ();
       }
  } // End of the main
```



par { } block in main () function

2.6 Parallel Arithmetic Expression

Many scientific problems contain many arithmetic expressions, which take much time to evaluate. Normally most of the arithmetic expressions may have some sub expression which doesn't affect the result of the given expression if it is evaluated in parallel. That is the result of some sub expression is not required at time of some other expression evaluation. Such type of sub expressions can be observed and safely executed in parallel. Our system provides this facility to evaluate the expression in parallel, when the operands of the expression is a process object and the operator followed by that process object in the expression are overloaded as a public method.

To illustrate the parallel expression consider the process objects *a, b, c* and *d*. The operators succeeded to it are overloaded operators and the operands followed by the operator are parameters to the operator function.

$$\text{par } \{ \text{ x = ((} a + 2 \text{) * (} c \text{ / f)) + (} d \text{ * 10) + (} b \text{ / f) } \}$$

Step 1 : (a + 2) , (c / f), (d * 10), (b + f) are computed simultaneously.
Step 2 : Result of (a + 2) * Result of (c / f) is computed
Step 3 : Result of step 2 + Result of (d * 10) is computed
Step 4 : Result of step 3 + Result of (b / f) is computed

In the above expression when (a + 2) is evaluated the result of other sub expression like (c / f), (d * 10) and (b / f) are not required. So these sub expressions are evaluated in parallel. That is the methods +, /, *, / of the process objects *a, b, c* and *d* respectively are invoked asynchronously.

The synchronization points of the invoked methods (operator function) in an arithmetic expression strictly depend on the precedence of the operator at the

evaluation. When the above expression is evaluated in serial, it requires seven steps, but in parallel, it is evaluated in four steps. But this gives good computational performance when large amount of computation (coarse grained) is performed in operator overloaded method.

2.7 Inheritance and Virtual Functions

The concepts of inheritance of object orientation raised the reusability and extendibility of existing code over other programming styles. Basically our system is based on single inheritance to provide parallel processing by derived a class from the class *Process*. The inheritance can continue to a class from the class derived by the class *Process*. OP++ also supports virtual function to give the polymorphism to the public method in conjunction with dynamic binding.

Our system supports both multi level inheritance and multiple inheritance. While using multiple inheritance, the only rule to follow is the parents of the class that emerge in multiple inheritance must derived virtually from the class *Process*. The reason is when any object is created to the class derived from the class *Process*, a unique process ID is created and maintained in a private data of the class *Process*. To avoid the ambiguity of the process ID in class *Process*, when any class is derived from more than a class that is derived from the class *Process*, we need to maintain a single copy of a class *Process* by means of virtual base class.

3 Principle of OP++

A program written in OP++ is passed through the C++ preprocessor to extract the definition about the classes that are derived from the class *Process*. From the extracted information, the system creates a file whose name is *filename_wrk.cpp* by removing the key word *par* { } and adding appropriate calls that are related to PVM library functions. The worker file contains the actual code of the written class, which is derived from the class *Process*. The native C++ compiler compiles this file to generate an executable file (worker program) and it is spawned in the processing nodes that added to the PVM network at the time of object creation. The file *filename.cpp* is regenerated by including necessary library functions that provide an asynchronous message passing between the worker programs and compiled to produce executable file (master program). When process object is created, it spawns the worker process and dispatches the message to it.

The message is packed of class index, function index and list of argument values. The spawned process, receipt and unpack the message and performs the desired operation, which is either the creation of a new object or a method invocation based on the value of class index and function index. After the execution of the method in the worker process the result is stored in the buffer. The master program collects the result when it needs. The indices, which are generated by this system, are unique values of every class and its public method(s) that are derived from the class *Process*. The node that runs the master program is said to be master node and other is worker node.

4 Related Works

Investigations in amalgamating the concept of object orientation and parallel programming have already been made for several years. The classification of merging these concepts is described in [14] i.e., Parallelism in object orientation and Object orientation in parallelism. Based on these several languages are compared with respect to what objects stand for. Object may consider as process, shared passive abstract data type or encapsulation of multiple process and data.

In this section, we highlight the contribution made already in connection with our research work. We *stress* the criteria of object orientation in parallelism, casual programming style, flexibility and modularity. As we are interested in message passing technique, we have not considered the systems such as pC++ [6], CC++ [2], Distributed Processing in C++ (DPC++) [16] and DoPVM [10] which are based on distributed shared memory (DSM), which employs some complex concepts like mutex, semaphores, deadlock and critical sections. Even though the message passing technique does not involve such complex concepts, it raises the programming complexity to manage the synchronization points. So we focused on to provide an object oriented programming model based on message passing technique.

Many object oriented parallel programming systems have been developed based on DSM. But very few systems were developed using message passing. Such systems can be categorized in two ways. The first one is the system which provides object oriented library functions that supports message passing like PVM++[3], Para++[4], CPPvm[8]. These systems do not support object orientation in perspective of programming methodology and these systems are good at system and not at application level. The second one is providing object orientation in programming methodology like Dome [1] and Mentat [9, 12] that are closely relevant to our work.

Dome is based on C++, and supports only data parallelism. It is related to our work as it is run on top of the PVM. Dome supports only SPMD (single program, multiple data) programming model. It provides a library of classes to express parallelism. When an object is instantiated to the tailor made classes the data inside the object is partitioned among the processing node of the PVM network that performs the same code on different data. There are several differences between our approach and that of Dome. Our system supports both SPMD and MPMD (multiple programs, multiple data) programming model often with in the same program. The Dome is fully constructed of tailor made classes. And also, the problem having functional parallelism cannot be implemented in Dome.

Mentat is a noteworthy portable C++ based language. The philosophy followed in our programming model and Mentat looks related to each other. But, the principle and the execution strategy are totally different from one another. One of the major concepts is the mechanism of constructing dependency graph. In our system the dependency is observed based on process object and not depends upon the method invocation or the value of the parameters of the method of the process object. But in Mentat the dependency graph is constructed based on the value passed to the method and not by Mentat object. However, the object creation and method invocation, which makes the cost of a call unclear to the programmer. That is, if Mentat object is

instantiated, it is not the instance itself. The new object has been created for every method invocation on that object. If N times the method is invoked in Mentat object it will create N instance of that class which may execute in parallel.

In Mentat, definition of each Mentat class must be written in a separate file and each file must be compiled separately. Even the Mentat support inheritance, the compilation order is restricted. The base class must be compiled before the derived class. This gives burden to the programmer to maintain the inheritance hierarchies at the time of compilation. But in our system, programmer can write the program in a single file that consists of any number of classes derived from the class *Process*. In Mentat, many new notations are introduced for the data types that cast down the causality of the programming style. One of the highlight of our system is, it allows to overload the arithmetic operators like +, -, \, * for parallel arithmetic expressions, function overloading, constructor overloading and virtual functions. Our system doesn't support friend functions, which access the private data of the object.

5 Example Program in OP++

In the given example code, the class MathProcess consists of two member functions that sum up and find the maximum value of the given integer array and return the results and the class is derived from the class *Process.* There are two arrays of process objects *ob1* and *ob2* are instantiated. The idea behind this is to split the problem array into 5 sub arrays and it is passed to each process.

```
# include <iostream.h>
# include <parallel.h>
# define SIZE 5000
# define NPROCS 5

class MathProcess : Process
{  public :
      int summation ( int num, int *data)
          { int result=0;
             for (int i=0; i < num; i++)
                 result+=data[i];
             return result; }

      int maximum ( int num, int *data)
          { int maxno=data[0];
            for (int i=0; i < num; i++)
                 if (maxno < data[i]) maxno = data[i];
              return maxno; }
 }; //End of the class

void main () {
  MathProcess ob1[NPROCS],ob2[NPROCS]; // 10 processes
  int *data,temp1[4],temp2[4],result1=0,result2=0;
  data = (int*) malloc (sizeof (int) * SIZE);
```

```
  // fill the problem array  ( test data)
  for (int i=0; i<SIZE; i++) data[i] = i % 25;

  par
   { for (i = 0; i < NPROCS; i++) {
       temp1[i]=ob1[i].summation(1000,&data[i * 1000]);
       temp2[i]=ob2[i].maximum (1000, &data[i * 1000]);}
   } // End of par

  result2 = temp2[0];
  for (i=0; i < NPROCS; i++)
     { result1+ = temp1[i];
       if (result2 < temp2[i]) result2 = temp2[i];}

   cout << "Summation " <<result1<<endl;
   cout << "Maximum Number " <<result2<<endl;
} // End of main ()
```

The process objects in the array *ob1* performs addition operation and the array *ob2* performs finding maximum number in an array passed to it. In *par* {} the method of process objects are invoked asynchronously by passing the array as a parameter. In method calling, the first parameter represents the number of items from the address specified followed by it or in other words the value before the address parameter specify the number of items from the given address. The array is passed by value. The copy of specified items is moved to the process. The worker process performed the operation and the result is received at the end of the *par* {} block and the intermediate result is calculated and displayed. The worker process is killed from the network when the destructor of the process object is reached.

6 Conclusion

In this article, we have described the object oriented parallel programming in OP++ based on C++ that integrates the concepts of object orientation and parallel programming. We have shown the program written in OP++ that gives a casual programming style. The major problematic point in object oriented parallel programming is providing inheritance. Often many systems compromised in most important area like single and multiple inheritance, operator overloading and constructor overloading. But we designed our system to use all the major concepts of object oriented programming in parallel programming and also we concentrate more on minimizing the insertion of new notation that look odd. We have implemented OP++ on network of workstations that runs on Microsoft's Windows 95/98/NT and UNIX and programmed varieties of application programs. Our experience in writing programs in OP++ shows the benefits of object orientation that provides flexibility, modularity and casual programming style.

Acknowledgement.
We wish to thank V. Giraga Durai for his support during this research to led this paper.

Reference

1. Adam Begulian, Erik Seligam, Michael strankey. *DOME : Distributed Object Migration Environment.* School of Computer Science, Carnegie Mellon University. Technical Report– CMU–CS–94–153.
2. K. Mani Chandy, Carl Kesselman. *CC++:A Declarative Concurrent Object Oriented Programming Notation.* A Technical Report. California Institute of Technology. March 12, 1993.
3. Daniel Cohen Laroque. *PVM++*. Institute of Computer Design & Fault Tolerance. University of Karlsrue, September 30, 1994.
4. Oliver Coulaud, Eric Dillon. *Para++ : C++ Bindings for Message Passing Libraries – User Guide*.March 1997. http:// www.loria.fr/para++.
5. Markus Fischer, Jack Dangarra. *Experience with Windows 95/NT as a Cluster Computing Platform for Parallel Computing.* Journal of Parallel and Distributed Computing Practice. 1999, Vol 2Nr2.
6. Dennis Gannon, Shelby X. Yang, Peter Beckman. *User Guide for a Portable Parallel C++. Programming System, pC++*. Deptartment of Computer Science and CICA, Indiana University, Bloominton, Indiana, USA, September 9, 1994.
7. G.A Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderam.*PVM : Parallel Virtual Machine. A User Guide and Tutorial for Networked Parallel Computing.* MIT Press, 1994.
8. Steffen Gorzig, *CPPVM : C++ interface to PVM.* University of Stuttgart, Institute of Parallel and Distributed High Performance Systems.
9. A.S. Grimshaw. *Introduction to Parallel Object Oriented Programming with Mentat.* Department of Computer Science. University of Virginia. Report No TR-91-07, April 4, 1991.
10. C.L. Hartley and V.S. Sunderam. *Programming with Shared Objects in Networked Environment*, International Parallel Processing Symposium, April 1993.
11. Shih-Chen Huang. *Supporting Flexible Programming Model on a Network of Workstations*. PhD dissertation. New York University. January 2000.
12. *Mentat 3.0 User Manual.* The Mentat research group. Department of Computer Science, University of Virginia. http://www.cs.virginia.edu/~mentat
13. Janusz Niemiec. *CC++, pC++, Charm++ and Orca: Languages for Parallel Programming*. CIS Department, Syracuse University, December 22, 1993.
14. A.A. Radenski. *Object Oriented Programming & Parallelism.* Information Science Vol. 93, Issue 1-2, North Holland. Department of Computer Science. Winston Salem University. 1996.
15. Seyed H Roosta. *Parallel Processing and Parallel Algorithms.* Springer Publications. 1999.
16. Adre Silvetia, Rafael Avila, Marcos Barreto, Philippe Navausx. *DPC++: Object Oriented Programming Applied to Cluster Computing.* Institute of Informatics, Federal University of Rio Grande do Sul, Porto Algere, Brazil. First International Workshop of Project SNOW, November 23, 2001.
17. B. Stroustrup. *The C++ Programming Language*, Addison-Wesley, 1991.

Paroxysmal Atrial Fibrillation Prediction Application Using Genetic Algorithms

Sonia Mota[1], Eduardo Ros[1], Francisco de Toro[2], and Julio Ortega[1]

[1] Departamento de Arquitectura y Tecnología de Computadores,
Universidad de Granada, Spain,
{sonia,eduardo,julio}@atc.ugr.es
[2] Departamento de Ingeniería Electrónica, Sistemas Informáticos y Automática,
Universidad de Huelva, Spain,
ftoro@uhu.es

Abstract. Paroxysmal Atrial Fibrillation (PAF) prediction viability is a line of research currently being investigated. The definition of new valid parameters for this task may generate various heterogeneous features. Genetic Algorithms (GAs) automatically find a set of parameters to maximize the diagnosis capabilities of a scheme based on the K-nearest neighbours algorithm. This is an efficient way of generating a number of possible solutions for the problem of PAF prediction. The present paper illustrates how GAs, rather than a statistical study of the database can be used to select the parameters giving the best classification rates.

1 Introduction

Atrial Fibrillation is the heart arrhythmia that most frequently causes embolic events, 75% of which generate cerebrovascular accidents [1, 2]. The automatic diagnosis of patients that suffer PAF episodes by the analysis of ECG registers that do not contain explicit PAF traces is a difficult task. Different authors have studied methods for PAF prediction based on different parameters of ECG traces [3, 4, 5, 6, 7] but none of them have obtained definitive results, and thus the problem remains open.

An international research effort has recently been made to study the viability of an automatic diagnosis algorithm to predict Paroxysmal Atrial Fibrillation; this concluded that such a solution is possible, with acceptable efficiency [8, 9]. An automatic algorithm that could identify individuals with PAF characteristics is clinically important because it would motivate more specific and complex diagnostic tests.

The discrimination power of the parameters is measured through the *Classification rate* (see equation (1) in Appendix). The problem of maximising this equation by weighting the parameters is a multimodal optimisation problem in the sense defined in [10], because it is desirable to determine several optima solutions rather than a single one. Solutions based on different parameters could be useful in cases of patients suffering known cardiac arrhythmias that would invalidate parameters such as heart rate, PR distance, etc. Therefore we would choose a classification scheme for

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1011–1019, 2003.
© Springer-Verlag Berlin Heidelberg 2003

PAF diagnosis based on a set of valid parameters (i.e. not corrupted by current cardiac arrhythmia).

Evolutionary algorithms are meta-heuristics based on natural selection, and have been applied successfully to a wide range of optimization problems. A detailed description of evolutionary computation may be found in [11]. In the present work, a sequential binarised version of the genetic algorithm described in [10], called AGEMM, has been used for the above multimodal optimisation problem.

The present paper describes the application in Section 2 and a modular classification algorithm in Section 3. The following section focuses on how the classification process can be described in a genetic manner and optimised by means of AGEMM. Finally section 5 presents a summary of the main conclusions.

2 Database Description and Problem Definition

A public database for PAF prediction evaluation is available [12] provided by Physiobank [9]. It is composed of the ECG registers of 25 healthy individuals (n files) and 25 patients diagnosed with PAF (p files). It is important to note that none of these files explicitly contain any PAF episode, and therefore the diagnosis algorithms proposed using this database will focus on ECG characteristics present in sinusoidal mode (normal heart state). While there are 50 n files for the 25 healthy subjects (2 for each of them), the 50 p files, corresponding to PAF patients, can be separated into two groups:

- 25 ECG registers (one for each individual) not previous to PAF episodes, which means that 45 minutes before and after the recorded ECG traces are free of PAF episodes.
- 25 ECG registers (one for each individual) immediately previous to a PAF episode.

The main topic addressed in the present study is the implementation of an algorithm for the automatic diagnosis of PAF patients, based on ECG traces in which PAF episodes do not appear explicitly. This means that the diagnostic capabilities of the algorithm do not depend on the detection of PAF episodes. This would make it possible to diagnose such a pathology in preventive medical examinations. With the available database two different topics can be addressed:

- PAF diagnosis based on ECG traces with no PAF episode. This application attempts to discriminate the registers of PAF patients among the whole database.
- PAF episode onset prediction. The aim of this topic is to focus on the files of PAF patients and to distinguish between the registers previous to PAF episodes and all others.

3 Modular Classification Algorithm

A low level processing algorithm was used to extract 48 parameters (RR rate, PR distance, P-wave width, P-wave integral, etc) that we considered important to characterise an ECG trace for PAF diagnosis [13]. In this way, each ECG file was translated to a 48-component vector (p_1, p_2, … , p_{48}). Each parameter represented a different physical magnitude of different range. For this reason, all parameters were typified: the average (M_i) of each parameter was calculated within the whole database and then each parameter was divided by its corresponding M_i. In this way the new vector components are adimensional and have a similar range; thus they can be compared in a multiparametric classification scheme.

A modular classification algorithm for this application based on the K-nearest neighbours has been described in [14]. The labelled vectors are the kernel of the classification algorithm. For each new non-labelled vector, the Euclidean distances to the labelled vectors are calculated. The label of the K-nearest neighbours is consulted and the final calculated label is the same as that of most of the K-neighbours. All the results shown in the following sections have been obtained with a single neighbour in the labelling step. The modular property of this algorithm makes it easy to consider a different number of parameters without changing the classification scheme. Therefore, an automatic optimisation method can be applied to search subsets of parameters that maximize the classification performance.

Some parameters have more discrimination power than others, and the algorithm must focus on some of them to obtain representative distance differences between PAF and healthy patterns.

In previous works [13, 14] the parameters were selected by means of a statistical study. In the approach proposed in the present study, the use of Genetic Algorithms avoids this data analysis and leads to diverse solutions based on different parameters.

There is a test database but it is composed of another 100 (non-labelled) registers that have significant statistical differences from the labelled ones used for the present study. Furthermore, to avoid label-guessing requests, the access to results obtained with test files was very restricted. To validate the test database, we processed the training and test registers in the same manner, extracting the same characteristic parameters. The classification algorithm described above was configured to discriminate the training and test vectors, obtaining classification results around 92%. This meant that the test files could not be used as benchmarks for classification schemes based on these parameters.

4 Performance Optimisation through AGEMM

Evolutionary algorithms use different selection mechanisms and new (candidate) solution generation by means of transforming old solutions. Typical transformation operators include mutation (random changes in a solution) and crossover (also called recombination) between any solutions in the *population* (set of candidate solutions). The quality of any solution is evaluated by means of a fitness function. This sort of cooperative interaction provides better performance than the classic search method [15]. However, evolutionary algorithms, in particular when using elitist selection,

tend to converge to one single optimum in the search space. In that sense, some strategies to maintain diversity in a population may be required in multimodal optimisation problems where more than one optimum must be obtained. Niching methods [16], inspired by the behaviour of ecosystems with limited resources to share between individuals in the population, include some such strategies.

In this work , the GA used in [10], has been applied for our purpose. This algorithm performs a two-level niching technique that incorporates the benefits of an island model [17] and the niching technique known as deterministic crowding [18].

In order to be able to modify the weight of the different parameters in the classification scheme automatically, the input pattern is multiplied by a weight vector (W), i.e. $I=(p_1, \ldots, p_{48})\cdot(w_1, \ldots, w_{48})$, by which the weights W (0 or 1) activate or inhibit a their corresponding parameter. These weight vectors represent the chromosome of the different solutions optimised by AGEMM to maximise the classification rate (fitness function) (equation 1 in Appendix). The AGEMM was used with a population size of 10000, a mutation rate of 0.1 and 100 generations. We have obtained interesting results for the different topics mentioned in section 2.

4.1 PAF Diagnosis Based on Two Types of ECG Traces

PAF diagnosis based on two types of ECG traces: those immediately previous to PAF episodes and those distant from PAF episodes.

For this approach, the P vector corresponding to a patient is calculated by adding the parameter vector extracted from the ECG trace previous to a PAF episode and the parameter vector extracted from the ECG trace not previous to a PAF episode. The GA generates a population of solutions characterised by their W vectors.

The GA obtains 1182 solutions with classification performance levels above 70% representing 11.82% of the total population size. For a more detailed study, we concentrate on the four betst solutions, those with a classification performance above 80%; their characteristics (see Appendix) are summarized in Table 1.

To illustrate that the four solutions are functionally different, we calculate the Hamming distance between them as the number of *not-common-components*: $d_{12}=16$, $d_{13}=21$, $d_{14}=23$, $d_{23}=27$, $d_{24}=21$ and $d_{34}=20$. The distance represents the difference between the solutions; thus, for example, solutions 1 and 3 together are based on 43 features and only 22 of them are shared because the distance is 21. This fact can also be shown by counting the number of *active-shared-components*, as shown in the histogram in Fig. 1.

4.2 PAF Diagnosis Based on ECG Traces Immediately Previous to a PAF Episode

The GA obtains 111 solutions with a classification performance above 70%, which represents 1.11% of the total population size. The four best solutions are summarised in Table 2.

Table 1. Best results of PAF diagnosis based on the addition for the parameters extracted of ECG traces far from PAF episodes and parameters extracted from ECG traces immediately previous to PAF episodes.

Solution Nº	Chromosome showing the active features
1	00001011000100010110000110100101100000000011111111
2	00001010100110101011010110010101100000011010011
3	01100101101001010100000110010001111101010110101 0
4	10101101111011110010110010010000101001000011 0111

Solution Nº	Classification Performance (%)	Sensibility (%)	Specificity (%)	Number of active features
1	84	92	76	20
2	82	88	76	22
3	82	88	76	23
4	80	84	76	25

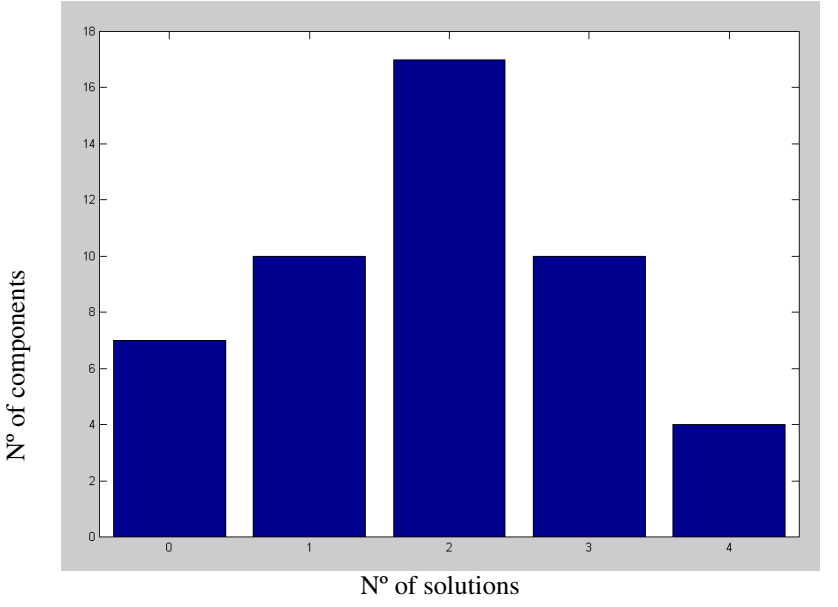


Fig. 1. Histogram that illustrates the frequency of shared components. Only 4 parameters are shared by the four solutions, 7 parameters are not used by any solution and 17 parameters are shared by two solutions.

Table 2. Best results of PAF diagnosis based on ECG traces immediately previous to a PAF episode.

Solution Nº	Chromosome showing the active features
1	00000001100010000111001001000110011110010011001
2	10000101011100001000000100011011000101000001111
3	00000000100011000010001000000011101111100000100
4	10110111111001011101001101111000000000000100010

Solution Nº	Classification Performance (%)	Sensibility (%)	Specificity (%)	Number of active features
1	78	84	72	19
2	78	80	76	19
3	78	80	76	14
4	76	80	76	22

The distances (number of not-common-components) between the different solutions are: d_{12}=26, d_{13}=17, d_{14}=27, d_{23}=23, d_{24}=21 and d_{34}=30. There is no parameter active in the four solutions and only 5 are active in three solutions.

4.3 PAF Diagnosis Based on ECG Traces not Previous to a PAF Episode

The GA obtains 879 solutions with a classification performance above 70% , which represents 8.79% of the whole population. Table 3 summarises the best four solutions.

Table 3. Best results of PAF diagnosis based on ECG traces not previous to a PAF episode.

Solution Nº	Chromosome showing the active features
1	00000111111100010001111111100111100101111010101001
2	01110111100100011111000101110010101101101010 1101
3	00010111000100011001011100011010000111000010 0010
4	00001011101001010011011100010111000011101100 1111

Solution Nº	Classification Performance (%)	Sensibility (%)	Specificity (%)	Number of active features
1	84	88	80	28
2	82	88	76	28
3	82	96	68	19
4	80	88	72	25

The distances between these solutions are: d_{12}=18, d_{13}=21, d_{14}=19, d_{23}=19, d_{24}=23 and d_{34}=20. Seven components are active in the four solutions, 12 components are active in three solutions, another 12 are active in two solutions, and 12 parameters are exclusively used in a single solution.

4.4 PAF Episode Onset Prediction

The GA obtains 640 solutions with classification performance levels above 70%, which represents 6.40% of the whole population. Table 4 summarises the best four solutions.

Table 4. Best results for PAF episode onset prediction.

Solution Nº	Chromosome showing the active features			
1	1101110110000001111110010011111111110011111010100			
2	1111100001111001101010101000011000000011111010101			
3	0101111111000011111111011100100110100111110000101			
4	1000000111001001101111101110110000011111010011001			
Solution Nº	**Classification Performance (%)**	**Sensibility (%)**	**Specificity (%)**	**Number of active features**
1	80	76	84	30
2	78	76	80	24
3	78	80	76	29
4	78	80	76	26

The distances between the individual solutions are : d_{12}=22, d_{13}=17, d_{14}=24, d_{23}=25, d_{24}=20 and d_{34}=23. Six components are active in the four solutions, 16 are active in three solutions, 14 components are used in two solutions, 9 are used exclusively by a single solution and 3 features are not used in any solution.

5 Discussion

This paper addresses the application of PAF prediction, i.e. PAF diagnosis based on ECG traces when no PAF episode occurs. We studied different topics related to this application. In previous works [13, 14] we defined 48 parameters that can be extracted from ECG traces for this application. The task of selecting some of these to maximise the classification rate of a simple algorithm based on the K-nearest neighbours [19] is not a trivial one. In this paper, a GA is used for the task, generating solutions represented by a weight vector that can be used to filter the input parameter vector, in order to activate the different parameters. It is shown that the GA reaches classification rates of around 84% for diagnosis, generating diverse solutions based on different parameters, and eliminating the possible redundancy of the 48 parameters. These parameters can also be used for PAF episode prediction, with classification rates of up to 80%. The generation of a whole population of solutions with different characteristics is interesting, to select the ones that best fit the application aims. In this sense, because the ECG is non-invasive, this approach to PAF detection can be considered a first diagnostic test, and therefore it would be interesting to obtain also high values for sensibility. This characteristic enhances the capability to detect patients with PAF even when the classification rates for healthy subjects decrease. Therefore, the solution with the highest sensibility is preferable among solutions with the same classification performance levels.

It is also interesting to note that the classification algorithm is modular, and so the inclusion of new parameters defined by other authors is straightforward. When the number of parameters increases, the GA optimisation task takes a long time (with 48 parameters the simulations in a Pentium III 500 MHz took a few minutes). This would motivate a parallelisation of the GA optimisation task [10] so that it could be run on a cluster

References

1. Cabin H.S.; Clubb K.S.; Hall C.; Perlmutter R.A.; Feinstein A.R.: "Risk of systemic embolization of atrial fibrillation without mitral stenosis", Am. J. Cardiol., vol. 61, pp. 714–717, 1990.
2. Petersen P.; Godtfredsen J.: "Embolic complications in paroxysmal atrial fibrillation", Stroke, vol. 17, pp. 622–626, 1986.
3. Ishimoto N, Ito M, Kinoshita M. Signal-averaged P-wave abnormalities and atrial size in patients with and without idiopathic paroxysmal atrial fibrillation. Am Herat J, 2000; 139:684–689.
4. Amar D, Roistacher N, Zhang H, Baum MS, Ginsburg I, Steinberg JS. Signal-averaged P-wave duration does not predict atrial fibrillation after thoracic surgery. Anesthesiology, 1999; 91:16–23.
5. Vikman S, Makikallio TH, Yli-Mayry S, Pikkujamsa S, Koivisto AM, Reinikainen P, Airaksinen KEJ, Huikuri HV. Altered complexity and correlation properties of R-R interval dynamics before the spontaneous onset of paroxysmal atrial fibrillation. Circulation, 1999; 100:2079–2084.
6. Hnathova K, Waktare JEP, Murgatroyd FD, Guo X, Baiyan X, Camm AJ, Malik M. Analysis of the cardiac rhythm preceding episodes of paroxysmal atrial fibrillation. Am Heart J. 1998; 135:1010–1019.
7. Kolb C, Nurnberger S, Ndrepepa G, Schreieck J, Zrenner B, Karch M, Schmitt C, Modes of initiation of paroxysmal atrial fibrillation an analysis of 157 spontaneously occurring episodes using 12-lead Holter monitoring. PACE, 2000; 23(4):607.
8. http://www.cinc.org/LocalHost/CIC2001_1.htm
9. Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. PhysioBank, PhysioToolkit, and Physionet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101(23):e215–e220 http://circ.ahajournals.org/cgi/content/full/101/23/e215; 2000 (June 13).
10. F.deToro, A.F.Díaz, C.Gil, J.Ortega. *AGEMM: Optimización Multimodal Paralela con Algoritmos Genéticos*. Actas de las XII Jornadas de Paralelismo (6 páginas), Septiembre, Valencia, 2001.
11. Bäck, T.; Hammel, U.; Schwefel, H.-P.:"Evolutionary Computation: comments on the history and current state". IEEE Trans. on Evolutionary Computation, Vol.1, No.1, pp. 3–17. Abril, 1997.
12. http://physionet.cps.unizar.es/physiobank/database/afpdb/
13. Mota S, Ros-Vidal E, Díaz AF. Extracción de parámetros del electrocardiograma para el diagnóstico de fibrilación auricular paroxística. CASEIB 2001, pp. 133-137, Madrid, 29–30 November, 2001.
14. Ros-Vidal E, Mota S, Fernández FJ. Diseño de un algoritmo modular de predicción de Fibrilación Auricular Paroxística. CASEIB 2001, pp. 305-308, Madrid, 29–30, 2001
15. François, O.:"An evolutionary strategy for global minimization and its Markov chain analysis". IEEE Trans. on Evolutionary Computation, Vol.2, No.3, pp.77–90. Sep., 1998.

16. B.Sareni and L. Krähenbühl, "*Fitness Sharing and Niching Methods Revisited*". IEEE Transaction on Evolutionary Computation, Vol 2, No. 3, 1998.
17. Cantú-Paz, E.:"A survey or Parallel Gas". Informe Técnico IlliGAL R.97003, 1997.
18. S.W. Mahfoud, *"A Comparison of Parallel and Sequencial Niching Methods"*, Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, CA, 1995.
19. Devijver, PA y Kittler, JV. *Pattern Recognition. A Statistical Approach*, Prentice Hall-Englewood Cliffs 1982

Appendix

For biomedical diagnosis applications, the final diagnosis is that a patient is **ill** (suffering a certain pathology) or **healthy** (free from this particular pathology). This means that the classification result can be seen as one of the following cases:

True Positive (TP). The algorithm classifies the subject as ill and the subject is in fact ill.

True Negative (TN). The algorithm classifies the subject as healthy and the subject is in fact healthy.

False Positive (FP). The algorithm classifies the subject as ill but the subject is healthy.

False Negative (FN). The algorithm classifies the subject as healthy but the subject is ill.

With these cases, different functions of interest can be defined:

Classification rate:

$$C = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Sensibility: represents the ratio between the detected ill patients and the total ill patients.

$$SENSI = \frac{TP}{TP + FN} \tag{2}$$

Specificity: represents the ratio between the detected healthy subjects and the total healthy subjects.

$$SPECI = \frac{TN}{TN + FP} \tag{3}$$

A Metadata Tool for Retrieval from Heterogeneous Distributed XML Documents

Young-Kwang Nam[1], Joseph Goguen[2], and Guilian Wang[2]

[1] Dept. Computer Science, Yonsei University, Wonjoo, Republic of Korea
yknam@dragon.yonsei.ac.kr,
[2] Dept. of Computer Science and Engineering, UCSD, La Jolla, CA 92093
goguen, guilian@cs.ucsd.edu

Abstract. A metadata approach for retrieval from heterogeneous distributed XML documents is given. A prototype system uses distributed metadata to generate a GUI data integration tool, for describing mappings between master and local DTDs, by assigning index numbers and specifying conversion functions. A DDXMI (Distributed Document XML Metadata Interchange) file is generated based on the mappings, and used to translate queries over the virtual master document into sub-queries to local documents. Quilt is the XML query language. An experiment testing feasibility is reported using three bibliography documents. The system runs under NT, using Java servelets and JavaCC.

1 Introduction

As more and more information sources become available online, it is often required to retrieve information from multiple sources, e.g., in ecology, sociobiology, medicine, and e-commerce. Hence convenient access to multiple, heterogeneous information sources through an integration mechanism is very desirable. Another trend is towards semistructured data formats such as XML, to combine the advantages of structured and unstructured data by imposing some structure on free text. XML is an easily processed format for extracting information, without the rigidity of a relational database [7].

Our DDXMI approach (for Distributed Documents XML Metadata Interchange), builds on XMI [19]. The master DDXMI file includes the XML document name or location, XML path information, and semantic information about XML elements and attributes. A prototype system has been built that generates a tool for meta-users to do meta-data integration, producing a master DDXMI file, which is then used to generate queries to local documents from master queries, and to integrate the results. This tool parses local DTDs, generates a path for each element, and produces a convenient GUI. The mappings assign indices, which link local elements to corresponding master elements and to the names of conversion functions. These functions can be built-in or user-defined in Quilt [6], our XML query language. The DDXMI is then generated based on the mappings by collecting over index numbers. User queries are processed by Quilt according to the generated DDXMI, by generating an executable query for each relevant local document. With DDXMI, users can get all the answers within the

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1020–1029, 2003.
© Springer-Verlag Berlin Heidelberg 2003

master DTD scope by just writing a query for the virtual master DTD instead of for each document.

This system is relatively simple, since some of the most complex issues, such as distributed queries and query optimization, are handed off to Quilt, and it is easy to use, due to its simple GUI. The system is also flexible: users can get any virtual integrated information system based on the same data sources, and different users can have different virtual information systems for their own applications.

2 Related Work

Besides data warehousing, many solutions to heterogeneity of multiple distributed data sources have been developed, based on the mediator architecture [18]. In structural approaches, the mediation engineer's knowledge of application specific requirements and local data sources are a crucial but implicit input. Integration is obtained through a virtual integrated schema that characterizes the underlying data sources. On the other hand, semantic approaches assume that enough domain knowledge for integration is contained in the exported conceptual models, or ontologies, of each local data source. This requires a common ontology among the data source providers, and assumes that everything of importance is explicitly described in the ontologies; however, these assumptions are often violated in practice.

ROBIN [11, 15], Tsimmis [17], MedMaker [16], MIX [1], and IIT Mediator [7] are structural, mostly global-as-view approaches. According to the integrated view definition, at query time the mediator resolves the user query into subqueries to suitable wrappers that translate between the local languages, models and concepts, and the global concepts, and then integrates the information returned from the wrappers. In some other structural approaches, users are given a language or graphical interface to specify only the mappings between the global schema and local schemas, from which the system generates the view definition. In Information Manifold (IM) [12, 17], the description logic CARIN is used to specify local document contents and capabilities. The IM mediator is independent of applications, since queries over the global schema are rewritten to sub-queries over the local documents (defined as views over the global schema) using the same algorithm for different combinations of queries and sources. The most important advantage of local-as-view approaches is that an integrated system built this way easily handles dynamic environments.

Clio [10, 13] introduced an interactive schema-mapping paradigm in which users are released from the manual definition of integrated views in a different way from IM. A graphical user interface allows users to specify value correspondences, that is, how the value of an attribute in the target schema is computed from values of the attributes in the source schema. Based on the schema mapping, the view definition is computed using traditional DBMS optimization techniques. In addition, Clio has a mechanism allowing users to verify correctness of the generated view definition by checking example results. However, Clio trans-

forms data from a single legacy source to a new schema; it remains a challenge to employ this paradigm for virtual data integration of multiple distributed data sources. Xyleme [5] provides a mechanism for view definitions through path-to-path mappings in the form of a set of pairs (abstract path, concrete path) in its query language, assuming XML data. Our prototype differs from Xyleme in its query language independence, and in using a local-as-view mapping description, which is translated to global-as-view when generating the corresponding DDXMI file. Hence it combines the virtues of both global-as-view and local-as-view approaches. The present paper differs from our previous paper [14] in its focus on problems relevant to information retrieval and documents, and in describing new features of the implemented system to handle mappings that involve attributes, whereas previously we could only handle mappings among elements.

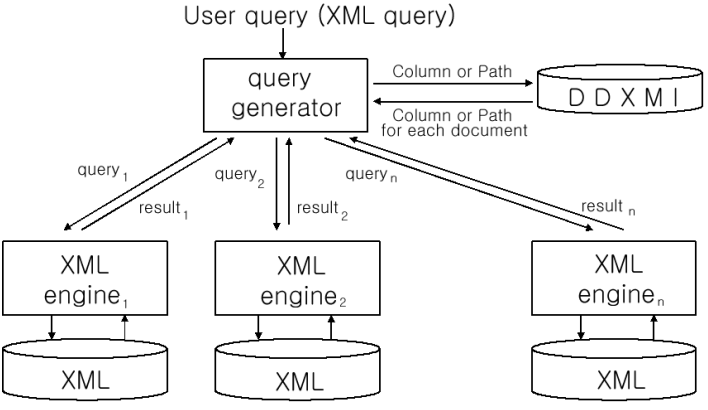


Fig. 1. DDXMI system structure for distributed documents

3 The Distributed Documents XML Interface (DDXMI)

An overview of the DDXMI system structure for retrieval from distributed documents is shown in Figure 1. We assume all documents are in XML, either directly or through wrapping. The basic idea is that a query to the distributed documents, called a master query, is automatically rewritten to sub-queries, called local queries, which fit each local document format using the information stored in the DDXMI by the query generator. The DDXMI contains the path information and functions to be applied to each local document, along with identification information such as author, date, comments, etc. The paths in a master query are parsed by the query generator and corresponding paths of each local document are substituted, if there are paths for the master query, by consulting the DDXMI. If not, a null query is generated for the corresponding path in the local query, meaning that this query cannot be applied to that document. The result collects all the answers from each document.

3.1 The Structure of DDXMI

The DDXMI is an XML document, containing meta-information about relationships of paths among documents, and function names for handling semantic and structural discrepancies. The DTD for DDXMI documents is shown in Figure 2. Elements in the master document DTD are called source elements, while corresponding elements in local document DTDs are called destination elements. When the query generator finds a source element name in a master query, if its corresponding destination element is not null, then paths in the query are replaced by paths to the destination elements to get a local query (we will see there can be more than one destination element). For example, different sites may represent the author field of the master document as 'author', 'authorname', 'name', 'auth' element, etc. in different local documents. Then the author source element in the DDXMI matches the destination element 'author', 'authorname', 'name', etc., as appropriate for each local document; more complex cases are discussed below.

```
<!ELEMENT DDXMI (DDXMI.header, DDXMI.isequivalent, documentspec)>
<!ELEMENT DDXMI.header (documentation,version,date,authorization)>
<!ELEMENT documentation (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT authorization (#PCDATA)>
<!ELEMENT DDXMI.isequivalent (source,destination*)*>
<!ELEMENT source (#PCDATA)>
<!ELEMENT destination (#PCDATA)>
<!ELEMENT documentspec (document, (elementname, shortdescription,
            longdescription, operation)*)*>
<!ELEMENT document (#PCDATA)>
<!ELEMENT elementname (#PCDATA)>
<!ELEMENT shortdescription (#PCDATA)>
<!ELEMENT longdescription (#PCDATA)>
<!ELEMENT operation (#PCDATA)>
```

Fig. 2. DTD for DDXMI

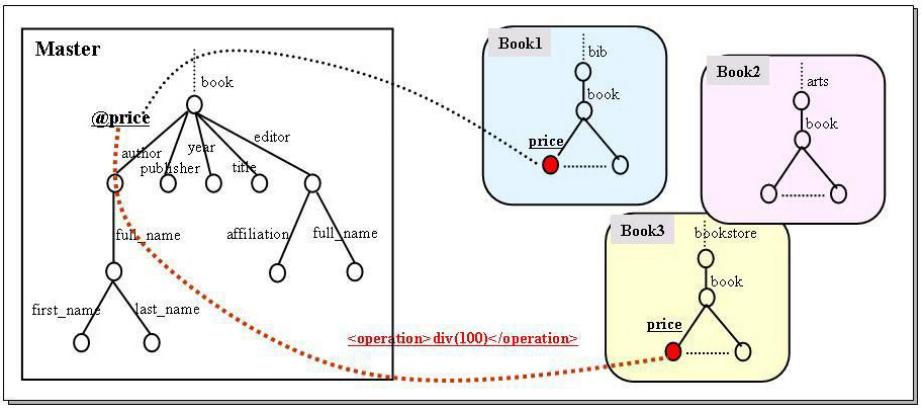
3.2 Mapping between elements or attributes

Mappings are classified as One-to-One, N-to-One, and One-to-N, according to the number of nodes (including both elements and attributes) involved in the master DTD and a local DTD. There are three different cases for One-to-One mapping: from an element to an element, from an element to an attribute, and from an attribute to an element. Mapping from an element to an element is skipped here, since it is simpler than mapping between an element and an attribute.

One-to-One, from element to attribute There are two cases when an element is mapped to an attribute. If the element is a terminal node, then the

element name is simply matched with the attribute name. Otherwise, the attribute name is removed from the path of the local document, since it does not contribute to the path name in the DTD tree.

One-to-One, from an attribute It is straightforward to handle an attribute in the master DTD, because an attribute is generally a leaf node of the master DTD tree. If it matches an attribute, just replace the name. But if an element is matched, that element will appear in the path name even if the attribute name is not in the master query. Figure 3 shows a mapping from a price attribute in the master query to a price element in a local document. The attribute name is replaced by the element name. Notice that, even for mappings between attributes and elements, there can be cases like those we classified in the previous section as 1:N and N:1, and there can also be semantic conversion functions.




```
<source>/book/@price</source>
    <destination>/bib/book/price</destination>
    <destination>/bookstore/book/price</destination>
```

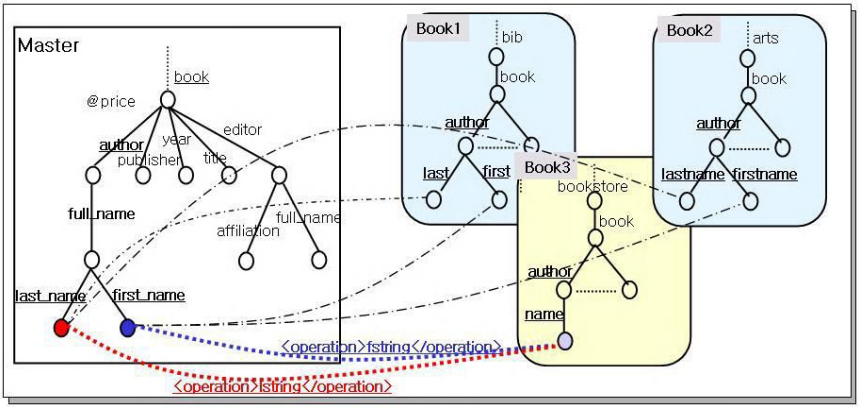
Fig. 3. One-to-One, attribute to element mapping

Conflicts can arise between representations. For example, the price field in the master DTD uses dollars, but the Book3.DTD in Figure 4 represents it in cents. Some mechanism is required to translate such representations. In this case, when the query is parsed, it is replaced by the result of applying the function `price/100` to convert to the dollar unit.

N-to-One, and One-to-N If two or more nodes of the master DTD correspond to one node in a local document, then the local DTD node will have more than one index number. For example, the first_name and last_name nodes in the master DTD tree in Figure 4 are mapped to the name node in the Book3 DTD tree. In this figure, only the Book3 DTD has full names; the others have separated first name and last name. Hence the separation function names, `fstring` and

`lstring`, are included in the DDXMI file for the full name node of the Book3 DTD.

The opposite case is where one node in the master DTD maps to several nodes in a local document. For example, the editor name in the master DTD may be represented with separate first and last names in a local document. The `con` function concatenates the first and the last name elements to get the full name.




```
<source>/book/author/full_name/first_name</source>
    <destination>/bookstore/book/author/name</destination>
<source>/book/author/full_name/last_name</source>
    <destination>/bookstore/book/author/name</destination>

<documentspec>
    <document>book.xml</document>
        <elementname>/book/author/full_name/first_name</elementname>
            <operation>lstring</operation>
        <elementname>/book/author/full_name/last_name</elementname>
            <operation>fstring</operation>
```

Fig. 4. N-to-One example

3.3 Replacing paths in a query

XML query languages have two kinds of paths, relative and absolute. A path name after '/' is absolute, and one after '//' is relative. A master element has a path name corresponding to local element names if they have the same index number. But some elements in the master DTD may have a longer path name in the local query. When assigning the index number, some nodes in the middle of the tree may be skipped without assigning a number. Then the node in that path will include the skipped node name in the path.

4 Query generation and execution examples

Assume we wish to build a library master document of DTD paths and three local document DTD trees as shown in Figure 5. The same index number is given to elements that have the same meaning, and each index sequence is different. Some nodes in the DTD tree have no number, since the master index does not include it. Book1.DTD and Book3.DTD have a price node but Book2.DTD doesn't. If some query includes the price element, then no query would be generated for Book2.XML since Book2.DTD doesn't have a price element. Based on the index assignment in Figure 5, the DDXMI file will be generated automatically by collecting paths with the same numbers. Using this file, users can modify or update the DDXMI file easily, or the file can be re-generated by reassigning the index numbers.

[A Master Library document DTD Paths]

0	Book.xml
1	/book
11	/book/@price
12	/book/author
1211	/book/author/full_name
1212	/book/author/full_name/first_name
13	/book/author/full_name/last_name
14	/book/title
15	/book/year
16	/book/publisher
161	/book/editor
162	/book/editor/affiliation
162	/book/editor/full_name

[Book1.Index]

0	Book1.xml
1	/bib/book
11	../price
12	../author
1211	../author/first
1212	../author/last
13	../title
14	../@year
15	../publisher
16	../editor
161	../editor/affiliation
162	../editor/last
162	../editor/first

[Book2.Index]

0	Book2.xml
1	/arts/book
12	../author
1211	../author/firstname
1212	../author/lastname
13	../subject
14	../@year
15	../publisher

[Book3.Index]

0	Book3.xml
1	/bookstore/book
11	../price div(100)
12	../author
1211	../name fstring
1212	../name lstring
13	../title

Fig. 5. Book1, Book2, Book3 indexed DTD trees, with master DTD

When a user enters a query name, our system generates the local queries. Figures 6 and 7 show examples of queries, their generated local queries, and the result of their execution, for the above three XML documents. Quilt is our XML query language, and Kweelt, developed at the University of Washington, is used to execute Quilt queries.

5 Conclusion and remaining issues

We have built a system for resolving both structural and semantic conflicts in a distributed document system (assuming each local document is in XML format, or the local sources accept XML queries and export XML data) using a GUI tool for generating a file called DDXMI that contains information about data structures and semantics. DTD trees are generated automatically, allowing a data integrator to assign the same index number to nodes with the same meaning and to name semantic functions to resolve representation differences. Then the

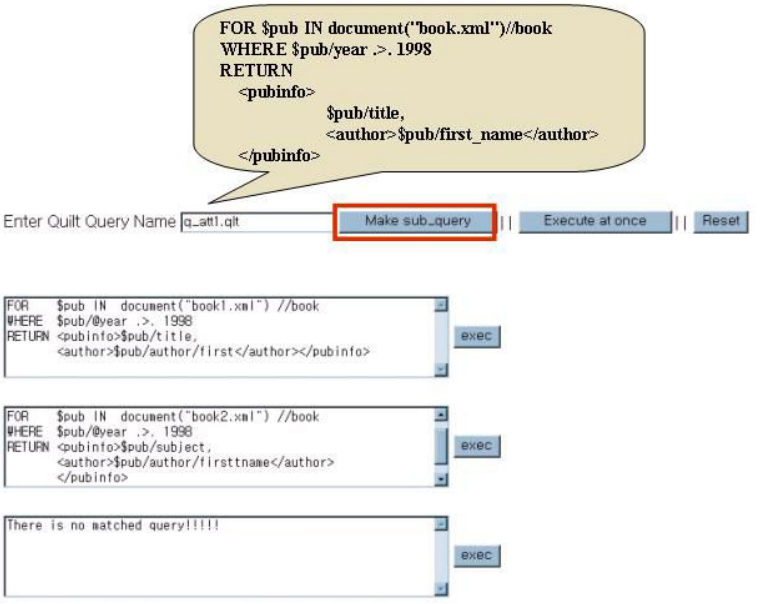


Fig. 6. Example of local query generation

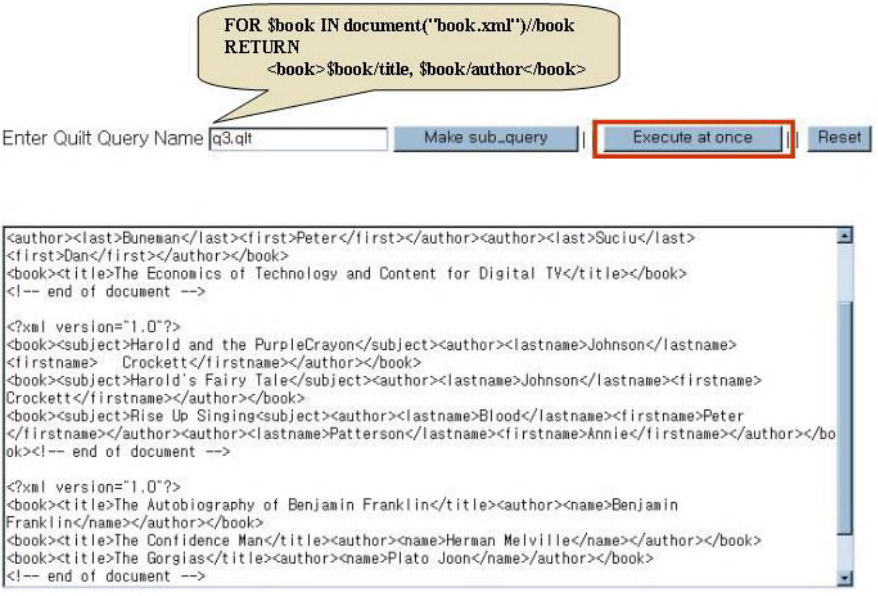


Fig. 7. A query execution result over three documents

DDXMI file is generated by collecting the index numbers. When generating the DDXMI file, the mapping direction is changed from the original local-as-view to global-as-view, to make query rewriting straightforward. A master query from a user is then translated to queries to local documents by looking up the corresponding paths in the DDXMI. Finally the results from local documents are integrated, using the named semantic functions. Our DDXMI system runs under the NT operating system using a Java servlet server and the JavaCC compiler. Any XML documents can be handled without rebuilding or modification. The only requirement for using this approach is to be familiar with the XML data model, and with the relevant documents.

However some issues remain to be investigated. The current mapping mechanism only allows describing is-equivalent mappings between the master paths and local paths. In order to fully use knowledge on the local documents for query decomposition and optimization, it is planned to extend the mapping description power to support describing and using more sophisticated kinds of relationship, and also relationships at more levels, such as local path vs. local path, document vs. document, and document vs. path. Second, the current prototype does not support paths that contain wildcards, but this may happen in a document DTD that includes recursive elements, such as a document for a family tree, or for locations that contain sub-location(s).

If some data does not appear in a local database but is known to the global database, we may speak of 1:0 mapping, and name a semantic function that provides the missing information. An example is when a publisher's database does not include a publisher element. This is not yet implemented, but should not be difficult. More generally, one might consider N:M mappings. If only queries are considered, this is unnecessary, since it decomposes to N mappings that are each 1:M, but if both queries and updates are considered, the concept of N:M mapping is convenient in certain cases.

Another significant issue to be addressed in the future concerns the evolution, or maintenance of metadata about documents and document collections. Even if the documents themselves do not change (which is unlikely if they are really databases, but likely they are really books, pamphlets, magazines, TV shows, etc), it is very possible for our interests and understandings of the documents to change, resulting in different DTDs, which would in turn require generating a new DDXMI. For this reason, it would be desirable to implement an incremental DDXMI tool, that would start from the previous DDXMI file and slightly modify it in light of slight modifications to DTDs, instead of generating it from scratch. Technology for this is well known in computer science, though the implementation would of course require some effort. In addition, we plan to migrate to XML schemas, instead of DTDs.

References

1. C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, V. Chu. XML-Based Information Mediation with MIX. Exhibition program, ACM Conf. Management of Data, SIGMOD'99, Philadelphia, 1999.

2. D. Brickley, R. Guha, eds. Resource Description Framework (RDF) Schema Specification. W3C Proposed Recommendation. March 1999. www.w3.org/1999/TR/PR-rdf-schema.
3. K. Beard, T. Smith. A framework for meta-information in digital libraries. In W. Sheth, W. Klas (eds), Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media. McGraw-Hill, pp. 341–365. 1998.
4. Online Computer Library Center, Inc. Dublin Core Metadata Element Set: Reference Description, 1997. Office of Research and Special Projects, Dublin, Ohio. www.oclc.org:5046/research/dublin_core/
5. S. Cluet, P. Veltri, D. Vodislav. Views in a Large Scale XML Repository. 27th VLDB, Roma, Italy, 2001.
6. D. Chamberlin, J. Robie, D. Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. WebDB 2000, LNCS, Springer, 2000.
7. D. A. Grossman, S. M. Beitzel, E. Jensen, O. Frieder. The IIT Intranet Mediator. IEEE IT Professional, January/February, 2002.
8. A. Gupta, B. Ludascher, M.E. Martone. Knowledge-Based Integration of Neuroscience Data Sources. 12th Intl. Conference Scientific and Statistical Database Management (SSDBM), Berlin, Germany, IEEE Computer Society, July, 2000.
9. D. Egnor, R. Lord. Structured Information Retrieval using XML. In Working Notes of the ACM SIGIR Workshop on XML and Information Retrieval, Athens, Greece. 2000.
10. L.M. Haas, R.J. Miller, B. Niswonger, M. Tork Roth, P.M. Schwarz, E.L. Wimmers. Transforming Heterogeneous Data with Database Middleware: Beyond Integration. Bulletin IEEE Computer Society Technical Committee on Data Engineering. 1999.
11. I. Nishizawa, A. Takasu, J. Adachi. A Schema Integration and Query Processing Information Retrieval. IPSJ SIGNotes DataBase System Abstract 094–009. 1993.
12. A.Y. Levy, A. Rajaraman, J.J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. VLDB, pp. 251–262, Bombay, India, 1996.
13. R.J. Miller, L.M. Haas, M.A. Hernandez. Schema Mapping as Query Discovery. 26th VLDB. Cairo, 2000.
14. Y.K. Nam, J. Goguen, G. Wang. A Metadata Integration Assistant Generator for Heterogeneous Distributed Databases. LNCS 2519, 2002, pages 1332–1344, Springer. Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems, October 2002.
15. A. Paepcke. An object-oriented view onto public, heterogeneous text databases. In Proceedings Ninth International Conference on Data Engineering, pages 484–493, April 1993.
16. Y. Papakonstantinou, H. Garcia-Molina, J. Ullman. MedMaker: A Mediation System Based on Declarative Specifications. Data Engineering (ICDE). 1996.
17. J.D. Ullman. Information Integration Using Logical Views. 6th Int. Conference Database Theory, LNCS 1186, Springer, 1997.
18. G. Wiederhold. Mediators in the Architecture of Future Information System. IEEE Computer 25:3, pp. 38–49, 1992.
19. XML Metadata Interchange. www-4.ibm.com/software/ad/library/standards/xmi.html. (XMI)

Effective Similarity Search Methods for Large Video Data Streams*

Seok-Lyong Lee[1], Seok-Ju Chun[2], and Ju-Hong Lee[3]

[1]School of Industrial and Information Engineering, Hankuk University of FS, Korea
sllee@hufs.ac.kr
[2]Dept. of Information and Comm. Engineering, KAIST, Korea
chunsj@islab.kaist.ac.kr
[3]School of Computer Science and Engineering, Inha University, Incheon, Korea
juhong@inha.ac.kr

Abstract. In this paper, we investigate the similarity search methods for large video data sets that are the collection of video clips. A video clip, a sequence of video frames describing a particular event, is represented by a sequence in a multidimensional data space. Each video clip is partitioned into *video segments* considering temporal relationship among frames, and then similar segments of the clip are grouped into *video clusters*. Based on these video segments and clusters, we define similarity functions and present two similarity search methods: the *HR* (hyper-rectangle)-*search* and the *RP* (representative point)-*search*. Experiments on synthetic sequences as well as real video clips show the effectiveness of our proposed methods.

1 Introduction

A video database may contain a number of video clips that can be represented by multidimensional data sequences (MDS's). In our earlier work [7], we have formally defined an MDS S with K points in the n-dimensional space as a sequence of its component vectors, $S = \langle S[1], S[2], \ldots, S[K] \rangle$, where each vector $S[j]$ $(1 \le j \le K)$ is composed of n scalar entries, that is, $S[j] = (S^1[j], S^2[j], \ldots, S^n[j])$. A video clip consists of multiple frames in the temporal order, each of which can be represented by a multidimensional vector in a feature space such as RGB or YCbCr color space. Thus, a video clip is modeled as a sequence of points in a multidimensional space such that each frame of the sequence constitutes a multidimensional point, whose components are feature values of the frame. A sequence is partitioned into video segments (or video shots) and then similar segments are grouped into a video cluster.

A video segment or cluster is represented by a hyper-rectangle (HR) that bounds all points in the segment or cluster and is an extension of a minimum bounding rectangle (MBR). The similarity search problem in this paper is described as follows:

Given: A set of video clusters, a query video, and a threshold ε

Goal: To find the sets of video segments that are similar to a query within ε

* This work was supported by Hankuk University of Foreign Studies Research Fund of 2003

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1030–1039, 2003.
© Springer-Verlag Berlin Heidelberg 2003

A query is given as a video clip which may consist of a single video frame, a single video segment with multiple frames, or multiple video segments. In the first two cases, a single answer set that contains relevant video segments is obtained, while multiple answer sets are obtained for the third case. It is natural to provide one answer set for each query video segment since video segments may be quite different each other even though they belong to a single video clip.

Similarity search methods for one-dimensional sequences have been extensively studied in the domain of time-series analysis [1, 2, 6, 9, 11]. All these methods however address the similarity search for one-dimensional time-series data in which each element of the time series is a real number. Thus they do not handle an MDS. In this paper, we have extended the traditional similarity search method for one-dimensional time series data to support multidimensional data sequences.

On the other hand, various methods [3, 4, 5, 12] have been proposed for the video search, most of which are a key-frame-based method. These methods extract a set of feature attributes like color, texture, shape, or layout from key frames. The retrieval is performed by matching the feature attributes of the query with those of the key frames in a database. However, they suffer from two problems. They miss relevant video segments since the retrieval mechanisms are based on the selected frames, leading not to preserve the correctness, and they are not very effective in representing video segments since their key frames are usually selected to be start and/or end frames. In reality, if a video segment is a part of cartoons or animation films, the contents of frames change drastically within the segment and thus start/end frames do not represent the segment well. To represent the contents of a video segment more exactly, various sophisticated methods have been studied, such as PanoramaExcerpts [10] which presents the entire visible contents of a segment extended with the camera pan or tilt by automatically detecting camera movements. It however requires considerable overhead.

In this paper, we present two methods. First, the *HR-search* is to find relevant segments in which the retrieval is processed on the basis of hyper-rectangles and the correctness is preserved. Any qualified segment with respect to a given query is not missed. Preserving the correctness is an important motivation of the *HR*-search. We believe that guaranteeing the correctness is one of the important features in the similarity search, especially for application domains that are correctness-sensitive, such as an application detecting criminal activities from stored CCTV films.

Our next method is based on the representative point (RP) of a video segment or cluster, called *RP-search*. The representative point is not chosen to be a start or an end point. We present a simple and effective method to select the representative point, considering geometric characteristics of a hyper-rectangle. The *RP*-search however does not guarantee the correctness. This method is designed to achieve a high precision rate while sacrificing the correctness slightly. It can be applied to the applications in which the precision is more critical than the recall.

2 Preliminaries

In this section, we provide basic definitions and mention the segmentation and clustering technique briefly to describe our proposed methods. A hyper-rectangle *HR*

with k points, P_j ($1 \le j \le k$), in the n-dimensional space, is represented by two endpoints, L(low point) and H(high point), of its major diagonal, and the number of points in the rectangle as follows: $HR = \langle L, H, k \rangle$, where $L = \{(L^1, L^2, \ldots, L^n) \mid L^i = min_{1 \le j \le k} (P_j^i)\}$, and $H = \{(H^1, H^2, \ldots, H^n) \mid H^i = max_{1 \le j \le k} (P_j^i)\}$ for $i = 1, 2, \ldots, n$. Using a hyper-rectangle we define a video segment and a video cluster formally as follows:

Definition 1. A video segment VS that contains k points in the temporal order, P_j for $j = 1, 2, \ldots, k$, is defined as follows: $VS = \langle sid, offset, HR \rangle$, where sid is a segment-id, $offset$ is an offset of VS from a start point of a sequence, and $HR = \langle L, H, k \rangle$ such that $L = \{(L^1, L^2, \ldots, L^n) \mid L^i = min_{1 \le j \le k} (P_j^i)\}$ and $H = \{(H^1, H^2, \ldots, H^n) \mid H^i = max_{1 \le j \le k} (P_j^i)\}$ for $i = 1, 2, \ldots, n$.

Definition 2. A video cluster VC with r video segments, VS_j for $j = 1, 2, \ldots, r$, is defined as follows: $VC = \langle cid, slist, HR \rangle$, where cid is a cluster-id, $slist$ is a list of sid's, and $HR = \langle L, H, k \rangle$ such that $L = \{(L^1, L^2, \ldots, L^n) \mid L^i = min_{1 \le j \le r} (VS_j.HR.L^i)\}$, $H = \{(H^1, H^2, \ldots, H^n) \mid H^i = max_{1 \le j \le r} (VS_j.HR.H^i)\}$ for $i = 1, 2, \ldots, n$, and $k = \Sigma_{1 \le j \le r} (VS_j.HR.k)$.

The segmentation is the repeating process of merging a point of a sequence into a segment if predefined conditions are satisfied. Consider a point of a sequence to be merged to a segment in the n-dimensional space. Then, the segmentation is done in such a way that if the merging of the point into the segment satisfies the conditions then it is merged into the current segment, otherwise a new segment is started from the point. We have defined two categories of conditions: the geometric bounding condition that considers the volume and edge of a segment and the semantic bounding condition that considers the distance between consecutive points of the sequence. Then similar segments in a sequence are grouped into a cluster considering those conditions. Due to the space limitation, we do not describe the segmentation and clustering algorithm in detail. Interested readers are referred to [8] for the details such as the conditions, algorithms and experimental results. Next, we introduce the metric to measure the distance between video segments. It is defined for same-length segments and then extended to different-length segments.

Definition 3. The distance $D_{VS}(VS_1, VS_2)$ between two video segments VS_1 and VS_2 of equal lengths, each of which has k points, is defined as the mean distance of the two, where the mean distance is defined as follows:

$$D_{VS}(VS_1, VS_2) = D_{mean}(VS_1, VS_2) = \frac{1}{k} \cdot \sum_{i=0}^{k-1} d(VS_1[i], VS_2[i]) \quad (1)$$

where $d(*,*)$ is the Euclidean distance between two points. For different-length segments that cannot be compared directly, point by point, the shorter segment is compared to the other by sliding it from the beginning to the end. The shortest of all distances of compared pairs is adopted as the distance between two segments. The following definition describes it more formally.

Definition 4. Let us consider the distance $D_{VS}(VS_1, VS_2)$ between two video segments

VS_1 and VS_2 of different lengths, each of which has p and q points respectively. That is, $p = VS_1.HR.k$ and $q = VS_2.HR.k$. Without loss of generality we assume $p \le q$. Then $D_{VS}(VS_1, VS_2)$ is defined as the minimum of mean distances of compared pairs, where the minimum mean distance is defined as follows:

$$D_{VS}(VS_1, VS_2) = \min_{1 \le j \le q-p+1} D_{mean}(VS_1[1:p], VS_2[j:j+p-1]) \quad (2)$$

3 Similarity Search Methods

3.1 HR-Search Method

To measure the distance between hyper-rectangles of video segments or video clusters in the n-dimensional space, we define the distance D_{HR} as the following, depending on their relative placement.

Definition 5. The distance D_{HR} between two hyper-rectangles, HR_1 and HR_2, in the n-dimensional space, is defined as the minimum Euclidean distance between two hyper-rectangles. That is:

$$D_{HR}(HR_1, HR_2) = \sqrt{\sum_{i=1}^{n} x_i^2}, \text{ where } x_i = \begin{cases} |HR_1.H_i - HR_2.L_i| & \text{if } HR_1.H_i < HR_2.L_i \\ |HR_1.L_i - HR_2.H_i| & \text{if } HR_2.H_i < HR_1.L_i \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Observation 6. The distance D_{HR} is shorter than the distance between any pair of points, one in a hyper-rectangle HR_1 and the other in a hyper-rectangle HR_2. That is:

$$D_{HR}(HR_1, HR_2) \le \min_{P_1 \in HR_1, P_2 \in HR_2} d(P_1, P_2)$$

where P_1 and P_2 are the points that are contained in HR_1 and HR_2, respectively.

Observation 7. When a hyper-rectangle HR_1 is contained in the other hyper-rectangle HR_2 ($HR_2 \supseteq HR_1$), for an arbitrary hyper-rectangle HR_3, $D_{HR}(HR_3, HR_2) \le D_{HR}(HR_3, HR_1)$.

Based on Observation 6 and 7, we derive the following two lemmas, showing the lower bounding relationships with respect to D_{HR} and D_{VS}.

Lemma 8 (*Lower Bounding Distance*: $D_{HR}(VS_q, VS_t) \le D_{VS}(VS_q, VS_t)$).
The distance $D_{HR}(VS_q, VS_t)$ between two hyper-rectangles of a query segment VS_q and a video segment VS_t in a database is the lower bound of the distance $D_{VS}(VS_q, VS_t)$ of the two segments.

$$D_{HR}(VS_q, VS_t) \le D_{VS}(VS_q, VS_t \quad (4)$$

Proof: Let VS_q, VS_t have k, l points, and P_q, P_t be the arbitrary points that are contained in VS_q, VS_t, respectively. Without loss of generality, we assume $k \le l$, since we can switch VS_q and VS_t if $k > l$. Then:

$$D_{VS}(VS_q, VS_t) = \min_{1 \le j \le l-k+1} D_{mean}(VS_q[1:k], VS_t[j:j+k-1]) \ge \min_{P_q \in VS_q.HR, P_t \in VS_t.HR} d(P_q, P_t)$$

By Observation 6, the following holds:

$$D_{HR}(VS_q, VS_t) \le \min_{P_q \in VS_q, P_t \in VS_t} d(P_q, P_t)$$

Therefore, we conclude: $D_{HR}(VS_q, VS_t) \le D_{VS}(VS_q, VS_t)$.

Lemma 9 (*Lower Bounding Distance*: $D_{HR}(VS_q, VC) \le min_{VSt \in VC} D_{HR}(VS_q, VS_t)$). Let a video cluster *VC* contain one or more video segments. Then, the distance $D_{HR}(VS_q, VC)$ between two hyper-rectangles of a query segment VS_q and a video cluster *VC* in a database is the lower bound of the distance $D_{HR}(VS_q, VS_t)$ between two hyper-rectangles of VS_q and any video segment VS_t in *VC* ($VS_t \in VC$). That is:

$$D_{HR}(VS_q, VC) \le min_{VSt \in VC} D_{HR}(VS_q, VS_t)) \quad (5)$$

Proof: By the definition of a video cluster in Definition 2, a hyper-rectangle of a video cluster tightly bounds all hyper-rectangles of video segments. Since $VS_t.HR \subseteq VC.HR$ for all $VS_t \in VC$, by Observation 7, the distance $D_{HR}(VS_q, VC)$ is equal to or shorter than the distance $D_{HR}(VS_q, VS_t)$ for all $VS_t \in VC$.

Using Lemma 8 and 9, we can use the distances $D_{HR}(VS_q, VS_t)$ and $D_{HR}(VS_q, VC)$ to prune irrelevant video segments from a database without *'false dismissals,'* since they provide the lower bound with respect to the distance $D_{VS}(VS_q, VS_t)$. Given a query segment VS_q, let us consider a query, "*Find a set of video segments* VS_t *such that* $D_{VS}(VS_q, VS_t) \le \varepsilon$". By Equation 4 and 5, the following holds:

$$D_{HR}(VS_q, VC) \le D_{HR}(VS_q, VS_t) \le D_{VS}(VS_q, VS_t) \le \varepsilon, \text{ for all } VS_t \in VC \quad (6)$$

We first prune irrelevant video clusters and collect candidate video clusters using Equation 5. Then, each video segment in candidate video clusters is evaluated with respect to Equation 4 in order to prune irrelevant video segments and collect candidate video segments. Those candidate video segments are finally evaluated in the refinement process using the distance $D_{VS}(VS_q, VS_t)$ to get the final answer set of video segments. Before a query is processed, we do some pre-processing to extract feature vectors from video frames of each video clip, and then to map the video clip to an MDS. Each MDS is segmented and clustered during the video segmentation and clustering process. The similarity search is performed to select similar video segments from a database. A brief description of the *HR*-search algorithm is as follows.

Algorithm HR-search

```
/* A query video is segmented to i(≥1) video segments(VS_i).*/
/* For VS_i, a single answer set VS_i_ANS is reported.          */
Input: a set VC of video clusters, a query video clip Q,
       and a threshold ε
Output: answer sets {VS_i_ANS} of video segments
Step 0: /* Initialization */
        VS_i_ANS ← φ,     /* an answer set for VS_i          */
        VS_Q ← φ          /* a set of query video segments */
Step 1: /* Query segment generation from a query video  */
        Map a query clip to a multidimensional sequence
        Segment the sequence to one or more video segments
        VS_Q ← {query video segments generated}
Step 2: /* Filtering by D_HR */
        for each query video segment VS_i in VS_Q
```

```
        for each video cluster VC_j in the set VC
          if D_HR(VS_i, VC_j) ≤ ε then
            for each video segment VS_k in the cluster VC_j
              if D_HR(VS_i, VS_k) ≤ ε then
                VS_i_ANS ← VS_i_ANS ∪ {VS_k}
Step 3: /* Refinement by D_VS: sequential scan of video segments*/
        for each VS_m in the set VS_i_ANS
          if D_VS(VS_i, VS_m) > ε then
            VS_i_ANS ← VS_i_ANS − {VS_m}
Step 4: return set {VS_i_ANS}
```

3.2 RP-Search Method

We propose a simple and effective method to choose a representative frame from each video segment and video cluster using geometric characteristics of a hyper-rectangle. The following two definitions describe how to select the representative point (i.e. frame) for the video segment and the video cluster, respectively.

Definition 10. Consider a video segment VS with k points P_j ($1 \le j \le k$). A representative point RP_{VS} of VS is defined as follows: $RP_{VS} = (Pt \mid Pt \in \wp, d(Pt, P_c) = min_{1 \le j \le k}\, d(P_j, P_c))$, where $\wp$ is a set of points in VS and P_c is the center of $VS.HR$ given by $P_c = \left(\frac{VS.HR.L^1 + VS.HR.H^1}{2}, ..., \frac{VS.HR.L^n + VS.HR.H^n}{2} \right)$.

Definition 11. Consider a video cluster VC with r video segments, VS_j for ($1 \le j \le r$). A representative point RP_{VC} of VC is defined as follows: $RP_{VC} = (Pt \mid Pt \in \wp, d(Pt, P_c) = min_{1 \le j \le r}\, d(RP_{VS,j}, P_c))$, where $\wp$ is a set of representative points of video segments in VC, $RP_{VS,j}$ is RP_{VS} of the jth video segment in VC, and P_c is the center of $VC.HR$ given by $P_c = \left(\frac{VC.HR.L^1 + VC.HR.H^1}{2}, ..., \frac{VC.HR.L^n + VC.HR.H^n}{2} \right)$.

The nearest point from the center of a hyper-rectangle may represent well the characteristics of all points in the video segment or cluster even though it is not always the case. The basic idea is that if this point is used for the similarity search processing instead of the start or end frame that is selected as a key frame, the search effectiveness would be improved since it summarizes the geometric characteristics of a hyper-rectangle better. Our experiment shows that the similarity search method based on this representative point performs better than the traditional key-frame based search. By applying a representative point to Definition 1 and 2, we can extend the definitions of a video segment and a video cluster to $VS = \langle sid, offset, HR, RP_{VS} \rangle$ and $VC = \langle cid, slist, HR, RP_{VC} \rangle$, respectively.

The distance between representative points of video segments or clusters, D_{RP}, used in the *RP*-search method, is obtained simply, since D_{RP} is basically the Euclidean distance between two points in the multidimensional data space. When V_x and V_y are arbitrary video segments or video clusters in the space, D_{RP} is defined as follows:

$$D_{RP}(V_x, V_y) = d(V_x.RP, V_y.RP) \quad (7)$$

The *RP*-search is based on the representative points of video segments and clusters. Different from the *HR*-search, only a representative point of each video segment is indexed for similarity search processing. We do not describe this method in detail since the retrieval procedure is almost the same as traditional methods such as [3, 5] that adopt the key frame based search. The only difference is that in our method the search is based on the representative point which is the nearest point from the center of a hyper-rectangle, while traditional methods use a start point and/or an end point of a video segment. A query clip is also parsed and segmented to one or more video segments. A representative point is selected for each query segment on which the query processing is based.

4 Experimental Evaluation

In order to measure the effectiveness of the proposed methods, we have conducted experiments on synthetic data sets as well as real videos, since real-life data sets are generally restricted in size and in setting various parameters while synthetically generated data provide better flexibility. Our methods are mainly designed for and evaluated on a video, but they are not restricted to be used on the video only. Other application domains whose data formats can be represented by sequences such as the time-series and signal processing can also benefit. That is one of the reasons we have also evaluated our methods on synthetic data sets. The evaluation system is implemented in Microsoft VC++ under Windows Server environment.

All experimental data sets are for convenience 3-dimensional, but our methods do not restrict the dimensionality. We use two data sets, synthetic data sequences that are obtained using a Fractal function and real video data that are captured from a collection of TV news, dramas, and documentary films, as depicted in Table 1.

Table 1. Test data used

	Synthetic data	Video data
# of segments	12,700	5,632
# of queries	50	30
Threshold (ε)	0.01-0.30	0.01-0.30

The number of query sequences is 50 for synthetic data and 30 for real video data, respectively. Multiple query results are collected and averaged to produce final results. The threshold range was chosen from 0.01 to 0.30. We believe this range provides the proper selectivity, since our search space is normalized to have the range [0, 1] for each dimension and the threshold 0.30 corresponds to the search sphere with a diameter 0.60. The evaluation is mainly made with respect to the precision and the recall that are well known in the similarity search applications. Let a set of retrieved segments by a query be *Set*(*ret*), and a set of relevant segments be *Set*(*rel*). When the number of elements in the set *S* is denoted by |*S*|, the precision and the recall are defined as follows:

$$precision = \frac{|Set(ret) \cap Set(rel)|}{|Set(ret)|}, \quad recall = \frac{|Set(ret) \cap Set(rel)|}{|Set(rel)|} \tag{8}$$

In order to measure the effectiveness of our proposed methods, it is necessary to set the ground truth that provides the basis to evaluate the retrieval quality. In general, a video contains a huge amount of information that consists of a great number of frames. It is not easy for the human to decide how similar a pair of video segments is. In our experiment, we set the ground truth as the set of retrieved segments by the sequential scanning method using a measure D_{VS}.

Precision: In Fig. 1, we present the comparison of the precision among the *SP*-search that selects a start frame as a representative, the *SE*-search that selects both a start and an end frame, the *RP*-search, and the *HR*-search for synthetic sequences and real videos. As we can see in the graphs, the *RP*-search shows the highest precision, 0.60-0.84 for synthetic sequences and 0.53-0.86 for real videos. It is superior to other methods along the whole range of threshold values. From the fact, we observe that the representative point by our method provides a good representation for a segment or a cluster. The *SP*-search is the next.

On the other hand, the *SE*- and the *HR*-search show relatively low precision. In these methods, two points are involved for the retrieval, start/end points for the *SE*-search and low/high points for the *HR*-search, respectively. It is general that more segments are retrieved when two points are used for the search, leading more irrelevant segments to be retrieved. That is the reason why those two methods show low precision. The precision of the *HR*-search is in the range of 0.17-0.70 for synthetic sequences and 0.26-0.81 for real videos. For thresholds higher than 0.05, it is above 0.38 for synthetic sequences and above 0.48 for real videos. We believe that this precision is quite usable in the practical environment. In addition, real videos show higher precision than synthetic sequences. It is because real-life data is usually well-segmented at the shot boundary, showing more agglomerative characteristics than synthetic data.

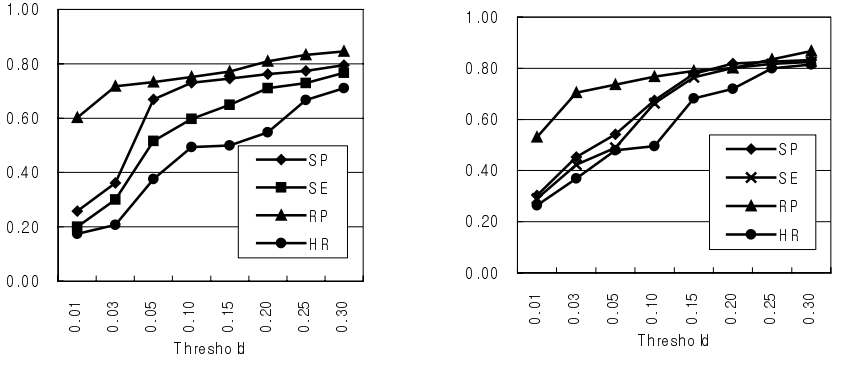


Fig. 1. Precision for synthetic sequences and real videos

Recall: Fig. 2 shows the recall of each method for synthetic sequences and real videos, respectively. In both graphs, the *HR*-search shows the recall of 1.0. It means that this method does not allow any false dismissal. The *SE*-search is the next, showing 0.16-0.77 for synthetic sequences and 0.55-0.85 for real videos. Different from the case of precision, the methods in which two points are involved for the search, such as the *SE*- and the *HR*-search, show higher recall than other methods that use a single point for the search. It is known that there is a trade-off between the precision and the recall. When the precision becomes high, the recall becomes low and vice versa. The *RP*-search shows relatively good recall. The range is 0.28-0.73 for synthetic sequences and 0.50-0.82 for real videos. In the case of real videos, the recall of the *RP*-search is very close to the *SE*-search, sometimes becoming superior for some thresholds. But the *SP*-search shows poor recall for both synthetic and real videos, preventing it to be used for recall-sensitive applications.

From experiments on the precision and recall, we conclude that the *RP*-search is the most preferable with respect to the precision while the *HR*-search is the best with respect to the recall.

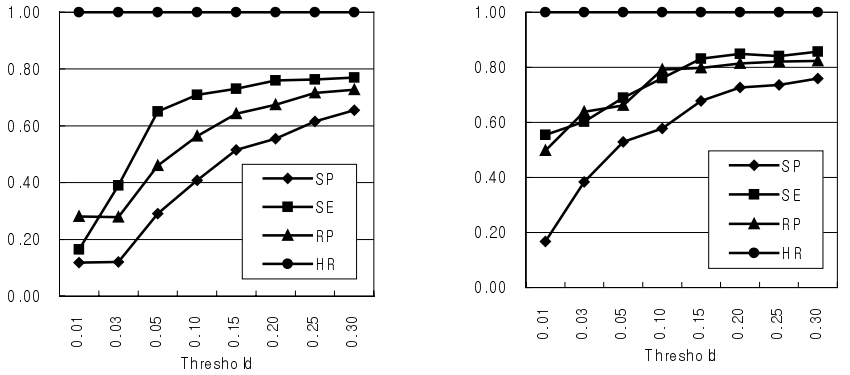


Fig. 2. Recall for synthetic sequences and real videos

5 Conclusions

The similarity retrieval of sequence data sets is one of the great potential areas in database applications since it can be extensively applied to various application domains, such as time-series data, digital and analog signals. In this paper, we have extended the traditional similarity search method for one-dimensional time series data to support multidimensional data sequences, and presented two similarity search methods. One is a correctness-preserving method called '*HR*-search' in which the query is processed based on the hyper rectangles of video segments and clusters, and the other is a method called the '*RP*-search' based on the representative points of video segments and clusters. To evaluate the similarity, we have defined the distance measures, D_{HR} between two hyper-rectangles and D_{RP} between two representative points, of video segments or clusters. The *HR*-search guarantees '*no false dismissal*'

while maintaining reasonable level of the precision, and the *RP*-search provides high precision compared to other traditional methods while maintaining a fairly good recall rate. Thus this method can be used for applications in which the precision is more important than the recall.

The methods proposed in this paper can be widely used for various application domains in which the data format is represented by multidimensional sequences. One of potential applications which is emphasized in this paper is the similarity query on large video data streams, but we believe other application areas can also benefit. As the future work, we plan to study on applying the proposed methods to other application domains considering their own characteristics, such as the voice signal matching and the region-based image search.

References

1. R. Agrawal, C. Faloutsos, and A. Swami: Efficient Similarity Search in Sequence Databases. Proceedings of Foundations of Data Organizations and Algorithms (1993)
2. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos: Fast Subsequence Matching in Time-Series Databases. Proc. of ACM SIGMOD (1994) 419–429
3. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker: Query by Image and Video Content: The QBIC System. IEEE Computer, Vol. 28, No. 9 (1995) 23–32
4. A. Hampapur, A. Gupta, B. Horowitz, C. F. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain: Virage Video Engine, Proc. of SPIE: Storage and Retrieval for Image and Video Databases V. (1997) 188–197
5. A. K. Jain, A. Vailaya, and X. Wei, Query by Video Clip, Multimedia Systems Vol.7 (1999) 369–384
6. E.J. Keogh, K. Chakrabarti, S. Mehrotra, and M.J. Pazzani: Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. Proc. of ACM SIGMOD (2001) 151–162
7. S. L. Lee, S. J. Chun, D. H. Kim, J. H. Lee, and C. W. Chung: Similarity Search for Multidimensional Data Sequences. Proc. of IEEE Int'l Conference on Data Engineering (2000) 599–608.
8. S. L. Lee and C. W. Chung: Hyper-Rectangle Based Segmentation and Clustering of Large Video Data Sets. Information Science, Vol. 141, No. 1–2 (2002) 139–168
9. D. Rafiei: On Similarity Queries for Time Series Data. Proc. of Int'l Conference on Data Engineering (1999) 410–417
10. Y. Taniguchi, A. Akutsu, and Y. Tonomura, PanoramaExcerpts: Extracting and Packing Panoramas for Video Browsing, Proceedings of ACM Multimedia (1997) 427–436
11. B. K. Yi and C. Faloutsos: Fast Time Sequence Indexing for Arbitrary Lp Norms. Proc. of Int'l Conference on VLDB (2000) 385–394
12. H. J. Zhang, J. Wu, D. Zhong, and S. W. Smoliar, An Integrated System for Content-Based Video Retrieval and Browsing, Pattern Recognition Vol. 30 (1997) 643–653

A New Method for Locating the Global Optimum: Application of the Cutting Angle Method to Molecular Structure Prediction

Kieran F. Lim (林 百 君)[1], Gleb Beliakov[2], and Lynn Batten[2]

[1] Centre for Chiral and Molecular Technologies, School of Biological and Chemical Sciences, Deakin University, Geelong, Victoria 3217, Australia
lim@deakin.edu.au

[2] School of Information Technology, Deakin University, Burwood 3125, Australia
{gleb, lmbatten}@deakin.edu.au

Abstract. Many problems in chemistry depend on the ability to identify the global minimum or maximum of a function. Examples include applications in chemometrics, optimization of reaction or operating conditions, and non-linear least-squares analysis. This paper presents the results of the application of a new method of deterministic global optimization, called the cutting angle method (CAM), as applied to the prediction of molecular geometries. CAM is shown to be competitive with other global optimization techniques for several benchmark molecular conformation problem. CAM is a general method that can also be applied to other computational problems involving global minima, global maxima or finding the roots of nonlinear equations.

1 Introduction

Many problems in chemistry depend on the ability to identify the global minimum or maximum of a function. Examples include important applications in chemometrics [1], flow rates or pH concentrations in the optimization of flow analysis methods [2-4], statistical process control in manufacturing, the maximization of yields in synthesis and manufacturing, non-linear least-squares analysis and the ability to predict molecular geometries. In particular, the last-mentioned application includes the areas of protein space structure elucidation, the investigation of host-guest interactions, the understanding of properties of superconductors and zeolites [5], and the identification of transition states [6]. In all of these situations, one needs to find the molecular geometry which corresponds to the global minimum of the potential energy surface (PES).

Global and deep local minima correspond to stable molecular conformations, and they dictate both physical and chemical properties of chemical substances. Hence, the search for the global minimum on the PES is one of the most important, albeit one of the most challenging, optimization problems in chemistry and mathematics. Typically,

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1040–1049, 2003.
© Springer-Verlag Berlin Heidelberg 2003

a mix of local and global search techniques is necessary. The presence of large numbers of shallow local minima on the PES makes the problem unsuitable for traditional local descent techniques in the absence of other information, and so new local search strategies have emerged in the last few years. On the other hand, global search techniques perform poorly when only localized information is available [7, 8]. Thus, the success of a local/global search approach relies heavily on the availability of both local and global knowledge of the PES.

The cutting angle method described here is a new global search procedure. The authors implement it alongside known local search procedures. Existing global algorithms used in the determination of molecular structures can be classified into four (overlapping) categories: (1) deterministic methods, (2) stochastic methods, (3) heuristic methods and (4) surface smoothing methods [9]. Reviews of these methods have been given in [10]; applications to molecular conformation problems can be found in [7, 8, 11].

The computational cost of all known global search methods increases exponentially with the number of variables and so those methods are "NP-hard" [12]. For the purposes of practical results in the molecular conformation problem, it is therefore desirable to make the algorithms as efficient as possible so that conformations of a reasonably large size (or dimension) can still be managed.

This paper applies the cutting angle method, which is a new, efficient, deterministic method for global optimization, and demonstrates that it compares favorably with other deterministic techniques (systematic search, branch-and-bound, generalized descent) on a chosen set of chemical structures. Like other global optimization methods, it can also be applied to a range of other generic problems in theoretical chemistry (e.g., based on cluster analysis, similarity searching, etc.[7]).

2 Theory

The need for global optimization techniques in molecular modeling has been emphasized by many authors [7, 8, 11]. The number of local minima on the potential energy surface typically grows exponentially with the number of variables, and enumeration of all local minima using multistart local descent methods is not feasible. Moreover any method that relies only on the local information (such as derivatives), has very slim chances to locate the global minima.

The cutting angle method (CAM) was first proposed in 1999 by Andramonov, Bagirov, Glover and Rubinov, and subsequently improved by them [13-16]. It can be used for both differentiable and non-differentiable functions. The one-dimensional Piyavskii-Shubert algorithm [17], illustrated in Figure 1, emerges as a special case of CAM. The concept of the method is to replace the original optimization problem with a sequence of simpler (relaxed) optimization problems, so that the sequence of solutions of relaxed problems converge to the global minimum of a real-valued function $f(x^k), x^k \in R^n$. CAM is based on the construction of a sequence of *supporting functions* (e.g., hypercones), and these must be chosen so that the maximum of the supporting functions is the lower approximation of *f*.

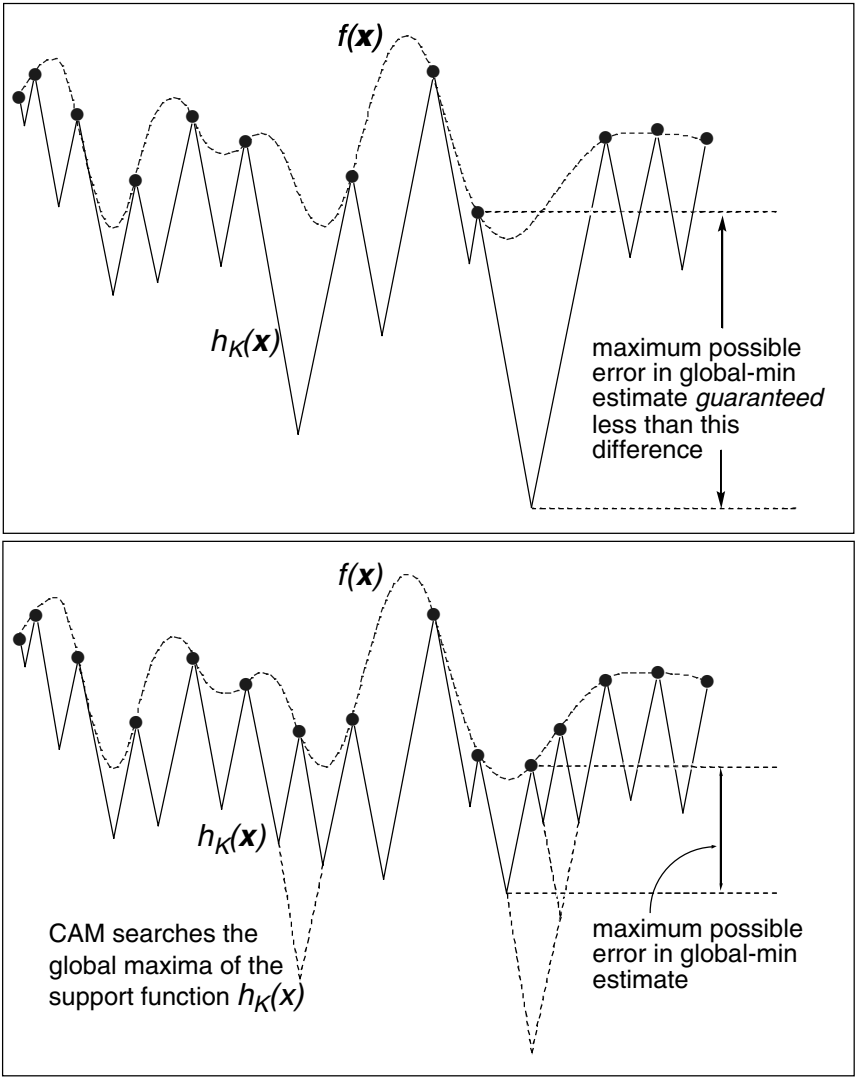


Fig. 1. Generalized cutting plane method of Lipschitz optimization (Pijavski-Shubert method). All local minima of the saw-tooth cover can be explicitly enumerated. As the number of function evaluations increases, the "teeth" become smaller and the supporting functions $h_K(x)$ converges pointwise to the required function $f(\mathbf{x})$. The global minimum of $h_K(x)$ converges to the global minimum of $f(\mathbf{x})$

For smooth functions, the *Lipschitz property* is usually interpreted as meaning that the function has a bounded first derivative. More rigorously, we define a real-valued function to be *Lipschitz continuous* if there is a real number L satisfying

$$| f(\boldsymbol{x}) - f(\boldsymbol{y}) | < L \ \| \boldsymbol{x} - \boldsymbol{y} \ \| \tag{1}$$

for all points x and y for which f is defined. While we use the absolute value norm on the left hand side of (2), on the right, any norm can be used. We define the *Lipschitz constant* for f to be the least possible value for L in Eq. (1), over all x and y. The Lipschitz constant is important in our situation because it provides global information about the PES: global information is required in order to find and confirm the global minimum of the PES on any surface. In the mathematics literature, Lipschitz programming has attracted much attention as a method of (deterministic) global optimization [11, 17]. We make fundamental use of the choice of Lipschitz constant in applying CAM to the molecular structures considered in this article.

Figure 1 shows a function f that has been evaluated at several points x^1, x^2, x^3, etc. From each evaluated point on the function f, it is possible to construct a hypercone, with slope L, and with its apex touching the function f. By using the l_∞-norm in (2), we obtain min-type support functions. After K function evaluations at x^1, x^2, …,x^K, the support function

$$h_K(x) = \max_{k=1,2,\ldots,K} \min_{i=1,\ldots,n} \frac{x_i}{x_i^k} f(x^k), \quad x \in S_1^n \tag{2}$$

is a lower approximation to the function f [15]. Note that the form of $h_K(x)$ is very irregular (Figure 1), which is why it is called the *saw-tooth cover of f*. As more points $f(\boldsymbol{x})$ are evaluated, $h_K(x)$ converges point-wise to the function f from below. In particular, the sequence of global minima of $h_K(x)$ converges to the global minimum of the function f [15]. An important result is that, for any particular K value, the global minimum of $h_K(x)$ is *guaranteed* to be a lower bound for the true global minimum of the PES f, thus *measuring the uncertainty in the estimated global minimum* of the PES f, when the algorithm is terminated at some finite K.

Uncertainty in estimated global minimum

$$< \min\{f(\boldsymbol{x}^k)\} - \min\{h_K(x)\}, \quad x \in S_1^n \tag{3}$$

$$\text{where } S_1^n = \left\{ x \in R^n, x_i \geq 0, \sum x_i = 1 \right\} \tag{4}$$

We have previously described a fast method of building support functions and presented the computational algorithm [18], including transformation of variables from R^n to S_1. This improvement to earlier results has allowed practical application of the CAM to medium size problems involving 10-30 variables. In this paper, we apply a version of the CAM to molecular modelling problems where a global minimum of PES is needed.

3 Application to Molecular Structure Prediction

In this section, the application of the cutting angle method to the problem of prediction of molecular conformation is described. The function f to be minimized is the potential energy surface V.

Many models have been developed describing interactions within a molecule in terms of atomic bonds and effective interactions. Parameterizations of molecular potential functions (also known as Force Fields) include ECEPP, AMBER, CHARMM, GROMOS, MM3, DISCOVER and others (see the review [19]). Interfacing with these and other public domain and commercial software packages is important for practical applicability of novel optimization methods. In this study we chose ECEPP/3 force field (and software package) to model polypeptides.

In common with other authors, all the bond lengths and bond angles have been held fixed (eg, see References [11, 20-24]). Hence the conformations of the various molecules considered here, depend only on the torsional or dihedral angles. In turn, the potential energy depends only on the torsional potential energy terms and those corresponding to interactions between non-bonded atoms:

$$V = \sum_{i>j} V_{\text{non-bonded}}(r_{ij}) + \sum_{k} V_{\text{torsion}}(t_k), \tag{5}$$

where t_k are the dihedral (torsional) angles and $r_{ij} = r_{ij}(t_1, t_2, t_3, \ldots)$ are the distances between atoms i and j, which in turn are functions of the dihedral angles t_k. In Eq.(5), the contribution of three-body and higher-order terms to $V_{\text{non-bonded}}$ has been neglected.

CAM was used to calculate the conformations of derivitised naturally-occurring amino acids of the form $CH_3CO\text{-}NH\text{-}CRH\text{-}CO\text{-}NH\text{-}CH_3$, where R is the amino acid residue. The parent amino acids were proline, alanine, histidine, tyrosine and glumine, covering the range from 5 to 10 dihedral angles.

The CAM algorithm was interfaced with the ECEPP/3 potential energy surface of Scherga et al.[24] CAM was used as the "driver" algorithm to optimize the dihedral angles (global variables), but at each set of dihedral angles, the potential energies were evaluated by calls to ECEPP/3, and the three-atom angles ("θ angles") in the end groups were optimized within the ECEPP/3 package as local variables (described in the Appendix).

Conformations of several single aminoacids were computed initially to test the performance of the combined CAM-ECEPP/3 running on a cluster of DEC Alpha workstations.

The CAM software was written in C. The ECEPP/3 package [24] was compiled on the same workstation and ran as an external process each time the potential energy was needed. The combined CAM-ECEPP/3 code worked as desired, and the choice of internal coordinates ensured that none of the chiral centres was inverted. (Racemisation can be a problem when Cartesian coordinates are used with Monte Carlo (Metropolis) methods.)

The CAM was then applied to a neutral form of met-enkephalin, shown in Figure 2, which is a benchmark molecular conformation problem [11]. It involves 24 independent

Table 1. Performance of the Cutting Angle Method on the prediction of conformation of single aminoacids and met-enkephalin

Parent aminoacid	Number of variables	Energy of most stable geometry	Function evaluations	CPU seconds
Proline (Pro)	5	-19.81	200	21
Alanine (Ala)	7	-5.18	5000	640
Histidine (His)	8	-8.92	8000	1067
Tyrosine (Tyr)	9	-8.48	4000	607
Glutamic acid (Glu)	10	-15.87	10000	4252
Met-enkephalin	24	-11.706	120,000	4740
Energies in kcal mol^{-1}. CPU time is for a single DEC Alpha workstation, except the last row, where CPU time is for a cluster of 36 DEC Alpha workstations				

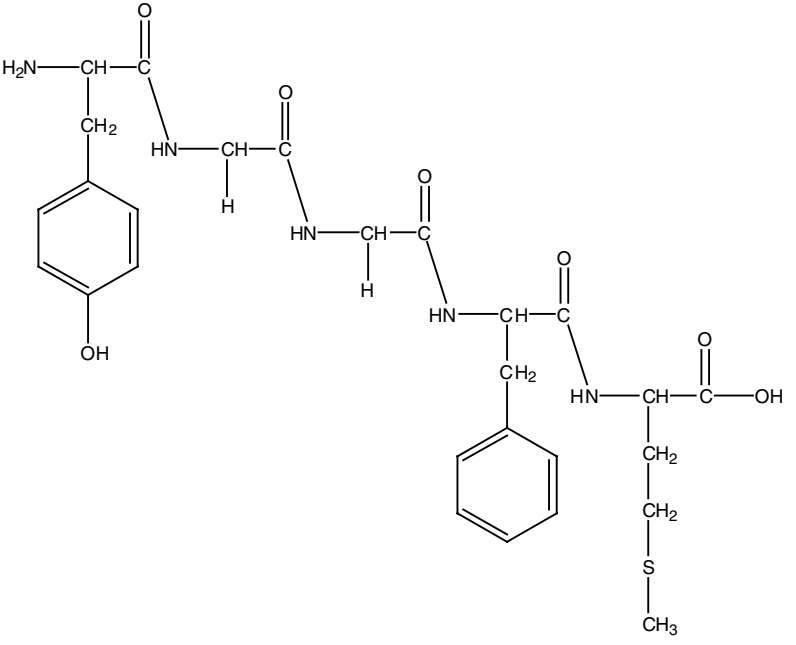


Fig. 2. The neutral form of met-enkephalin (Tyr-Gly-Gly-Phe-Met)

dihedral angles, giving rise to a very complex PES, involving. in the order of 10^{11} local minima [11]. This problem is very challenging because of the existence of several strong local minima, which terminate (or "trap") local-descent algorithms. Local-search algorithms using random initial points fail to identify the global minimum; moreover, failures have been reported even if the initial point is chosen close to the

global minimum [11]. As with the substituted amino acids, CAM was interfaced with the ECEPP/3 package [24] (described in the Appendix).

The global minimum at -11.706 kcal mol^{-1} was found by the combination of the CAM and ECEPP/3 internal local optimizer using the procedure described in the Appendix.

4 Discussion

Floudas has previously reported that the αBB technique [11] has outperformed other methods (Monte Carlo, electrostatically driven Monte Carlo, diffusion equation, simulated annealing and others) for the met-enkephalin molecule. Calculations on this molecule have also illustrated the inadequacy of multistart local search techniques [11]. The αBB technique required about 400,000 function evaluations to locate the global minimum. Using the same ECEPP/3 potential function [24], the CAM of this work was able to locate the global minimum using 120,000 function evaluations, *thus out-performing the other optimization methods*. The met-enkephalin calculations have been performed in 79 minutes on a cluster of 36 DEC Alpha workstations.

ECEPP/3 was used as a standalone software package to which calls were made from our CAM driver module. CAM requires only the ability of third party software to compute the value of the objective function (PES), and thus poses little restriction on the choice of such software. This paper demonstrates that CAM can be implemented successfully on a multiprocessor computer.

Since CAM constructs a sequence of support functions that are strict lower bounds for the PES, Eq. (3) measures the uncertainty in the estimated global minimum of the PES when the algorithm is terminated after K iterations. Conversely, provided that there are sufficient computer resources, the algorithm can be continued until the estimated global minimum is within some *guaranteed tolerance* of the true global minimum. For computational convenience, the CAM algorithm uses normalized variables Eq. (4), which makes it compatible with third-party computer code describing any Lipschitz continuous function.

Although this paper has applied CAM to geometry optimization, CAM be applied to any global optimization method, including clustering and other problems in chemometrics [1], transition-state modelling [6], variational transition state theory [25], and optimization of flow analysis methods [2-4].

CAM can also be used to find the root(s) of nonlinear equations, e.g.

$$F(\boldsymbol{x}) = 0$$

through the transformation

$$G(\boldsymbol{x}) = \|F(\boldsymbol{x})\|^2 = 0$$

where G is the square of the norm of the original vector function F. The roots of F correspond to the global minima of G. Applications of this kind include the variational-theory method in quantum mechanics and Monte Carlo sampling in statistical

mechanics. These applications of the cutting angle method will be utilized in future work.

5 Conclusion

The problem of finding the global minimum on a hypersurface with multiple local minima has many important applications in chemistry. In this paper, we have applied a new global optimization method to the prediction of molecular conformation modelling. In these applications, the typical number of local minima grows exponentially with the number of variables, and is enormous even for relatively small molecules. Traditional local search techniques fail to detect global minima, and hence miss the likely molecular conformations, whereas stochastic global methods fail to confirm or give an estimate of the global minimum.

This paper has demonstrated that the cutting angle method (CAM) is competitive with (if not better than) other global optimization techniques in finding the minimum-potential-energy structures, when applied to benchmark molecular calculations for the met-enkephalin molecule.

CAM is a general global optimization method that can also be applied to other computational problems involving global minima or global maxima. It is a robust technique that can be applied to the optimization of non-differential (but Lipschitz continuous) functions.

Acknowledgments. The authors wish to acknowledge support from the Australian Partnership in Advanced Computing (APAC) (Merit scheme grant d88) and School of Information Technology, Deakin University (internal grant). KFL thanks Ms Jeanne Lee (李 静 宁) (Australian Catholic University) for encouraging and helpful discussions.

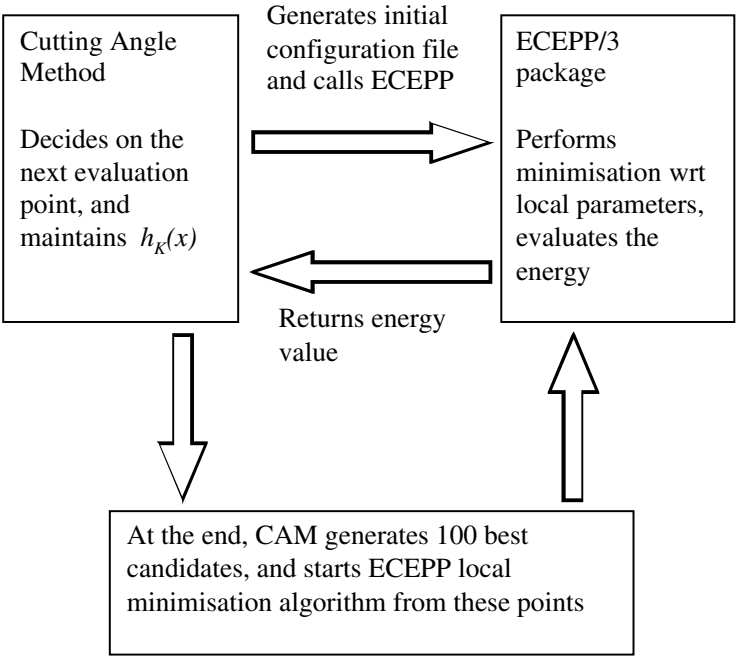
References

1. M. J. Adams, *Chemometrics in Analytical Spectroscopy*, Royal Society of Chemistry, London (1995)
2. A. Cladera, E. Gomez, J. M. Estela and V. Cerda, *Effect of Variables Influencing S-1/2 in Sequential Injection Analysis – Extrapolability of S-1/2 Based Results Between SIA Designs*, Talanta, 43 (1996), 1667–1674
3. S. M. Sultan, A. M. Almuaibed and A. Townshend, *Flow injection chemiluminescence determination of medazepam*, Fresenius Journal of Analytical Chemistry, 362 (1998), 167–169
4. A. Townshend and R. A. Wheatly, *Oxidative chemiluminescence assay of 2,4-dinitrophenylhydrazine*, Analyst, 123 (1998), 1041–1046

5. J.-R. Hill, C. M. Freeman and L. Subramanian, *Use of force fields in materials modeling*, in K. B. Lipkowitz and D. B. Boyd, eds., *Reviews in Computational Chemistry*, Wiley-VCH, New York, 2000, 141–216
6. J. E. Eksterowicz and K. N. Houk, *Transition state modeling with empirical force fields*, Chemical Reviews, 93 (1993), 2439–2461
7. A. Neumaier, *Molecular modeling of proteins and mathematical prediction of protein structure*, SIAM Review, 39 (1997), 407–460
8. P. M. Pardalos and C. A. Floudas, *Optimization in computational chemistry and molecular biology : local and global approaches*, Kluwer Academic Publishers, Boston (2000)
9. R. V. Pappu, R. K. Hart and J. W. Ponder, *Analysis and application of potential energy smoothing and search methods for global optimization*, Journal of Physical Chemistry B, 102 (1998), 9725–9742
10. M. P. Allen and D. J. Tildesley, *Computer Simulations of Liquids*, Oxford Science Publications, Oxford (1990)
11. C. A. Floudas, *Deterministic global optimization: Theory, methods, and applications*, Kluwer Academic Publishers, Dordrecht ; London (2000)
12. R. Horst and P. M. Pardalos, *Handbook of global optimization*, Kluwer Academic Publishers, Dordrecht ; Boston (1995)
13. M. Andramonov, A. Rubinov and B. Glover, *Cutting angle methods in global optimization*, Applied Mathematics Letters, 12 (1999), 95–100
14. A. Bagirov and A. Rubinov, *Global minimization of increasing positively homogeneous function over the unit simplex*, Annals of Operations Research, 98 (2000), 171–187
15. A. M. Rubinov, *Abstract convexity and global optimization*, Kluwer Academic Publishers, Dordrecht ; Boston (2000)
16. A. Bagirov and A. Rubinov, *Modified versions of the cutting angle method*, in N. Hadjisavvas and P. M. Pardalos, eds., *Convex analysis and global optimization*, Kluwer, Dordrecht, 2001, 245–268
17. P. Hansen and B. Jaumard, *Lipschitz optimization*, in R. Horst and P. Pardalos, eds., *Handbook of global optimization*, Kluwer, Dordrecht, 1995, 407–493
18. L. M. Batten and G. Beliakov, *Fast algorithm for the cutting angle method of global optimization*, Journal of Global Optimization, 24 (2002), 149–161
19. M. Jalaie and K. B. Lipkowitz, *Published force field parameters for molecular mechanics, molecular dynamics, and Monte Carlo simulations*, in K. B. Lipkowitz and D. B. Boyd, eds., *Reviews in Computational Chemistry*, Wiley-VCH, New York, 2000, 441–486
20. J.-P. Ryckaert, *Special geometrical constraints in the molecular dynamics of chain molecules*, Mol. Phys., 55 (1985), 549–556
21. C. Ciccotti and J.-P. Ryckaert, *Molecular dynamics simulations of rigid molecules*, Comput. Phys. Reports, 4 (1986), 345–392
22. R. Kutteh and T. P. Straatsma, *Molecular dynamics with general holonomic constraints and application to internal coordinate constraints*, in K. B. Lipkowitz and D. B. Boyd, eds., *Reviews in Computational Chemistry*, Wiley-VCH, New York, 1998, 75–136
23. G. Némethy, M. S. Pottle and H. A. Scheraga, *(ECEPP/2: empirical conformational energy program for peptides and proteins)*, Journal of Physical Chemistry, 87 (1983), 1883
24. G. Némethy, K. D. Gibson, K. A. Palmer, C. N. Yoon, G. Paterlini, A. Zagari, S. Rumsey and H. A. Scheraga, *(ECEPP/3: empirical conformational energy program for peptides and proteins)*, Journal of Physical Chemistry, 96 (1992), 6472
25. R. G. Gilbert and S. C. Smith, *Theory of Unimolecular and Recombination Reactions*, Blackwells Scientific, Oxford (1990)

Appendix: Schematic Overview of the Software Setup

ECEPP/3 software was installed on the same computer as our CAM global optimization algorithm. The variables in CAM were transformed into the appropriate domain for the dihedral angles $[-\pi,\pi]$, and their values were supplied to the ECEPP/3 software in a suitable configuration file. ECEPP/3 was called as an external process, which minimized the potential energy with respect to local variables keeping the global variables constant. The CAM then read the value of the energy from the file generated by ECEPP/3. This process was repeated at each iteration of the CAM, and at the end (after a predefined number of iterations), CAM generated 100 good starting points for the ECEPP/3 local minimizer. Finally, the local optimization process with respect to all variables started from the specified 100 points to quickly improve the CAM solutions. The best solution was selected as the global minimum.



Performance Characteristics of a Cost-Effective Medium-Sized Beowulf Cluster Supercomputer

Andre L.C. Barczak[1], Chris H. Messom[1], and Martin J. Johnson[1]

Massey University, Institute of Information and Mathematical Sciences, Albany Campus Private bag 102 904,
North Shore Mail Center, Auckland, New Zealand
{a.l.barczak, c.h.messom, m.j.johnson}@massey.ac.nz
http://www.massey.ac.nz

Abstract. This paper presents some performance results obtained from a new Beowulf cluster, the Helix, built at Massey University, Auckland funded by the Allan Wilson Center for Evolutionary Ecology. Issues concerning network latency and the effect of the switching fabric and network topology on performance are discussed. In order to assess how the system performed using the message passing interface (MPI), two test suites (mpptest and jumpshot) were used to provide a comprehensive network performance analysis. The performance of an older fast-ethernet/single processor based cluster is compared to the new Gigabit/SMP cluster.

1 Introduction

The Helix is the newest and most powerful supercomputer (Beowulf cluster) in New Zealand. Its hardware was funded by Alan Wilson Center for Molecular Ecology and Massey University a centre of research excellence established in 2002. The primarily purpose of the Helix is to run Bioinformatics programs such as BLAST for whole genome searches, however secondary users span the fundamental sciences, mathematics and computer science. Having built an experimental cluster three years ago [1], the Center for Parallel Computing at the Institute of Information and Mathematical Sciences (IIMS) was given the opportunity to design and build a significantly larger cluster. Several research groups in mathematics, computer science and bioinformatics within New Zealand would benefit from such a machine. The budget was extremely limited and it was necessary to choose hardware that would give the best price/performance possible. To evaluate the price/ performance ratio both the floating point and integer performance were considered. Many benchmarks are available today (see [2]) but few are able to characterize a cluster in a useful and reproducible way. Tests are often misleading, Gropp and Lusk discuss many of the traps found in performance measurement [3]. The Linpack benchmark was chosen as the floating point benchmark as it would test the scalability of the Helix rather than just the raw peak performance. The integer benchmark was not required to scale as many bioinformatics applications that primarily make use of integer operations

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1050–1059, 2003.
© Springer-Verlag Berlin Heidelberg 2003

are "embarrassingly parallel", that is, they require little interprocess communication, meaning that they scale well. As well as the macro performance of the machine as illustrated by the Linpack benchmark, we were also interested in the micro performance of the Helix system. In order to do this we used the 'mpptest' test suite developed by Gropp and Lusk [3]. Using these programs to measure different characteristics of the cluster has lead to some interesting observations. A ring test was also carried out, showing that the topology has virtually no effect on point to point operations.

1.1 Background

With the availability of cheap commodity processors and network interconnects, the cluster architecture for supercomputers has become popular. As commodity processors have increased in cock speed and are able to simultaneous execute multiple floating point instructions they have started to rival the performance of dedicated scientific processor architectures. The programming techniques suitable for networked computers are fully discussed in [4] For several years researchers have been investigating the clustering of legacy hardware using discarded pcs, servers. and network cards[5], [6]. These systems can produce good performance figures for embarrassingly parallel applications, but suffer from scalability problems due to small memory, low bandwidth and high latency interconnects. An interesting study of the effects on applications and the sensitivity to differences in bandwith and latency is found in [7]. With recently released processors such as the Intel Xeons and AMD Athlons, cheap RAM and commodity gigabit Ethernet with reasonable latency it has become possible to build scalable high performance computer system.

2 A Brief Description of the Clusters

2.1 The Sisters

The Sisters was the first Beowulf cluster built at Massey University. It consists of 16 nodes connected by a fast Ethernet network. The nodes use Pentium III CPUs running at 667MHz with 256MB Ram. The server is a dual Pentium III with 1GB Ram. There is a single switch linking the main server and the nodes. Key bottlenecks in the system include the high latency and low bandwidth network interconnects.

2.2 The Helix

Given the requirement of a homogeneous cluster for the Helix, the main decision was whether to use Xeon or Athlon processors. For similarly rated cpus Xeon 2.2GHz and Athlon MP2200, the Xeon's clock speeds are higher, but the floating point performance are about equivalent while the Athlon has a significant advantage on integer operations. Bioinformatics applications rely heavily on integer

operations, therefore even with equivalent costs the applications would favour the choice of the Athlon processor. In terms of price/ performance the floating point performance favours the Athlon. Considering all options the nodes were built with dual Athlon MP2100+ CPUs, 1GB memory and dual 3com 64bit gigabit network interface cards. The dual processor configuration was favoured so that the network interconnect costs would be contained. Standard unmanaged gigabit switches with two network interface cards in each node were chosen. The two network interface cards will theoretically increase each nodes available bandwidth, but due to bus contention issues, the main advantage of the two cards are the increased connectivity, namely the number of nodes that are directly connected via a single switch. The nodes are arranged in rows and columns on the same switch and each node will statically choose the best path (either located in the same switch or with the smallest number of hops) to the next node. Typically, if they are within the same switch they will have a low latency and high bandwidth. Given the constraints, i.e., the number of nodes, availability of ports on the switches etc, the final design is shown in figure 1. In this design there is either a direct connection to another node via a single switch, or at most up to 3 switched.

3 Performance

Tests were conducted with just two nodes, network cards and one network switch from each vendor. The selection of the nodes and switches was determined by the performance on the Linpack benchmark for four processors. Two dual AMD 2100 resulted in a 8.45 Gflops,(Rmax) while two dual Xeon 2.0 resulted in 8.36 Gflops. The bandwidth and latency characteristics of the network are shown in table 1.

Table 1. Gigabit Ethernet real bandwith versus PCI bus

PCI bus	Bandwith
64 bit (133 MHz)	920 Mbps
64 bit (100 MHz)	850 Mbps
32 bit (100 MHz)	600 Mbps

Having selected and assembled the cluster, the testing of the Helix performance on the Linpack benchmark and specific issues regarding the network infrastructure and its effects on overall performance (using mmptest and jumpshot) were investigated.

3.1 Linpack Performance Results

The Linpack benchmark was run as soon as the Helix's construction was completed. The Linpack benchmark measures the performance of a distributed mem-

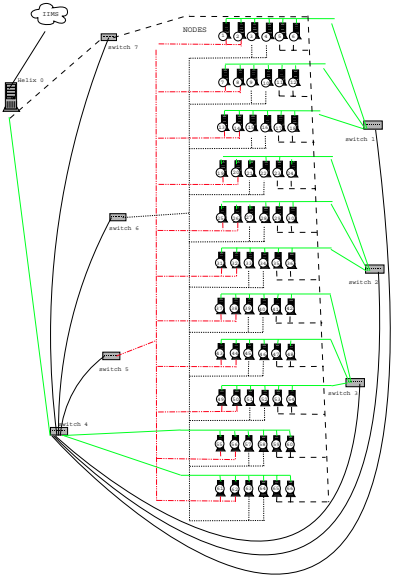


Fig. 1. Helix's topology.

ory computer while solving a dense linear system in double precision. Several factors determine the performance achieved on this benchmark, but the bandwidth and latency are a significant component. High bandwidth and low latency ensure that the communication is not a bottleneck in the algorithm. The algorithm tends to be more efficient for larger problem sizes and since the optimal performance is when the problem fits into available system memory, the larger the memory of the machine the better. Each of the nodes in the Helix contains 1GB of RAM, so this provides a limit to the maximum problem size. Since we require 8 bytes per double precision floating point number, the maximum theoretical matrix size is 94,118 by 94,118. However since there is operating system overhead, a more realistic size using 80% of the memory gives a figure of 80,000 by 80,000. The theoretical peak performance (Rpeak) of the Helix (the maximum Gflop rating that the machine is guaranteed never to exceed) is 448.8 Gflops [9], however this is not a useful measure since it will not be achieved even if every machine is working independently. The peak Linpack performance (Rmax) using a problem size of 82080 was 234.8 Gflops [10].

The performance of the Helix on the Linpack benchmark vs processors (and memory) shows that the system scales almost linearly (see figure 2), this is due to the high bandwidth, reasonable latency switching and the grid layout of the nodes. The switches each have 24 ports and as seen in figure 1 the vertically connections connect up to 22 nodes on a single switch. This means that up to 44 processors can be used with a single high bandwidth hop between processors. When a larger number of processors are required for the Linpack benchmark, then the nodes must be selected to make optimal use of the grid layout. The

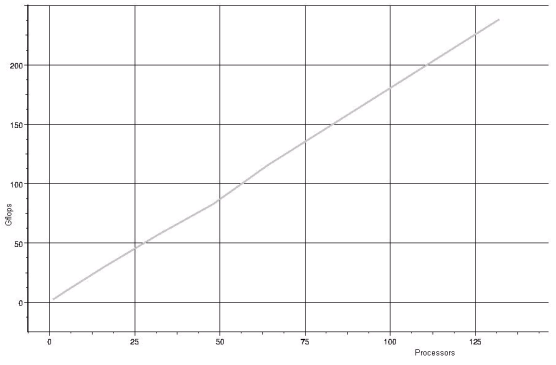


Fig. 2. Linpack Rmax rating versus number of processors.

almost linear increase in performance versus number of processors for the Helix indicates that the architecture is scalable and so additional nodes can be added to the Helix to increase the size of problems solved as well as the speed at which they will be solved. The grid layout can be expanded to accommodate row and column sizes of 23 nodes which will be equivalent to 1058 processors with an expected Linpack rating of 1.88 Tflops putting it in the 24th position in the November 2002 Top 500 list at an estimate cost of approximately US$ 1 million.

3.2 An Analysis of Performance Using Mpptests

There are two main reasons we chose 'mpptests' for the performance assessments. There was a need to assess performance when using standard libraries in a fairly standard way, offering average numbers for the programmers to check their own applications performance. This set of applications allows evaluation of how efficiently the hardware is being used by the whole software structure, including the operating system and MPI. The second reason for using 'mpptests' is that their results are reproducible despite the simplicity of the tests. There are other benchmarks available that will test a variety of the MPI calls, for example SkaMPI [8]. It has been noted before that a single factor can drag performance down significantly. For example, it is well known that certain device drivers do not use the full bandwidth or work poorly with certain Gigabit adapters. For this reason a careful selection of cards and drivers was done.

A point to point test shows that for short messages in a Gigabit Ethernet the times vary very little up to 1000 bytesand their values are around 35 microseconds. This may be considered as an indication of the latency time. This compares favourably with the same test run in the older cluster (the sisters) with fast Ethernet. Due to the low bandwidth as the message size increases the times build up quickly and it goes to 250 microseconds for a message size of 1000 bytes. It is interesting to note that theoretically the gigabit Ethernet should be 10 times faster but due to latency effects, the speedup is not as high for small

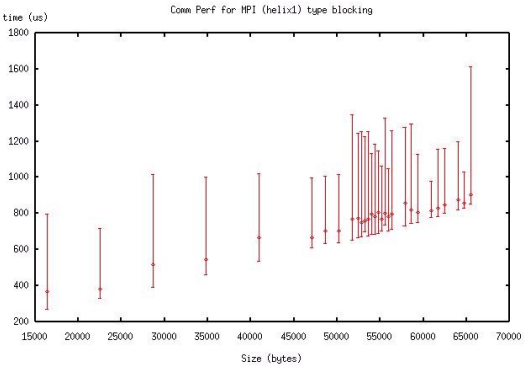


Fig. 3. Point to point messages in Helix (long, blocking)

messages. The latency of less than 40ms is reasonable for the gigabit Ethernet system. The times for even very short messages are about 75% when compared to the fast Ethernet and there is a gain in speed for any message size. The bigger the messages, the more advantageous becomes the use of Gigabit Ethernet.

With longer messages, the picture changes dramatically. Now the bandwidth on the Gigabit Ethernet shows the full advantage of its use. Figure 3 shows that the times grows linearly up to 65KB in the Helix. The same tests running on the Sisters are shown on figure 4. For 65KB the transfer times are approximately 6.7 times the ones obtained on Helix. The same trends were observed when using non-blocking/point to point messages. The maximum theoretical bandwidth in the fast Ethernet and on a Gigabit Ethernet are 12.5MB/s and 125MB/s respectively. For long messages (figures 3 and 4) Fast Ethernet is approximately 87% as fast as the maximum. A similar analysis for the Gigabit Ethernet shows that it is only about 59% as fast as the theoretical maximum.

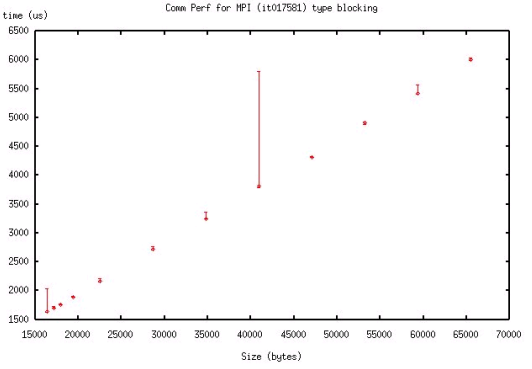


Fig. 4. Point to point messages in Sisters (long, blocking)

The other common tests are the ones with collective operations. Both clusters were measured using "goptest" with -bcast option operations. An interesting effect of the smp machines using Allreduce operations can be noticed in figure 5. The nodes were ordered in such a way that only one processor was used up to 64 nodes, then the list restarts from node 1 and went in order up to the 64th node (the order was 1-2-3-4-...64-1-2-3-4...64). When the Allreduce operation was done over more then 64 processors, the time suddenly increases and it seems

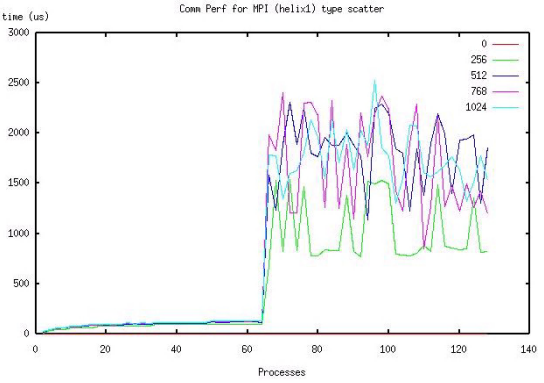


Fig. 5. Goptest running in 128 processors with the default order

to be independent of the message sizes. The test was repeated several times to make sure the effect was not due to some specific state of the networking structure. It was noticed that the same test running with the nodes arranged in a different order had a different impact on the times. Many combinations were tried with varying results. Ordering the nodes properly the effect on the Allreduce operations can be different. In figure 6 the order was 1-2-3-4...63-64-64-63-...4-3-2-1. This effect seems to be due to the way that the AllReduce operator is implemented in mpich. Using the new ordering the number of processes that communicate with each other on the same smp node are reduced, eliminating this detrimental effect.

3.3 Ring Test Using Analysis Jumpshot

An interesting observation was made by Zaki et. al. [11].They found that the Beowulf network structure can sometimes be revealed by running a ring test application in MPI and observing the CLOG file. Applying this same concept, ring tests were run in the Helix cluster. When two neighbouring nodes are not located in the same switch, it would be expected that a slightly longer message passing time would be measured. For example, there are two hops between nodes 36 and 37 (figure 1). Running a ring test with a 127KB message size on the cluster the following CLOG file was obtained and it is shown in figure 7:

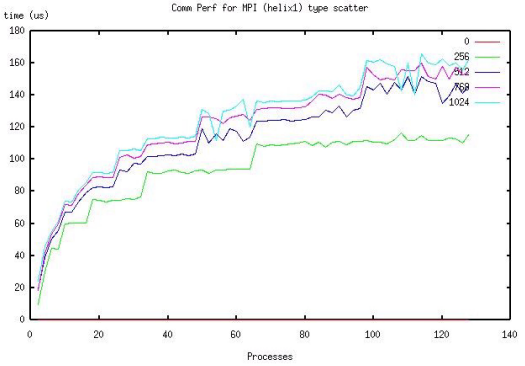


Fig. 6. Goptest running in 128 processors using a different node ordering

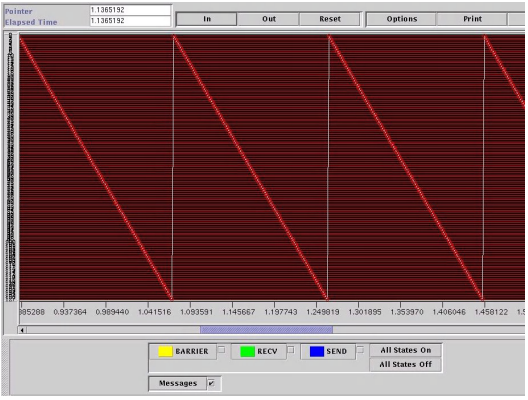


Fig. 7. A ring test using 127KB message and 128 processors

Figure 7 shows that there is no noticeable effect of the topology, even when using relatively long messages. Several tests using the processors in different order were carried out to confirm the results. This figure also shows that the point to point messages are being delivered with the same speed as the single point to point messages in figure 3 with little degradation when making multiple hops over switches. This could be explained by the fact that the additional delay introduced by the switches is small compared to the communication overhead of MPI.

3.4 Bisectional Bandwidth

Figure 8 shows that the bisectional bandwidth of the Helix is approx 220Mbps for a broad range of message sizes. These results were obtained using just one process at each node as reliable reproducible results could not be obtained running two processes in each node. These results are consistent with a theoretical

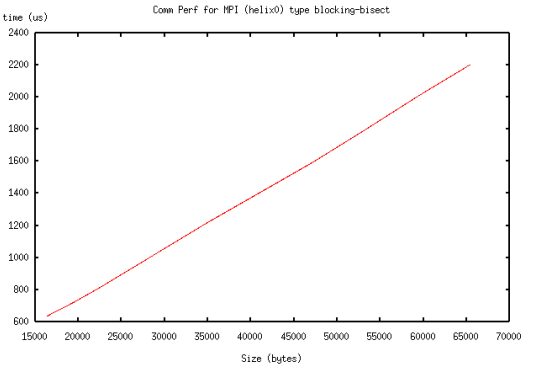


Fig. 8. Bisectional bandwidth of the Helix

analysis of the network topology where we section the network in two halves at the node 32-33 boundary. This means that we have about 110Mbps of full duplex bisectional bandwidth available for each processor which means that the architecture will give reasonable performance for algorithms that have not necessarily been optimised for the Helix. The reasonable bisectional bandwidth is a result of the large number of nodes (almost half of the cluster) that are connected directly to each node by a Gigabit Ethernet connection. As the network size increases to the proposed maximum of 23 x 23 nodes, or 1058 processors, the bisectional bandwidth will not scale well and so the algorithms will have to be modified to suit the grid topology.

4 Conclusions

The use of gigabit Ethernet has proved to be a good choice, even considering that the gain in performance for short messages is not 10 times. Although the relative latency is high, especially with very short messages, there are gains for any message size and the absolute latency is reasonable. The bigger the message the better the gains when compared to fast Ethernet. The effect of using SMP machines with two Gigabit Ethernet adapters each proved to be economical and performs well. Even considering that the two cards cannot achieve their maximum bandwidth when communicating simultaneously due to motherboard constraints it still plays an important role of fairly distributing communication loads across the network. The collective tests showed that by carefully ordering the way the processors work with certain MPI collective operations a better result can be achieved.

References

1. Grosz, L. and Barczak, A.: Building an Inexpensive Parallel Computer. Res. Lett. Inf. Math. Sci.**1** (2000) 113–118
2. Benchweb: www.netlig.org/benchweb/
3. Gropp, and Lusk, : Reproducible measurements of MPI performace characteristics. In: Dongarra,J., Luque, E., and Margalef, T., (editors): Recent Advances in Parallel Virtual Machine and Message Passing Interface, volume 1697 Lecture Notes in Computer Science, pages 11–18. Springer Verlag (1999).
4. Wilkinson, B. and Allen, M.: Parallel Programming. Prentice Hall, New Jersey (1999)
5. Savarese, D. F. and Sterling, T.: Beowulf. In: Buyya, R. (ed.): High Performance Cluster Computing, Vol.1. Prentice Hall PTR, New Jersey (1999)
6. Ridge, D., Becker, D., Merkey, P., Sterling, T.: Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs Proceedings, IEEE Aerospace, 1997
7. Plaat, A., Bal, H.E., Hofman, R.F.H., Kielmann, T.: Sensitivity of parallel applications to large differences in bandwidth and latency in two-layer interconnects Future Generation Computer Systems 17 (6): 769-782 APR 2001
8. Reussner, R.: Recent Advances in SKaMPI In: Krause, E. and Jäger, W. (eds): High Performance Computing in Science and Engineering 2000 Transactions of the High Performance Computing Center Stuttgart (HLRS) 520–530, SPRINGER (2001)
9. clusters.top500.org
10. www.top500.org
11. Zaki,O. and Lusk E. and Gropp,L. and Swider, D: Toward Scalable Performance Visualization with Jumpshot. High Performance Computing Applications vol 13 number 2 (1999) 277–288

A Novel Approach to Role-Based Access Control*

Song-hwa Chae[1], Wonil Kim[1], and Dong-kyoo Kim[2]

[1]The Graduate School of Information and Communication, Ajou University,
Suwon, 442-749, Republic of Korea
{portula,wikim}@ajou.ac.kr
[2]College of Information and Computer Engineering, Ajou University,
Suwon, 442-749, Republic of Korea
dkkim@ajou.ac.kr

Abstract. With the rapid growth of distributed and network systems, sharing resources among many users become more common. As a result of that, we encounter with new problems concerning security and privacy on the shared resources. An access control mechanism such as role-based access control (RBAC) is one of the solutions to cope with these problems. RBAC is an efficient access control mechanism for organization data with role and permission management. In this paper, we propose a new implementation method for RBAC, which uses neural networks instead of tables. By employing neural network, it has advantages of not using multiple storages for role-permission tables and extra mutual exclusive data tables. It also reduces access time for requested role and permission sets.

1 Introduction

As resource sharing among many users becomes more prevalent, we need to apply proper access mechanism on the shared resources. System administrators and software developers need to manage resources efficiently and ensure proper access control of them. An access control mechanism such as role-based access control is widely used method to solve these problems.

There have been many researches on access control mechanism in information security. Access control mechanism is categorized into three areas, such as mandatory access control (MAC), discretionary access control (DAC) and role-based access control (RBAC). MAC is suitable for military system, in which data and users have their own classification and clearance levels respectively. DAC is another access control method on objects with user and group identifications. RBAC has emerged as a widely acceptable alternative to classical MAC and DAC [2][5]. It can be used in various computer systems.

In this paper, we propose a new implementation method for access control mechanism. The main idea is employing neural network for role-based access control mechanism instead of fixed tables. It enables to eliminate the search time of relation tables and easily detects mutual exclusive roles.

* This study was supported by the Brain Korea 21 Project in 2003.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1060–1068, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Chapter 2 explains the basic concept of RBAC and neural network, chapter 3 describes the proposed RBAC system using neural network, and chapter 4 evaluates the proposed system and chapter 5 concludes with future works.

2 Background

2.1 RBAC

RBAC is a very popular mechanism widely used in both research and industry. It does not allow users to be directly associated with permissions, instead each user can have several roles and each role can have multiple permissions. A relationship between users, roles and permissions is shown in Fig1.

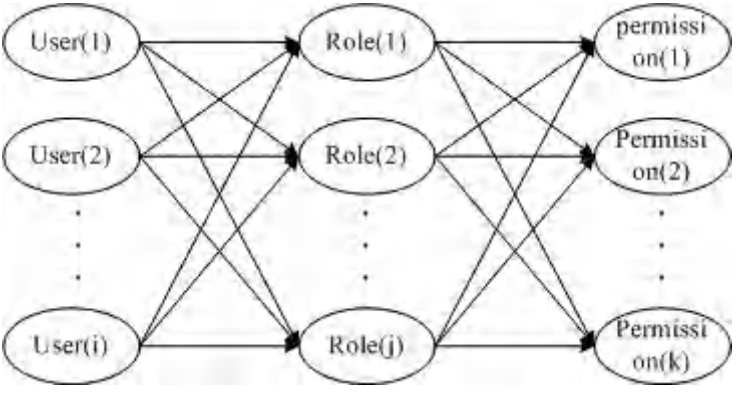


Fig. 1. Relationship among user, role and permission

Each group can be represented as a set of user U, a set of role R, and a set of permission P.

- $U = \{u_1, u_2, \cdots, u_i\}$
- $R = \{r_1, r_2, \cdots, r_j\}$
- $P = \{p_1, p_2, \cdots, p_k\}$

Two different types of association must be managed by the system; one is the association between user and role, the other is the association between role and permission [6]. It is characterized as user-role (UR)[1] and role-permission (RP) relationship[2].

- $UR = \{u \in U, r \in R \mid u \rightarrow 2^{|r|}\}$
- $RP = \{r \in R, p \in P \mid r \rightarrow 2^{|p|}\}$

[1] UR is role assignment function that assigns roles to users.

[2] RP is permission assignment function that assigns permissions to roles. Both UR and PR are represented by bits. It means active or inactive.

Consequently, in order to have proper management, the system needs to maintain two separate association tables. General RBAC implementation uses relation tables as shown in Table 1 and Table 2.

Table 1. User and role example table.

	r_1	r_2	⋯	r_i
u_1	1	0	⋯	1
u_2	1	1	⋯	1
⋯	⋯	⋯	⋯	⋯
u_j	0	0	⋯	1

Table 2. Role and permission example table.

	p_1	p_2	⋯	p_k
r_1	1	0	⋯	1
r_2	1	1	⋯	1
⋯	⋯	⋯	⋯	⋯
r_i	0	0	⋯	1

2.2 Mutual Exclusive Role in RBAC

Conflicts of interest in a role-based system may arise as a result of a user gaining authorization for permissions associated with conflicting roles [4]. For example, if one role requests expenditures and another role approves them, the system must prohibit the same user from being assigned or active to both roles. In order to solve these conflict problems, there have been many researches since middle of 1990's. Finally, National Institute of Standards and Technology (NIST) proposed two Separation of Duty (SOD) reference models in 2001; Static Separation of Duty (SSD) and Dynamic Separation of Duty (DSD). To implement these reference models, the system must have extra tables to protect against activating mutual exclusive roles. According to the NIST reference models, SSD should check the mutual exclusive role in every role assignment, and the role-permission in every user's session request [Fig2]. DSD should check all the three relations for every user's session request [Fig3]. SSD is rarely used in practice and DSD imposes a lot of overload in a system with many users due to excessive table accessing time. In fact, general processes do not always use mutual exclusive roles.

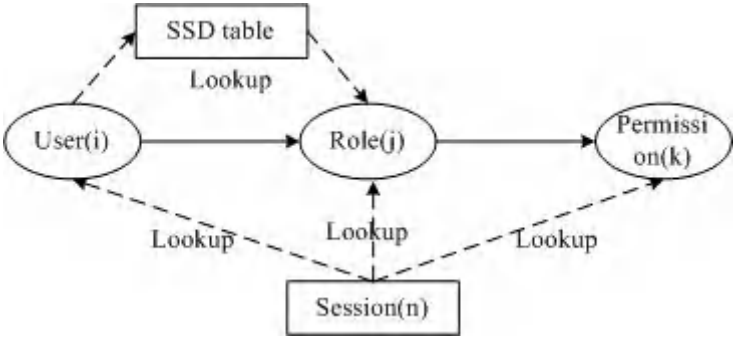


Fig. 2. SSD within RBAC

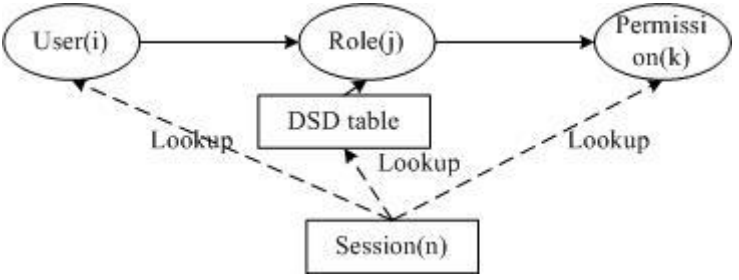


Fig. 3. DSD within RBAC

2.3 Neural Network

Neural network is modeling of human brain's neuron processing. The proposed system employs neural network instead of relation tables. It is trained using backpropagation algorithm.

Backpropagation (BP) algorithm [1] is one of the method of neural network learning. It learns the optimal mapping between input and desired output data by iterative weight update. In this paper, two neural networks are used; one learns the mapping from roles to permissions, and the other learns the reverse mapping between them, from permissions to roles.

3 RBAC Using Neural Network

When a system employs RBAC, it needs at least 3 fixed tables for a session, such as user-role table, role-permission table and mutual excusive role table. It requires extra storage as well as checking time for both role-permission and mutual exclusive role tables. To cope with these problems, we propose an effective implementation method

of RBAC. The proposed RBAC system uses neural networks instead of fixed RBAC relation tables. By employing neural network in RBAC, the system can check the user's permissions in each session without using relation tables. This method does not only reduce access time for authorization but also prevent a user from being activated with mutual exclusive permission. It enables the system effectively applied to DSD in RBAC with many users.

3.1 System Architecture

In the proposed system, roles, permissions and mutual exclusive roles are defined by security administrator as in other RBAC systems. The neural network is trained to learn the relations among them. When a particular user logs into a system, the system is given the user's role set. The system produces the user's whole permission set by using the neural network. The permission provided by neural network is used for authorization. It is assumed that roles, permissions and their associations are static information, whereas associations between user and role are dynamically changed. User can have several roles and role may have multiple permissions too. The process of this system consists of three phases: 1) neural network learning phase, 2) role assignment phase, 3) permission extraction phase. Depending on whether a user's role set contains mutual exclusive roles or not, two cases are considered in system processing.

3.2 Non-mutual Exclusive Role Case

This case is SSD in RBAC. Users do not contain mutual exclusive roles. Thus, the system checks only user's role and permission.

1. Neural Network Learning Phase
   In the learning phase, the training data is the role-permission relation provided by the system administrator. Each role and permission is represented by input and output respectively. The value of input (role) and output (permission) is either '1' (active/permit) or '0' (non-active/denial). Since the proposed system should be able to accommodate hierarchical RBAC, the high level role may contain the low level permissions too. After the neural network learns the relation, the system should be able to respond user's permissions for access control.
2. Role Assignment Phase
   In this system, a user is defined as a human being and a role is a job function within the context of an organization [4]. Therefore, the system can assign multiple roles to the user according to the required job. The association of the user and the role can be changed dynamically. Our RBAC system can successfully respond to these changes. It produces the proper set of permissions dynamically even though the user-role association changes.
3. Permission Extraction Phase
   When the user tries to access data, the system makes decisions such as permit or denial. The proposed system makes the decision using the user's permission set. This permission set is generated by the neural network in the user logging phase.

This set has the user's whole permissions. Since all the permissions do not contain any mutual exclusive role, mutual exclusive resolution is not necessary.

3.3 Mutual Exclusive Role Case

This case is DSD in RBAC. The user can have multiple roles, among which may contain mutual exclusive roles, such as requesting expense and approving that. These two permissions come from two different roles. The main point of this process is to reduce the role set, so that the reduced role set does not have any mutual exclusive permission. With this process, the neural network will be able to produce the reduced permission set according the reduced role set. The process of producing the reduced permission set is as follows.

1. Neural Network Learning Phase
   In this case, we represent permissions with three types of values; '0','1', and '0.5'. The '0' and '1' has the same meaning as defined in non-mutual exclusive role case. Especially, '0.5' means mutual exclusive permission. An example of this is shown in A.1. If there is a high level role containing low level mutual exclusive permission, that node (permission) is also set to '0.5'. In this case, second neural network is employed to produce the reduced role set. It will lean the relation between permissions and roles. Each permission and role will be represented by input and output respectively. An example of this is shown in A.2.
2. Role Assignment Phase
   Same as in non-mutual exclusive case.
3. Permission Extraction Phase
   The DSD system must decide permit or denial on the given request in every session. The proposed system makes the decision using the user's permission set. This permission set is generated by the neural network in the user logging phase. This set has the user's whole permissions including mutual exclusive permission. The process of making the whole permission set is similar to non-mutual exclusive case. When the user tries to access data defined as mutual exclusive permission, the system recognizes mutual exclusion case using permission's value, which is approximately '0.5'. In this case, the system should make a least user's permission set for the session. This permission set is a reduced set which does not contain mutual exclusive permission. Second neural network is employed to produce reduced role set. In order to limit user's permission, it changed a bit to '1' in user's permission set and others to '0'. This changed user's permission set is used to produce the limited role set. The limited role set is used to produce the reduced permission set using the first neural network for this session. After this process, the user has reduced permission set which is not containing mutual exclusive permission. In any session, if the request permission is found to be the mutual exclusive permission, the system does the same process recursively. After the session, it will return to the previous permission set. This process protects from executing mutual exclusive permission in any given session.

4 Evaluation

The neural network was trained and evaluated on a prototype organization [4], such as a small company in Fig 4. It had 2 work parties, 10 roles and 6 mutual exclusive roles. This system could accommodate unlimited number of users. The evaluation result showed that the proposed system detected mutual exclusive role activation and produced the user's permission set. The role-permission relationship of Fig 4 is represented in A.1.

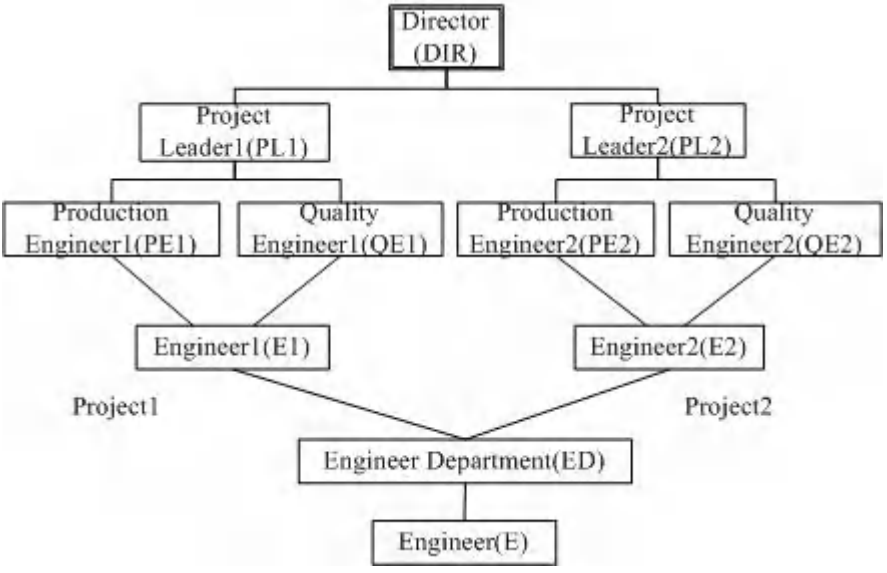


Fig. 4. Example of an organization

We assumed that PL1 and PL2 had mutual exclusive roles. PE and QE had mutual exclusive roles too. Therefore, the permissions of these roles are set to '0.5'.

We trained two neural networks using A.1 and A.2 respectively, with learning and the momentum rates are 0.001. The proposed neural network had one hidden layer (30 nodes) and used sigmoid functions for hidden and output nodes. The five sample users and their role set (representations) are given below;

- User1 : {PE1, PE2} : (0100000100)
- User2 : {PE1, QE2} : (0100001000)
- User3 : {PE1} : (0100000000)
- User4 : {PL1, PE2} : (0001000100)
- User5 : {DIR} : (0000100000)

The system produces user's whole permission set using first neural network. If the user tries to access any mutual exclusive permission, it produces reduced role set using second neural network. For example, user5 is assigned as DIR, the first neural network will produced the whole permission set of {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10} with access values between 0.0 and 1.0. If DIR try to access any mutual

exclusive permission, such as p9, the system recognize it's mutual exclusive case and produce the reduced role PE1 using second neural network. As a result, the role of user5 which was originally defined as a DIR, has reduced to PE1 in this particular session and the permission sets as {p4, p7, p9} accordingly. The whole role and permission set and reduced role and permission set of User1~ User5 are shown in Table3.

Table 3. Whole role set and reduced role set

Case	Whole Role Set	Whole permission Set	Requested Job(permission)	Reduced Role Set	Reduce Permission Set
user1	PE1,PE2	{p2,p4,p7,p9,p10}	p9	PE1	{p4,p7,p9}
user2	PE1,QE2	{p2,p4,p7,p8,p9}	p8	QE2	{p2,p4,p8}
user3	PE1	{p4,p7,p9}	p9	PE1	{p4,p7,p9}
user4	PL1,PE2	{p1,p2,p3,p4,p7,p9,p10}	p10	PE2	{p2,p4,p10}
user5	DIR	{p1,p2,p3,p4,p5,p6,p7,p8,p9,p10}	p9	PE1	{p4,p7,p9}

5 Conclusion and Future Works

RBAC is efficiency access control method for organization data with role and privilege management. It constructs with role, privilege, user, and session, is implement with each relation tables. In this paper, we proposed a new implementation method for RBAC. The proposed methods can be applied dynamically with user's role change and has advantages of not using multiple storages for role-permission tables and extra mutual exclusive data tables. It also reduces access time by eliminating excessive table search for mutual exclusive roles. This method can be easily extended to various access control mechanisms.

References

1. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. Nature, 323, (1986) 533–536
2. E.H.Choun, A Model and administration of Role Based Privileges Enforcing Separation of Duty. Ph.D. Dissertation, Ajou University(1998)
3. K.Mehrotra, C.K.Mohan,S.Ranka, Elements of Artificial Neural Networks, MIT Press (1997)
4. D.F.Ferraiolo, R.Sandhu, E.Gavrila, D.R.Kuhn, R.Chandramouli, Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and System Security, Vol4, No3 (2001) 224–274
5. G.Ahn, R.Sandhu, Role-Based Authorization Constraints Specification, ACM Transactions on Information and System Security, Vol3, No4,207–226 (2000)
6. D.FFerraiolo, J.F.Barkley, D.R. Kuhn, A Role-Based Access Control Model and Reference implementation Within a Corporate Intranet, ACM Transactions on Information and System Security, Vol2, No1 (1999) 34–64

Appendix

A.1 Input and Output Data for First Neural Network Training

	Input data	Out data									
	Role	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
ED	0000000001	0	0	0	1	0	0	0	0	0	0
E1	1000000000	0	0	0	1	0	0	1	0	0	0
E2	0000000010	0	1	0	1	0	0	0	0	0	0
PE1	0100000000	0	0	0	1	0	0	1	0	1	0
PE2	0000000100	0	1	0	1	0	0	0	0	0	1
QE1	0010000000	1	0	0	1	0	0	1	0	0	0
QE2	0000001000	0	1	0	1	0	0	0	1	0	0
PL1	0001000000	0.5	0	1	1	0	0	1	0	0.5	0
PL2	0000010000	0	1	0	1	0	1	0	0.5	0	0.5
DIR	0000100000	0.5	1	0.5	1	1	0.5	1	0.5	0.5	0.5

A.2 Input and Output Data for Second Neural Network Training

	Input data										Out data
	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	Role
ED	0	0	0	1	0	0	0	0	0	0	0000000001
E1	0	0	0	1	0	0	1	0	0	0	1000000000
E2	0	1	0	1	0	0	0	0	0	0	0000000010
PE1	0	0	0	1	0	0	1	0	1	0	0100000000
PE2	0	1	0	1	0	0	0	0	0	1	0000000100
QE1	1	0	0	1	0	0	1	0	0	0	0010000000
QE2	0	1	0	1	0	0	0	1	0	0	0000001000
PL1	0	0	1	1	0	0	1	0	0	0	0001000000
PL2	0	1	0	1	0	1	0	0	0	0	0000010000
DIR	0	1	0	1	1	0	1	0	0	0	0000100000

Designated Verification of Digital Watermark for Network Based Image Distribution*

Hyung-Woo Lee[1] and Im-Yeong Lee[2]

[1] Dept. of Software, Hanshin University, Osan, Kyunggi, Korea, 447-791
hwlee@hanshin.ac.kr
http://infocom.cheonan.ac.kr/~hwlee/index.html
[2] Div. of Information Technology Engineering, Soonchunhyang University, Chungnam, Korea, 336-745
imylee@sch.ac.kr

Abstract. Digital watermark have been proposed as the means for copyright protection on network based digital image distribution framework. However, there are some problems in existing digital watermarking mechanism as the rightful ownership cannot be resolved by current watermarking schemes alone. Specially, some one can provide counterfeited watermark which can be performed on a watermarked image to allow multiple claims of rightful ownerships. Therefore, we can use undeniable verification scheme in the watermarking process. And using designated verifier proofing schemes, watermark can be verified by commitment methods. The confirmer of a watermark can sure that only the intended verifier can be convinced about the validity of the watermark.

1 Introduction

An unlimited number of perfect copies can be illegally produced is a serious threat to the rights of content owners. The primary tool available to help protect content owners's rights is to use a encryption module. However, after receipt and subsequent decryption, the data is no longer protected and is in the clear.

Watermarking compliments encryption. A digital watermark is a piece of information that is hidden directly in media content, in such a way that it is imperceptible to a human observer, but easily detected by a computer. The principal advantage of this is that the content is inseparable from the watermark. This makes watermarks suitable for several applications, including:

- Signatures: The watermark identifies the owner of the content. This information can be used by a potential user to obtain legal rights to copy or publish the content from the contact owner.
- Fingerprinting: Watermarks can also be used to identify the content buyers. This may potentially assist in tracing the source of illegal copies. Each copies holds a watermark that uniquely identifies the player in every movie that is played.

* This work was supported by grant No. (R05-2001-000-01468-0) from the Basic Research Program of the Korea Science & Engineering Foundation.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1069–1078, 2003.
© Springer-Verlag Berlin Heidelberg 2003

- Authentication: The watermark encodes information required to determine that the content is authentic. If the watermark is present, and properly matches the content, the user of the content can be assured that it has not been altered since the watermark was inserted.
- Copy control: The watermark contains information about the rules of usage and copying which the content owner wishes to enforce.
- Secret communication: The embedded signal is used to transmit secret information from one person (or computer) to another, without anyone along the way knowing that this information is being sent. This is the classical application of steganography – the hiding of one piece of information within another.

A digital watermark is a signal added to digital data such that it could be used (1) to identify source of the data or uniquely establish ownership, (2) to identify its intended recipient, and (3) to check if the data has been tampered with. Within each class of applications, there could be variations on the requirement of the watermarking scheme[1]. Therefore, digital watermarks have been proposed as the means for copyright protection of multimedia data on network based digital image distribution structure. However, there are some problems in existing digital watermarking mechanism as the rightful ownerships cannot be resolved by current watermarking schemes alone. Specially, some one can provide counterfeited watermark which can be performed on a watermarked image to allow multiple claims of rightful ownerships[2].

There are several attacks in current digital watermark system. In first case, we can suppose that Alice and Bob use the same digital watermarking technique to watermark and there is one unique decoding scheme to extract the labels embedded in the images. In this case, if a watermarked image contains both Alice and Bob's signatures, how can we decide the originality of this marked image? And we also think another scenario. Let's suppose that Alice and Bob use different watermarking techniques. And Given a watermarked image, Alice can take this image and decode the label using her decoding scheme. Similarly Bob can perform the label extraction process with his decoding scheme. If Alice's decoder indicates that the image belongs to her while Bob's decoder indicates that it is his image, then how can we decide the ownership of this marked image? In these two cases, rightful ownership cannot be resolved by invisible watermarks alone. They present in detail a counterfeit watermarking scheme that allows multiple claims of ownerships.

Undeniable watermark differ from ordinary digital watermark in that the verifier should be unable to distinguish between valid and invalid watermark. But, original watermarker can change it into correct one. If the message in is copyright owner's information, the scheme can be non-invertible. And we can also introduce zero-knowledge based watermarking mechanism. Using zero-knowledge protocols[3], watermarks can be made public enough to be detected yet private enough to be unremovable. i.e., watermarking scheme in which the watermark's presence can be verified without ever revealing enough information to remove the mark.

We first present the overall digital watermark mechanism and review the existing model in section 2. We then consider re-watermarking attack and its possible solution in section 3. And then we propose designated verification process with commitment scheme in section 4. Furthermore, we discuss the proof of ownership in section 5 and conclude this study with the consideration of the future works in final section.

2 Network Based Digital Image Distribution

2.1 Watermark: Function and Mechanism

In general, there are two types of digital watermarks addressed in existing literature, (1) visible and (2) invisible watermarks. These watermarking schemes are designed mainly for two purposes : copyright protection and data authentication. In this paper we shall focus on the applicability of watermarking techniques for one instance of ownership verification, identification of an content's rightful owner(s)[2].

The watermarks embedded in an image have to be recoverable, despite intentional or unintentional modification of the image. They must also be invulnerable to deliberate attempts to forge, remove, or invalidate watermarks. Unfortunately, many of existing schemes have not addressed the ends of invisible watermarking schemes. They instead focused on the means to label an image invisibly and the robustness of the inserted labels against malicious attacks. Without any verification techniques or specification of certain requirements for digital watermark, anyone can claim ownership of any image by using his/her own verification methods. Therefore, it is crucial that watermark for copyright protection be able to be *undeniably verified* with original watermark embedder's verification.

We define in general terms the process of watermark insertion into an image and the use of undeniable watermarks to determine the ownership of a watermarked image. A watermark is inserted into an image, and the decoding process by which a watermark is recovered and then compared to the inserted watermark. We use to denote I an image, W a watermark consisting of a sequence of ownership labels $W = \{ w_1, w_2, ..., w_n \}$ and the watermarked image I^W . E is an encoder function if it takes an image I and a watermark W and generates a new image which is called the watermarked image I^W i.e., $E(I,W) = I^W$.

We can construct both encoding and corresponding decoding processes. One common approach is to embed a watermark $W = \{ w_1, w_2, ..., w_n \}$ into a set of derived features $F(I) = \{ f_1(I), f_2(I), ..., f_n(I) \}$. The embedding process is achieved by an insertion $\oplus$ operation, $f_i^* = f_i \oplus w_i$. The insertion operation has an inverse operation, namely the extraction operation, $\ominus$ that we denote by i.e., $f_i^* \ominus f_i = w_i$.

2.2 Watermark Verification

A watermarking scheme with detection consists of four polynomial-time algorithms GEN_{KEY}, GEN_W, embed function E, and detection function D. GEN_{KEY} and GEN_W are probabilistic and generate a key-pair (k_{emb}, k_{det}) and a watermark W. The algorithm $E(I, W, k_{emb})$ imperceptibly embeds the watermark W into the cover-data I, using the key k_{emb}. This results in stego-data I^W (watermarked version of I). The algorithm $D(I', W, I, k_{det})$ returns a boolean value, which states whether the cover data I' contains the watermark W relative to the reference data I, using key k_{det}. A symmetric watermarking scheme needs the same key k_w) for detection as for embedding. Unkeyed watermarking schemes need no key for embedding or detection. Watermarking schemes whose detect algorithms D do not require the input of reference data I are called blind, in contrast to non-blind schemes.

We can consider three watermark mechanisms for digital data D', which are the (1) watermark(W) casting mechanism, (2) signature(D^*) signing mechanism, and (3) signature hiding and verification mechanism. In this Fig. 2, we can easily understand two mechanisms among these three mechanism. We can embed some mark W on digital data D'. So, it is the common watermark embedding and verification(recovery) mechanism(*mark casting mechanism*). The other one is the cryptographic mechanism as we can attach digital signature D^* on digital data D' for authentication(*signing mechanism*). From these two mechanism, we can consider new hybrid mechanism(*signature hiding and verification mechanism*). As we can think signature as its watermark information, we can hide this signature on its digital data for verifying ownership and authentication. So, we can construct hybrid mechanism on digital watermark verification and authentication system.

3 Attack and Its Solution

3.1 Re-watermarking Attack

In common case, we can assume that there is only one watermark in image for ownership dispute. It is not unreasonable to assume that multiple watermarked versions of the same image may indeed exist. But, if there are multiple watermark in one image, how can we decide the original watermark? And if we cannot find out which so-called *original* image is the *true* original, how can we decide which watermarked version of an image is the *truely watermarked* version in circulation? Bob is free to create as many watermarked versions with his own watermarks embedded after he has reverse-engineered a counterfeit original from Alice's watermarked version. To counter the claims, Bob has to prove the presence of his watermark in these watermarked images and the watermarked version of Alice, in addition to the presence of his watermark in Alice's original. S. Craver solves this problems as follows. He define *non-invertible watermark* and advanced non-quasi-invertible watermark scheme. By using this concepts,

he can do prevent multiple watermark attacks. But there are some unsolved problems in above mechanism.

3.2 Solution: Undeniable Verification

Multiple watermarking problem must be solved easily if we consider it in *verification* process. In case of several attack proposed by S. Craver, we only review its robustness on verification module. Therefore, we can solve proposed attack problems from different directions compared with S. Craver's mechanism. In proposed system, if there are multiple watermarked image, the owner(*pseudo-owner:prover*) can cooperate its verification process. So, in case that the examined watermark (*pseudo-watermark*) can also be matched with the verifier's input message on *true-watermark*, we can trust the pseudo-owner as *really true-owner*.

3.3 Undeniable Property of Digital Watermark

Digital watermarks are verified as authentic by anyone using the verification process. However, this *self-verification* property is unsuitable from many applications such as watermark based commercial digital contents distribution system.

The validity or invalidity of an undeniable watermark can be ascertained by conducting a verification process with original watermarker. If a *confirmation* process is needed, the cooperating original watermarker gives exponentially-high certainty to the verifier that the watermark does correspond to the legal one. We can use *undeniable* scheme in the watermarking process. This is because the confidence measure relies on the fact that the watermark vector can be later extracted, and using a commitment mechanism make designated verifier can certificate the legal owner on embedded watermark information.

Multimedia image company could embed the watermark which they signed using an undeniable scheme. Only someone who had directly purchased the image from that company could verify the watermark and be certain that the image were right. Undeniable watermark verification could also be useful in any situation in which an individual wishes to sign a data anonymously.

3.4 New Verification Mechanism

We can propose a new verification mechanism by using designated verification mechanism as Fig. 1. In first case, the original watermarker can embed his own secret(*true watermark*) on cover data and generate watermarked data. And it is distrubuted to the buyer by using network protocol. And then the authorized user can buy or get this data. However, the attacker can also multiply re-watermark his own fabricated watermark on distributed data. In this case, illegal data can be generated by attacker. So, if this data will be used by unauthorized user, we must verify and proof its legality and correctness on copied data. Therefore, we can suggest a new designated verification mechanism on watermarked data as the authorized user can verify its correctness by doing a undeniable-style proof with original watermarker.

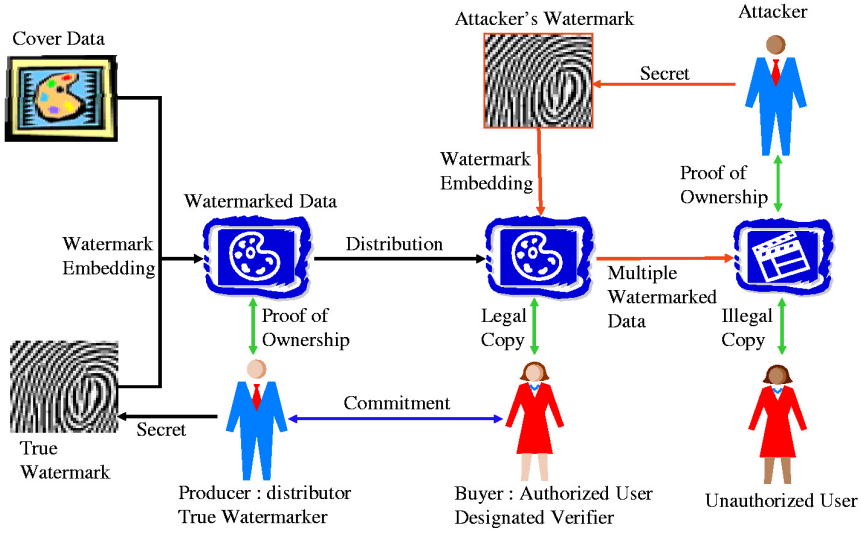


Fig. 1. New Watermark Verification Mechanism.

4 Designated Verification for Network Based Image Distribution

When a content owner sells a digital products, he only want to paying customers to be able to verify the validity of these image. Therefore, the designated verifier mechanism resolves this problems. We can *designate a verifier* when we ascertain that nobody but this participant can be convinded by the proof. With a small change in the confirmation protocol, we can designate verifier and demonstrate watermark verification scheme.

We can define *designated verifier* as follows. Let (P_A, P_B) be a protocol for Alice to prove the truth of the watermark to Bob. In this case, we can say that Bob is a *designated verified* if the following is true: For any protocol (P_A, P'_B, P_C) involving Alice, Bob and Cindy, in which Bob proves the truth of v to Cindy, there is another protocol (P''_B, P_C) such that Bob can perform the calculations of P''_B and Cindy cannot distinguish transcripts of (P_A, P'_B, P_C) from those of (P''_B, P_C).

4.1 Commitment Scheme[4]

A commitment scheme $(com, open)$ for the message space M and commitment space C consists of a two-party protocol com to commit to a value $m \in M$ and a protocol $open$ that opens a commitment. A commitment to a value m is denoted by $com(m, par_{com})$ where par_{com} stands for all public parameters needed to compute the commitment value. To open a commitment com the committer

runs the protocol $open(com, par_{com}, sk_{com})$ where sk_{com} is the secret opening information of the committer.

The security requirements are the *binding(committing)* and *hiding(secrecy)* properties. The first one requires that a dishonest committer cannot open a commitment to another message m' than the one to which he committed and the second one requires that the commitment does not reveal any information about the message m to the verifier.

We require that the commitment scheme has following homomorphic property: Let $com(m_1)$ and $com(m_2)$ be commitments to arbitrary messages $m_1, m_2 \in M$. Then the committer can open $com(m_1) * com(m_2)$ to $m_1 + m_2$ without revealing additional information about the image of $com(m_1)$ and $com(m_2)$. So, we can combine these commitment mechanism on digital watermark verification procedures, if additional trap-door one-way properties are hold.

4.2 Trap-Door Commitment

For hybrid this commitment mechanism on designated verification technique, we consider trap-door primitives, in which we say that c is a *trap-door commitment scheme* if and only if :

1. no polynomial-time machine can, given y_i, find a multiple $(\omega_1, r_1), (\omega_2, r_2)$ such that $c(y_i, \omega_1, r_1) = c(y_i, \omega_2, r_2)$.
2. no polynomial-time machine can, given y_i and $c(y_i, \omega, r)$, output ω.
3. there is a polynomial-time machine that given any quadruple $(x_i, w_1, r_1, \omega_2)$ in the set of possible quadruples finds r_2 such that $c(y_i, \omega_1, r_1) = c(y_i, \omega_2, r_2)$ for the public key y_i corresponding to the secret key x_i.

Using this definitions[4], we can review simple trap-door commitment scheme.

1. Secret extracting(decryption) scheme from watermarked data : $D(\cdot)$.
2. Public embedding(encryption) scheme for watermarked data : $E(\cdot)$.
3. Value to commit to : $\omega \in Range(E)$.
4. Commitment : Alice randomly selects $r \in Range(E)$. And Alice calculates a commitment $c = E(\omega) \oplus E(r)$, where $\oplus$ is a combiner such as XOR.
5. Decommitment : Alice sends Bob (ω, r).

It is a trap-door commitment scheme if arbitrary collisions (ω_1, r_1), (ω_1, r_1) such that $E(\omega_1) \oplus E(r_1) = E(\omega_2) \oplus E(r_2)$ can be found if and only if $D(\cdot)$ is known. By using this mechanism, we can show hot to change the normal watermark verification scheme for undeniable verification. Using this scheme, we can apply it into designated verification for undeniable watermark system.

4.3 Undeniable Verification with Commitment

Let $(com, open)$ be a secure commitment scheme. A watermark verification protocol for the (GEN_{KEY}, GEN_W, E, D) is a undeniable proof protocol between

a *prover* P and a *verifier* V: The common protocol input of P and V is the stego-data I^W, $com(I)$, $com(W)$, $com(k_w)$, i.e., commitments on the watermark, the reference data and the detection key respectively, as well as the public parameters $par_{com} = (par^W_{com}, par^I_{com}, par^{k_w}_{com})$ of these commitments. The private input of the *prover* is the secret opening information of these commitments $sk_{com} = (sk^W_{com}, sk^I_{com}, sk^{k_w}_{com})$. P outputs a boolean value to the verifier, stating whether to accept the proof or not.

We can think a *non-interactive* designated verifier proof of an undeniable watermark. Such a scheme can diminish the difference between public watermark and undeniable watermark, in that it limits who can verify it without help from the prover, but does not necessary condition. We can use commit protocol. Therfore, the common three steps can be changed into one single step.

1. Generating and Proprocessing a Key/Data
   - Generate *prover* Alice's secret key/public key pair : x_A and $y_A \equiv g^{x_A} \bmod p$.
   - Generate *verifier* Bob's secret key/public key pair : x_B and $y_B \equiv g^{x_B} \bmod p$. (In this step, we use key generation function GEN_{KEY}).
   - Using hash function $h(\cdot)$, generate $m = h(I)$ on cover data I
   - Alice selects $\omega, r, t \in Z_q$.
2. Generating and embedding a Watermark.
   - The verifier sends hashed value m to prover.
   - The prover generate a watermark W using his own secret key $x_A : W \equiv m^{x_A} \bmod p$. (In this step, we use watermark generation function GEN_W and we can tell the prover as a *watermarker or signer*).
   - The prover embed this watermark W into cover data I. Then the watermarked data I^W will be generated. (In this step, we use embedding function E).
3. Constructing a Proof.
   - The prover calculates $c = g^{\omega} y_B^r \bmod p$, $G \equiv g^t \bmod p$, $M \equiv m^t \bmod p$, and $d \equiv t + (x_A \cdot \omega) \bmod q$.
   - And the prover sends commitment (ω, r, G, M, d) to the verifier.
4. Verifying a Proof.
   - The verifier extract the embedded pseudo-watermark W' from the I'. (In this step, we use detection function D).
   - Using the commitment, the verifier calculates $c \equiv g^{\omega} y_B^r \bmod p$ and $G \cdot y_A^{\omega} \equiv g^t \cdot g^{x_A \cdot \omega} \bmod p \equiv g^d \bmod p$.
   - On the extracted pseudo-watermark, the (designated) verifier calculate $M \cdot W^{\omega} \equiv m^d \bmod p$.

5 Discussion: Zero Knowledge Proof of Ownership

Existing mechanism suffer from the watermark verification process: the disputing parties have to disclose information, which is necessary to detect the watermark, to the dispute-resolver. However, this information is in most cases also sufficient

to remove attack and re-watermarking attack from this opened watermark(the disputed data). As a solution, zero-knowledge watermark verification eliminates this security risk, because it enables a prover to prove to an untrusted verifier that a certain watermark is present in watermarked data without *revealing any information about the watermark, the reference data and the detection key.* We can combine this properties with proof of owner procedures.

If ownership is challenged, both challenger and owner has to disclosed the proof of their ownership. So ownership is an issue, though people are seldom aware of it. If you are not able to prove you are the owner of something in the first place, you cannot ask for any payments concerning any kind of copyright. It is also important in digitized watermark system.

In proposed scheme, we suggest undeniable verification procedure. And it can be applicable to the proof of copyright or onwership of some resources by trusted third party T or certificate authority CA. The copyright hodler H can proof his own privilege as follows. We can considers H to be the owner of a work I' if and only if the following conditions hold:

- H has previously registered a new work I' on trusted third party T
- I is similar to I' and
- I' is the first registered work to which I is similar.

Therefore, we can construct ownership proofing function from the undeniable verification system. And the detailed mechanism can also be suggested by combining identification module with watermark verification. So, the real owner can be determined by similar verification procedure proposed in this study.

6 Conclusions

Digital watermarks have been proposed as the means for copyright protection of multimedia data. However, there are some problems in existing digital watermarking mechanism as the rightful ownerships cannot be resolved from diverse attacks such as re-watermarking. In this study, we solve multiple watermarking problem as we consider it by the *designated verification* process. In proposed system, if there are multiple watermarked image, the original owner can cooperate its verification process. So, in case that the examined label*pseudo-watermark* can also be matched with the verifier's input message as a *true-watermark*, we can trust the verifier*pseudo-owner* as real *true-owner* by the proposed designated verification mechanism.

As there are some problems in existing digital watermarking mechanism, the rightful ownership cannot be resolved by existing watermarking schemes alone. Specially, some one can provide counterfeited watermark which can be performed on a watermarked image to allow multiple claims of rightful ownerships. Therefore, we proposed undeniable verification scheme in the watermarking process on network based secure image distribution. And using designated verifier proofing schemes, watermark can be verified by commitment methods. In the future, we can consider zero-knowledge based designated verification mechanism when the

confirmer of a watermark verify the validity of the watermark owner without revealing the secret of true watermark information of originator.

References

1. Andre Adelsbach, Birgit Pfitzmann, Ahmad-Reza Sadeghi, "Proving Ownership of Digital Content", 3rd International Information Hiding Workshop (IHW '99), LNCS 1768, Springer-Verlag, 117–133, 1999.
2. Scott Craver, Nasir Memon, Boon-Lock Yeo, Minerva M. Yeung, "Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitations, Attacks, and Implications", IEEE Journal on Selected Areas in Communications 16/4 (1998) 573–586.
3. Andre Adelsbach, Ahmad-Reza Sadeghi, "Zero-Knowledge Watermark Detection and Proof of Ownership", 4th International Information Hiding Workshop (IHW '01), LNCS 2137, Springer-Verlag, 273–288, 2001.
4. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo, "Designated verifier proofs and their applications", Advances in Cryptology - EUROCRYPT 96, LNCS 1070, Springer-Verlag, 143–154, 1996.
5. T. P. Pedersen, "Distributed provers with applications to undeniable signatures", Advances in Cryptology, Proceedings of Eurocrypt91, LNCS 547, pages 221–242, 1991.
6. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithm", IEEE Transactions on Information Theory, Vol. IT-30, No. 4, pp. 469–472, 1985.
7. I. B. D. J. Boyar, D. Chaum and T. P. Pedersen, "Convertible undeniable signatures", Advances in Cryptology, Proceedings of Crypto90, pages 189–205, 1991.
8. Alfred J. Menezed, Paul C. van Oorschot, Scott A. Vanstone (ed.), "Handbook of Applied Cryptography", CRC Press, 1996.
9. B. Schneier, "Applied Cryptography", Second Edition, Wiley, 1996.
10. D. Chaum, "Undeniable signatures", Advances in Cryptology, Proceedings of CRYPTO89, pages 212–216, 1989.
11. W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, IT-22(6):472-492, November 1976.

A Parallel Loop Self-Scheduling on Extremely Heterogeneous PC Clusters

Chao-Tung Yang and Shun-Chyi Chang

Department of Computer Science and Information Engineering
Tung-Hai University
Taichung, Taiwan 407, R.O.C.
ctyang@mail.thu.edu.tw

Abstract. Cluster system is viable and less expensive alternative to SMP. However, the approaches to deal with scheduling and load balancing on heterogeneous cluster computer system are not mature. Self-scheduling schemes which are suitable for parallel loops with independent iterations on heterogeneous cluster computer system have been designed in the past. However, these schemes, such as *FSS*, *GSS* and *TSS*, can not achieve load balancing in extremely heterogeneous environment. In this paper, we propose a heuristic approach to solve parallel regular loop scheduling problem on extremely heterogeneous PC cluster environment.

1 Introduction

Parallel computers are becoming increasingly widespread, nowadays many of these parallel computers are no longer shared-memory multiprocessors, but rather follow the distributed memory model for scalable. These systems may consist of homogeneous workstations, where all the workstations have processors, memory and caches with exactly the identical specifications. According to Moore's law, CPU clock will double in 18 months. As the law still works today, to build clusters consisting of extremely different computer performance becomes demanding. However, more and more systems are now composed of a number of heterogeneous workstations clustered together, where each workstation may have different CPU performance capabilities, different amounts of memory and caches, and even different architectures and operating systems.

To exploit the potential computing power of cluster computers, an important issue is how to assign tasks to computers so that the computer loads are well balanced. That is how to assign the different parts of a parallel application to the computing resources to minimize the overall computing time and to efficiently use the resource. Loops often comprise a large portion of a program's parallelism. An efficient approach to extract potential parallelism is to concentrate on the parallelism available in the loops. A loop is called a DOALL loop if there is no cross-iteration dependence in the loop; i.e., all the iterations of the loop can be executed in parallel. If all the iterations of a DOALL loop are distributed among different processors evenly, a high degree of parallelism can be exploited. Parallel loop scheduling is a method that attempts to evenly schedule a DOALL loop on multiprocessor systems.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1079–1088, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In homogeneous environment, workload can be partitioned equally to each working computer, but in heterogeneous environment, this method will not work. Some researches were proposed to solve parallel loop scheduling problems on heterogeneous cluster environments by using self-scheduling schemes. These self-scheduling schemes work well in moderate heterogeneous cluster environment but not in extremely heterogeneous environment where the performance difference between the fastest computer and the slowest computer is large.

In this paper, we will propose a loop scheduling based on self-scheduling scheme to approach load balancing on extremely heterogeneous cluster. The experimental results are conducted on a PC Cluster with six nodes and the fastest computer is 7.5 times faster than the slowest ones in CPU-clock. In our experiments, assigning 80% workload corresponding to the CPU clock, and 20% workload using traditional self-scheduling will achieve good load balancing.

The rest of the paper is organized as follows. In section 2, a brief overview of self-scheduling is given. Section 3 states our approach and reports the experiments. Finally, the conclusion remarks are given in section 4.

2 Background

The major source of parallelism in a program is loops. If the loop iterations can be distributed to different processors as evenly as possible, the parallelism within loop iterations can be exploited. Loops can be roughly divided into four kinds, as shown in Figure 1: uniform workload, increasing workload, decreasing workload, and random workload loops. They are the most common ones in programs, and should cover most case. These four kinds can be classified two types: regular and irregular.

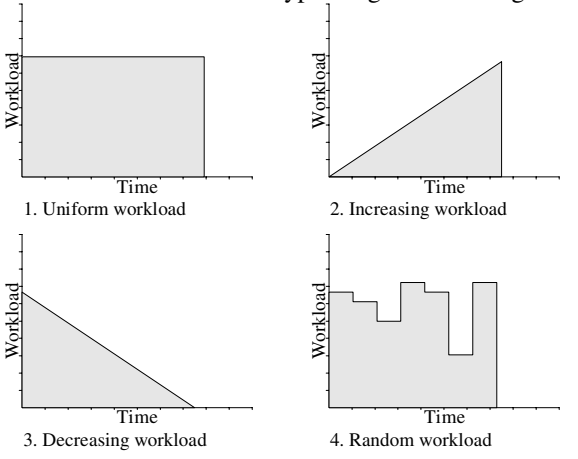


Fig. 1. Four kinds of loops.

In a parallel process system, two kinds of parallel loop scheduling decisions can be made either statically at compile-time or dynamically at run-time.

Static scheduling is usually applied to uniformly distributed iterations on processors [6]. It has the drawback of creating load imbalances when the loop style is not uniformly distributed; when the loop bounds cannot be known at compile-time; when system is heterogeneous; or when locality management cannot be exercised. In contrast, dynamic scheduling is more appropriate for load balancing; however, the runtime overhead must be taken into consideration. In general, parallelizing compilers distribute loop iterations by using only one kind of scheduling algorithm, either static or dynamic.

2.1 Static Scheduling

Traditional static scheduling [6] is applied when each loop iteration takes roughly the same amount of time, and the compiler knows how many iterations will be run and how many processors are available for use at compile-time. It has the advantage of incurring the minimum scheduling overhead, but load imbalances may occur. These static scheduling schemes including Block Scheduling, Cyclic Scheduling, Block-D Scheduling, Cyclic-D Scheduling… etc [6]. Nevertheless, these scheduling schemes were unsuitable in heterogeneous environment.

Theoretically, workload can be partitioned according to their computer performance. Unfortunately, in heterogeneous system, it is difficult to evaluate each computer performance. Intuitively, CPU clock speed may be a good evaluation value. But it seems not enough. Many factors affect computer performance, such as the performance capability of the CPU, the amount of memory available, the cost of memory access, the communication medium between processors… etc [5]. Bohn and Lamont try to evaluate the performance of computer in compiler-time [4]. In their experiment, HINT is a good benchmark. It evaluates processor and memory performance for any data type and returns a single value, "QUIPS". Bohn and Lamont declared "QUIPS" can present the computer performance. It has the advantage of all computers being working computer - no control computer is needed. But, HINT requires hours to execute. It means this way will not be scaling well. It takes a long time to add one more computer and if we want to change the peripheral, for example to replace RAM from PC100 to PC133, we might have to rerun HINT.

2.2 Dynamic Scheduling

Dynamic scheduling adjusts the schedule during execution and is especially suitable whenever the number of iterations is uncertain or each iteration may take a different amount of time. Although it is more suitable for load balancing between processors, runtime overhead is the cost.

We use master/slave computation patterns to model problems, i.e., the master coordinate data distribution to the slaves, which perform computations and transmit the results back to the master. The master is not responsible for workload, the idle slave requests to the master for new loop iterations, and no communication occurs between slaves. How many iterations that a slave should be assigned is a critical issue. Improper assignment will cause bad system performance.

Self-scheduling is a large class of adaptive/dynamic centralized loop scheduling schemes. In a common self-scheduling scheme, p denotes the number of processors, N denotes the total iteration and $f()$ is a function to produce the chunk-size at each step. At the i-th scheduling step, the master computes the chunk-size Ci and the remaining number of tasks R_i,

$R_0=N$, $C_i=f(i,p)$, $Ri=R_{i-1}-C_i$

where $f()$ possibly has more parameters than just i and p, such as R_{i-1}. The master assigns C_i tasks to an idle slave and the load imbalancing will depend on the execution time gap between t_j, for $j=1,\cdots,p$ [7].

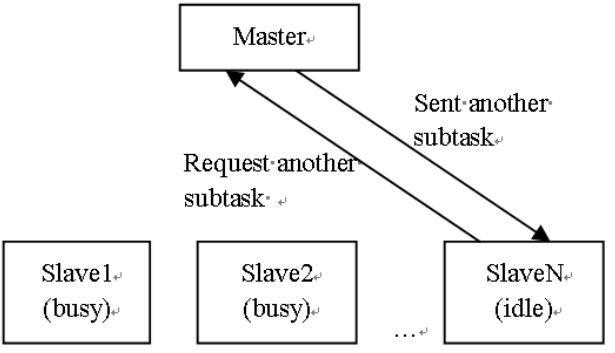


Fig. 2. A master/slave model.

Different ways to compute C_i have given rise to different scheduling schemes. The most notable examples are as following:

Pure Self-Scheduling (*PSS*). This is the easiest and most straightforward dynamic loop scheduling algorithm [8]. Whenever a processor is idle, an iteration is assigned to it. This algorithm achieves good load balancing but also induces excessive overhead.

Chunk Self-Scheduling (*CSS*). Instead of assigning one iteration to an idle processor as in self-scheduling, *CSS* assigns k iterations each time, where k, called the chunk size, is fixed and must be specified by either the programmer or the compiler [8]. When the chunk size is one, this scheme is pure self-scheduling, as discussed above. If the chunk size is set to the bound of the parallel loop equally divided by the number of processors, the scheme becomes static scheduling. A large chunk size will cause load imbalancing while a small chunk is likely to produce too much scheduling overhead. For different partitioning schemes, we adopted *CSS*(s), which is a modified version of *CSS*, where s means the size of chunks.

Guided Self-Scheduling (*GSS*). This algorithm can dynamically change the number of iterations assigned to each processor [2]. More specifically, the next chunk size is determined by dividing the number of remaining iterations of a parallel loop by the number of available processors. The property of decreasing chunk size implies an effort is made to achieve load balancing and to reduce the scheduling overhead. By assigning large chunks at the beginning of a parallel loop, one can reduce the frequency of communication between master and slaves. The small chunks at the end of a loop partition serve to balance the workload across all the working processors.

Factoring. In some cases, *GSS* might assign too much work to the first few processors, so that the remaining iterations are not time-consuming enough to balance the workload. This situation arises when the initial iterations in a loop are much more time-consuming than the later iterations. The Factoring algorithm addresses this problem [1]. The assignment of loop iterations to working processors proceeds in phases. During each phase, only a subset of the remaining loop iterations (usually half) is divided equally among the available processors. Because Factoring assigns a subset of the remaining iterations in each phase, it balances loads better than *GSS* does when the computation times of loop iterations vary substantially. In addition, the synchronization overhead of Factoring is not significantly larger than that of *GSS*.

Trapezoid Self-Scheduling (*TSS*). This approach tries to reduce the need for synchronization while still maintaining a reasonable load balance [3]. *TSS*(N_s, N_f) assigns the first N_s iterations of a loop to the processor starting the loop and the last N_f iterations to the processor performing the last fetch, where N_s and N_f are both specified by either the programmer or the parallelizing compiler. This algorithm allocates large chunks of iterations to the first few processors and successively smaller chunks to the last few processors. Tzen and Ni proposed *TSS(N/2P, 1)* as a general selection. In this case, the first chunk is of size *N/2p*, and consecutive chunks differ in size $N/8p^2$ iterations. The difference in the size of successive chunks is always a constant in *TSS* whereas it is a decreasing function in *GSS* and in Factoring.

Table 1 shows the different chunk sizes for a problem with the number of iteration N=1000 and the number of processor p=4.

Table 1. Sample partition sizes

Scheme	Partition size
PSS	1,1,1,1,1,1,1…
***CSS*(125)**	125,125,125,125,125,125,125,125
FSS	125,125,125,125,63,63,63,63,31,…
GSS	250,188,141,106,79,59,45,33,25,…
TSS	125,117,109,101,93,85,77,69,61,…

3 Our Approach and Experiment

In extremely heterogeneous environment, each slave computer in a cluster system owns extremely different performance. In this condition, an additional slave computer may not lead to a better performance since these known self-scheduling schemes partition size of loop iteration according to formula instead of computer performance. In *FSS*, for example, every slave gets a size of *N/2p* workload. If the performance difference between the fastest computer and the slowest computer is larger than *N/2p*, then load imbalance happens. Furthermore, dynamic load balancing approach should not be aware of the run-time behavior of the applications before execution. But in *GSS* and *TSS*, to achieve good performance, computer performance of each computer in the cluster has to be in order in extremely heterogeneous environment, which is not very applicable.

A combination of different machine types is used to test the behavior of these approaches in a heterogeneous computing environment, and the matrix multiplication is chosen as the test application to get a heuristic result due to its regular behavior. This experiment included four computers. One of them is assigned as master using some self-scheduling approach to partition size of loop iteration. The master is a PC with 300 MHz CPU speed and 208MB physical memory. Three slaves are PCs, respectively, with 1.5GHz CPU speed and 256MB physical memory, 233 MHz CPU speed and 96MB physical memory, and 200MHz CPU speed and 64MB physical memory. The slaves are added into the system sequentially in this order. We respectively use *TSS* and *FSS* approach to test matrix multiplication with different problem sizes from 512*512, 1024*1024, to 2048*2048 by floating point operations. Table 2 shows our experiment result. Note that just one slave in Table 2 means that all work is done by the fastest computer only. It shows the system having two slave computers gets worse performance than having only one slave computer. An additional PC does not help the performance.

Table 2. The result performance of number of slaves in extremely heterogeneous environment.

No. of slaves	Execution time (*TSS*)			Execution time (*FSS*)		
	512*512	**1024*1024**	**2048*2048**	**512*512**	**1024*1024**	**2048*2048**
1	0'12"066	1'44"357	17'12"483	0'12"136	1'44"688	17'11"402
2	0'17"520	2'49"652	19'34"016	0'18"371	3'16"561	23'48"723
3	0'13"339	1'53"202	16'30"651	0'14"543	2'00"491	16'36"007

For the programs with regular loops, intuitively, we may want to partition problem size according to their CPU clock in heterogeneous environment. However, the CPU clock is not the only factor which affects computer performance. Many other factors also have dramatic influences in this aspect, such as the amount of memory available, the cost of memory accesses, and the communication medium between processors… etc[5]. Using this intuitive approach, the result will be degraded if the performance prediction is inaccurate. A computer with largest inaccurate prediction will be the last one to finish the assigned job.

We propose to partition problem size in two stages. At first stage, partition the α% of workload according to their performance weighted by CPU clock. In the way, the communication between master and slaves can be reduced efficiently. At second stage, partition following (100-α)% of workload according to known self-scheduling scheme. In the way, load balancing can be archived. With this approach, we don't need to know the real computer performance. The computer finishing its job early gets another larger job. The parameter α should not be too small or too big. In former case, the dominate computer will not finish its work. In the latter case, the dynamic scheduling strategy is rigid. In both cases, good performance can not be attained. An appropriate α value will lead to good performance and reduce communication times. Many α values are applied to the experiments, and α=80 result in the best performance.

```
Algorithms MASTER and SLAVE in pseudo code:
Module MASTER
/* performs task scheduling and load balancing */
Initialization
Gather CPU clock of all slave computers
r=0;
For (i=1; i<number_slave; i++) {
  Partition α% of loop iteration corresponding
    their CPU clock speed and sent data to slave
  r++;
}
Partition (100-α)% of loop iteration into task queue
using some known self-scheduling scheme
Probe if some data in
Do{
  Distinguish source and receive data
  If task queue not empty
    Sent other data to this idle slave
    r--;
  Else
    sent TAG=0 to this idle slave
}while (r>0);
Finalization
END MASTER

Module SLAVE /* worker */
Initialization
Sent my CPU clock to master
Probe if some data in
While (TAG>0){
  Receive initial solution and size of subtask work and
  compute to find solution
  Send the result to master
  Probe if some data in
}
Finalization
END SLAVE
```

3.1 The Experiments and Results

The approach is applied in an extremely heterogeneous environment which includes six computers. One of them is assigned as the master. The master is a PC with 300 MHz CPU and 208MB physical memory. Two of the slaves are PCs with 200 MHz CPU and 64 MB physical memory. The other three slaves are PCs, respectively, with 233 MHz CPU and 96MB physical memory, 533MHz CPU and 128MB physical memory, and 1.5GHz CPU and 256MB physical memory. Those computers may own

various NIC and cost of memory access, regarding as part of computer performance. SWAP occurs in some computers. If SWAP does not occur often, this will not affect the result.

Table 3 and Figure 3 show the result in α=80. The column name "None" stands for "none load-balancing" and workload be partitioned just by CPU clock. "*FSS*/80" stand for "α=80, and remainder use *FSS* to partition" and so on.

Table 3 shows using this approach in 2048*2048 matrix multiplication will get 25%, 30%, 21% performance improvement than *FSS*, *GSS*, *TSS* respectively. Note that in extremely heterogeneous environment, known self-scheduling get worse performance than schemes partitioning workload merely according to the CPU clock.

Table 3. The result of our approach in extremely heterogeneous environment (α=80).

	None	***FSS***	***FSS*/80**	***GSS***	***GSS*/80**	***TSS***	***TSS*/80**
512*512	8.1	9.8	7.0	10.0	7.5	9.1	7.6
1024*1024	74.9	98.7	56.6	115.5	59.4	71.6	63.0
2048*2048	598.6	678.1	509.3	732.6	509.0	666.1	521.3

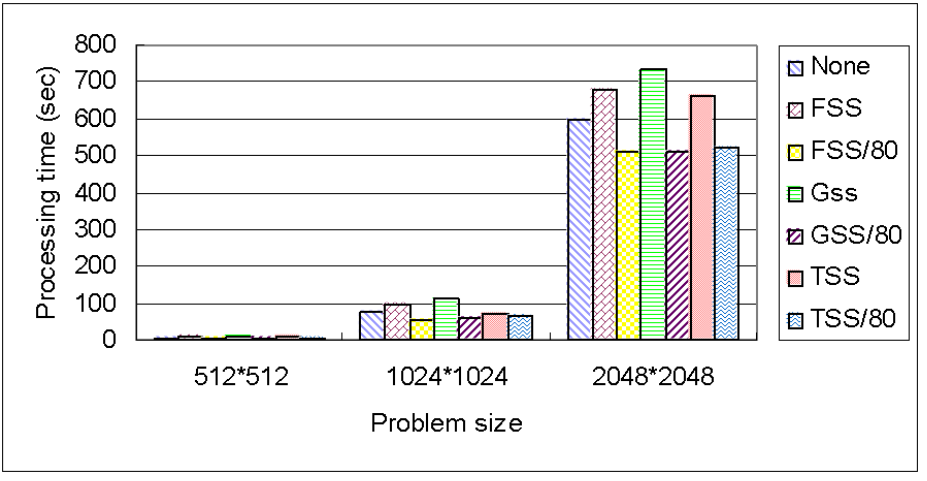


Fig. 3. The result of our approach in extremely heterogeneous environment (α=80)

Our approach is also useful in moderate environment. Following experiment includes five computers. One of them is assigned as the master. The master is a PC with 900 MHz CPU and 256MB physical memory. Two slaves are PCs with 600GHz CPU speed and 256MB of physical memory. The other two slaves are PCs, respectively, with 975 MHz CPU and 512MB physical memory, 900MHz CPU and 256MB physical memory. Table 4 and Figure 4 present the result when the approach is applied to matrix multiplication, as α = 80. It shows that our approach has equal, even better, performance than the known self-scheduling.

Table 4. The result of our approach in moderate heterogeneous environment (α=80).

	None	*FSS*	*FSS*/80	*GSS*	*GSS*/80	*TSS*
1024*1024	64.6	59.1	59.1	59.1	59.0	65.2
2048*2048	520.8	477.3	474.4	477.6	474.3	505.8
3072*3072	1792.8	1720.9	1711.0	1715.8	1714.7	1792.1

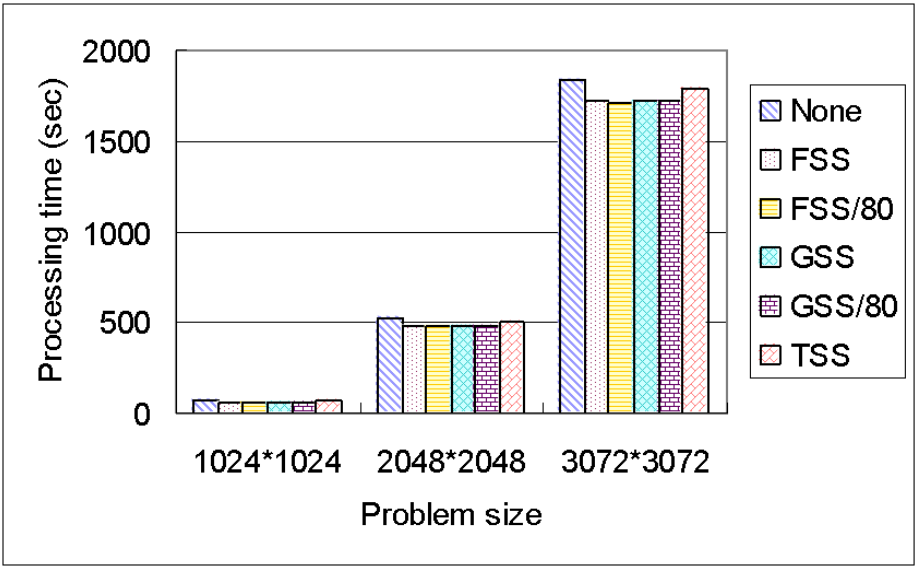


Fig. 4. The result of our approach in moderate heterogeneous environment (α=80).

4 Conclusion and Future Work

In extremely heterogeneous environment, known self-scheduling schemes cannot achieve good load balancing. In this paper, we propose an approach to partition loop iterations and achieve good performance in such an environment, that is that partitioning the 80% of workload according to their performance weighted by CPU clock and the rest 20% of workload according to known self-scheduling. In case of matrix multiplication with 2048*2048 problem size, our approach can obtain as much as 30% performance improvement than *GSS*. Therefore, our approach is suitable in all applications with predictable loops.

Fann, Yang, Tseng and Tsai propose a knowledge-based approach to solving loop-scheduling problems [9]. A rule-based system, called *IPLS*, is developed by combining a repertory grid and an attribute ordering table to construct a knowledge base. *IPLS* chooses an appropriate scheduling algorithm by inferring some features of loops and assigning parallel loops to multiprocessors to achieve significant speedup. However, this system is based on UMA architecture. In near future, we will migrate

IPLS to cluster architecture. Also, we will solve parallel loop scheduling problems with unpredictable loops on extremely heterogeneous PC clusters and integrate our approach into the new *IPLS*.

References

1. S. F. Hummel, E. Schonberg, L. E. Flynn, "Factoring, a Scheme for Scheduling Parallel Loops," Communications of the ACM, Vol 35, No 8, Aug. 1992.
2. C. D. Polychronopoulos and D. Kuck, "Guided Self-Scheduling: a Practical Scheduling Scheme for Parallel Supercomputers," IEEE Trans. on Computers, Vol 36, Dec. 1987, pp 1425–1439.
3. T. H. Tzen and L.M. Ni, "Trapezoid Self-Scheduling: A Practical Scheduling Scheme for Parallel Compilers," IEEE Trans. on Parallel and Distributed Systems, Vol 4, No 1, Jan. 1993, pp 87–98.
4. Christopher A. Bohn, Gary B. Lamont, "Load Balancing for Heterogeneous Clusters of PCs," Future Generation Computer Systems 18 (2002) 389–400.
5. E. Post, H. A. Goosen, "Evaluating the Parallel Performance of a Heterogeneous System," in the Proceedings of HPCAsia2001.
6. H. Li, S. Tandri, M. Stumm and K. C. Sevcik, "Locality and Loop Scheduling on NUMA Multiprocessors," in Proceedings of the 1993 International Conference on Parallel Processing, Vol. II, 1993, pp. 140–147.
7. A. T. Chronopoulos, R. Andonie, M. Benche and D.Grosu, "A Class of Loop Self-Scheduling for Heterogeneous Clusters," in Proceedings of the 2001 IEEE International Conference on Cluster Computing, pp. 282–291
8. P. Tang and P. C. Yew, "Processor self-scheduling for multiple-nested parallel loops," in Proceedings of the 1986 International Conference on Parallel Processing , 1986, pp. 528–535.
9. Yun-Woei Fann, Chao-Tung Yang, Shian-Shyong Tseng, and Chang-Jiun Tsai, "An intelligent parallel loop scheduling for multiprocessor systems," Journal of Info. Science and Engineering - Special Issue on Parallel and Distributed Computing, vol. 16, no. 2, pp. 169–200, March 2000.

Robust Reference Point Detection Using Gradient of Fingerprint Direction and Feature Extraction Method

Junbum Park and Hanseok Ko

Department of Electronics Engineering, Korea University
5Ka-1, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea.
jbpark@ispl.korea.ac.kr
hsko@korea.ac.kr

Abstract. A novel reference point detection method is proposed by exploiting the GPM(Gradient Probabilistic Model) that captures the curvature information of fingerprint texture. The detection of reference point is accomplished through searching and locating the points of occurrence of the most evenly distributed gradient in probabilistic sense. We also propose a novel filterbank method to improve shortcoming of existing filterbank method in verification part. Existing filterbank method can lose the discerning attributes because the sectors of the outer band from the reference point are larger in size than those of the inner bands. Such shortcomings of the filterbank method are resolved by maintaining the attribute regions to equal size.

1 Introduction

Recent surge in the interest of personal identification by means of biometrics is gaining even more attention due to the needs of reliable security system that's both affordable in cost while enabling user convenience. Biometric solution is particularly attractive since the traditional hand-carrying ID card has the risk of being lost or stolen and that an imposter can easily forge their way to get access. Among the many possible biometric schemes, the fingerprint is one of the most reliable methods for the identification of individuals because there are no two persons having the same fingerprint and for each fingerprint, it remains unchanged over a lifetime and is easy to acquire. For the fingerprint verification, the most technically critical elements are the detection of a reliable reference point and the extraction of key features based on the reference point detected. Fig. 1 depicts the usual flow of data processing required to perform the fingerprint verification. Poincare index has recently been by far the most widely used approach in detecting the target reference point (e.g core point or delta point)[1][2][3]. It essentially looks for the position of occurrence of high gradient components in the texture field. However, because the method is conditioned on detecting the variations in gradient components, it is prone to errors induced by cuts in a flow of finger print ridge contour or foreign materials coated on the finger surface. As a result, the procedure is sensitive to the quality of captured image and thus a detailed preprocessing is required. Thus, while it is well suited for detecting the reference point in fingerprints of loop type, double-loop type or whole type, it

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1089–1099, 2003.
© Springer-Verlag Berlin Heidelberg 2003

performs poorly on the arch types having vague gradient. To remedy the weakness over the arch type textures, we employ the gradient direction information (0,45,90,135) in the form of gradient probability distribution. In particular, we determine the gradient probability distribution over the surface texture of the fingerprint and then search the point of occurrence of equal probability in the four principal directions. Through finding the spot of uniform density, we attain the robustness in the verification performance on various fingerprint types including those of the arch type, while suppressing the performance sensitivity to the image quality. This paper is composed as the following sections, we will present the details of our reference point detection algorithm(GPM) and feature extraction algorithm using new filterbank method. Section 2 presents our reference point detection algorithm. In Section 3, feature extraction algorithm is presented. Moreover, In Section 4, we present our experimental results obtained by the employed semiconductor based sensor that show the improvements in terms of accuracy and computational load under various noisy scenarios. The conclusions and future research directions are presented in Section 5.

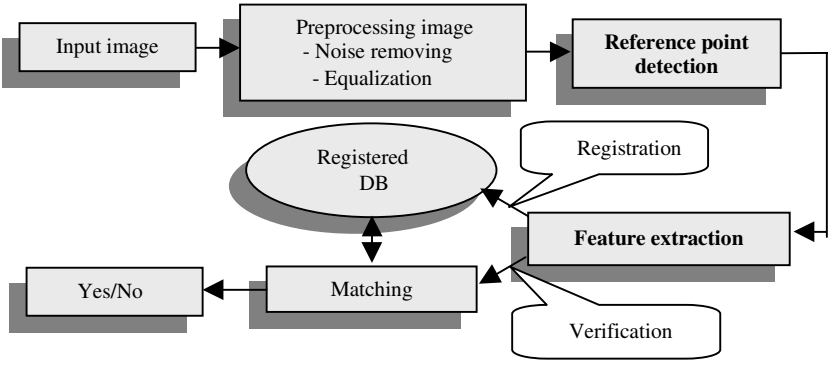


Fig. 1. Typical fingerprint verification procedure

2 Reference Point Detection Using GPM(Gradient Probabilistic Method)

In this section, we propose GPM that uses the directional information and to expand the coverage of the fingerprint types that includes the arch-type. The directional information can be attained by extracting 4 directional data (0, 45, 90, 135) in fingerprint by taking a gradient over the entire image and by computing the directional probability at each of the directional angles in 50-by-50 raster scanning windows. We then search for the region where the direction probability distribution is most flat (uniform density). In addition, because the method uses the directional distribution information, no detailed preprocessing is required as it is not as sensitive to precise image events. The Poincare index method requires costly and detailed

preprocessing such as connecting ridge lines when the lines are cut by noise. The procedure of the proposed GPM is shown below Fig. 2.

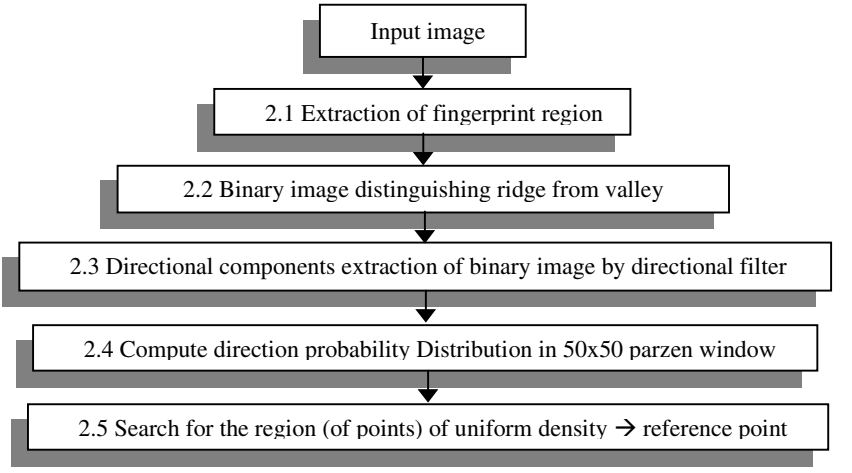
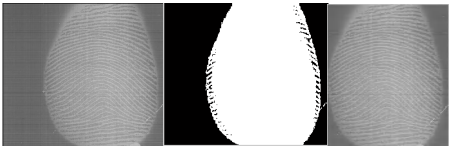


Fig. 2. Reference Point Detection Algorithm using GPM

2.1 Extraction of Fingerprint Region

Fingerprint region is obtained by dilation/erosion operation(DE). To obtain the fingerprint region(FR)(Fig. 3(c)), we compute mean value of input image(Fig. 3(a)). After binary image is obtained by (Eq.1), we extract fingerprint region using dilation and erosion operation.

$$binary \quad image \ (i, j) = \begin{cases} 1, & ipnut \quad image \ (i, j) > threshold \quad (mean \ + \alpha) \\ 0, & Otherwise \end{cases} \tag{1}$$



(a) Input image (b) DE image (c) FR image

Fig. 3. Fingerprint region extraction image

2.2 Image Distinguishing Ridge from Valley

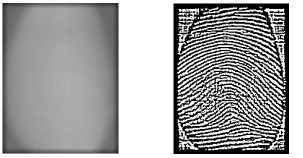
To obtain the binary image, we divide the method into using global threshold and local threshold. The fingerprint image used can be either bright or dark because of the variations in illumination from the sensor. Thus, to blindly set the global threshold is not appropriate. We, therefore, use the local threshold to distinguish the ridge from

valley. The Local threshold takes the average image of fingerprint by convolving a circle of 17×17 size to the captured image. The binary image (Fig. 4(b)) distinguishing ridge from valley is then obtained by following Eq. 2.

$$binary\ \ image\ (i,j) = \begin{cases} 1, & FR\ \ image\ (i,j) > average\ \ \ image\ (i,j) \\ 0, & Otherwise \end{cases} \tag{2}$$

$$average\ \ \ image\ (i,j) = FR\ \ image\ (i,j) * unit\ \ circle\ \ \ filter$$

$$unit\ \ \ circle\ \ \ \ filter\ \ = \left\{ (x-i)^2 + (y-j)^2 = r^2 \right\}$$



(a) Average image (b) Binary image

Fig. 4. Average image and distinguishing ridge from valley

2.3 Direction Component Extraction of Fingerprint

Direction information of a fingerprint is divided into four components. Namely, direction components of 0,45,90,135 degrees are extracted Fig. 5 and the size of direction filter is set at 5×5 masking window.

(a) 0° **filter** **(b)** 45° **filter** **(c)** 135° **filter** **(d)** 90° **filter**

Fig. 5. Direction filter of 5×5 size

Moreover, the images in Fig. 6 are obtained by operating convolution between direction filters of Fig. 5 and the binary image of Fig. 4(b).

(a) 0° **(b)** 45° **(c)** 135° **(d)** 90°

Fig. 6. Extracted binary direction components

2.4 Reference Point Detection

In Fig. 6, we search for the regions where all four direction probabilistic distributions are the most equal using a Parzen window[4]. The point of uniform density can be extracted by counting the sampling numbers of each direction component in a fixed window. Parzen window is a good approximating approach that establishes the probability distribution at each direction components in the window prescribed by the following procedure.

1) Set Parzen window of fixed size $N \times N$.
2) Perform convolutions of each direction components shown in Fig. 6 and the Parzen window. Count the number of samples of each direction component as the window is raster scanned across the entire image.
3) Select the point of the most equal probability in all direction. In step 1, the size of window is set at 50×50 by experiment. The Parzen window contains 2005 pixels. Computation of the probability distribution in step 2 is denoted in Eq. 3.

$$P(\theta_i \mid I(x,y)) \Rightarrow \sum_i P(\theta_i \mid I_i(x,y)) = 1, \quad P(\theta_i) = \frac{count(I_i(x+\Delta x, y+\Delta y)}{S} \tag{3}$$

where, $I_i(x,y)$ is coordinate*(x,y)* of all pixels in image, $P(\theta_i)$ is probability of each directional component in coordinate*(x,y)*, S is an area of Parzen window and $i = 0^\circ, 45°, 90°135°$. After direction probability of all pixels in image is computed, we use a relative entropy concept to find the point of the most equal probability. If A, B are distribution, a relative entropy(H) defined by

$$H(A(x+\Delta x, y+\Delta y), B(x+\Delta x, y+\Delta y)) = \left| \sum_i a_i(x+\Delta x, y+\Delta y) \log \frac{a_i(x+\Delta x, y+\Delta y)}{b_i(x+\Delta x, y+\Delta y)} \right| \tag{4}$$

where, $H(A,B)$ is a relative entropy between two distribution A and B, $a_i(x+\Delta x, y+\Delta y)$ is PMF(Probability Mass Function) really computed in all position $(x+\Delta x, y+\Delta y)$ of i-direction and $b_i(x+\Delta x, y+\Delta y)$ is PMF(threshold) in all position $(x+\Delta x, y+\Delta y)$ of i-direction. Namely, Eq. 4 is to measure the distance between A and B. Therefore, the more similar A and B, the smaller the relative entropy(H). In here, threshold PMF of each direction set 0.2 by experiment. Finally, we select a reference point having the most minimum value by Eq. 5.

$$\arg\min \left(\sum_i H_i(x+\Delta x, y+\Delta y) \right) \tag{5}$$

where, H_i is entropy of PMF really computed by Parzen window, $(x+\Delta x, y+\Delta y)$ is coordinate. Fig. 7 shows the distance between A and B using relative entropy. In here, a solid arrow denotes a standard PMF for all pixels of each direction and a dotted arrow denotes PMF computed by Parzen window at each coordinate. Moreover, d_i denote distance between two distribution by relative entropy in i-direction. We

select point of the most equal probability by relative entropy in all direction. Therefore, the point has the most equal direction components.

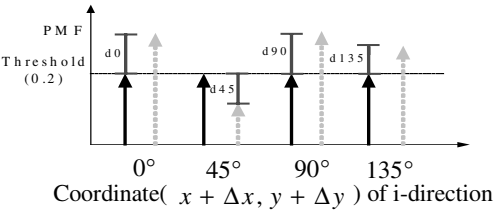


Fig. 7. Distance between two distribution by relative entropy

Fig. 8 shows reference point selected using entropy.

(a) Image by relative entropy (b) Reference point

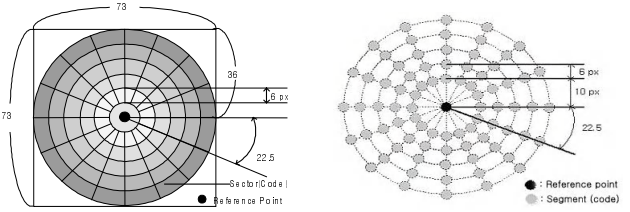
Fig. 8. Reference point detection image by relative entropy

3 Feature Extraction Using Filterbank Method

Once a reference point is established, next step is to extract unique and characterizing features of the fingerprint to be able to distinguish it from others. Existing filterbank method can lose the discerning attributes because the sectors of the outer band from the reference point are larger in size than those of the inner bands as shown in Fig. 9(a). Moreover, the nearest sectors from the reference point were found to be sensitive to the specific position of reference point extracted. Such shortcomings are avoided by maintaining the attribute regions to equal size. A novel filterbank method employed here extracts identifying features from each region as listed in the following procedure:

1) Extract 8 direction components (between 0and 157.5, 22.5 interval) using Gabor-filter[5][6][7] over the region.
2) Proceed with a normalization procedure in which the mean and variance of the entire image is adjusted because Gabor-filtered image is influenced by brightness.

In this way, eighty features (mean, variance) with respect to the reference point were determined by computing the parameters over each normalized image. In step 1, Gabor-filter is defined by



(a) Existing filterbank method (b) Novel filterbank method

Fig. 9. Segmentation of regions for feature extraction

$$g(x, y, \theta_k, \sigma_x, \sigma_y) = \exp\left[-\frac{1}{2}\left(\frac{x_{\theta_k}^2}{\sigma_x^2} + \frac{y_{\theta_k}^2}{\sigma_y^2}\right)\right] \times \exp(i2\pi f x_{\theta_k}), \tag{6}$$

$$x_{\theta_k} = x\sin\theta_k + y\cos\theta_k, \qquad y_{\theta_k} = x\cos\theta_k - y\sin\theta_k,$$

Note that f is the frequency of the sinusoidal plane wave, that is, the number of pixels between band and band. Also, θ_k is the orientation of the Gabor-filter, while σ_x and σ_y are the standard deviations of the Gaussian envelop along the x and y axes, respectively. Fig. 10(b) shows normalized image of eight-direction by Gabor-filter using Eq. 6. The model to compute mean and variance can be expressed by

$$D(\deg, x, y) = \sum_{w=-10}^{10}\sum_{v=-10}^{10} g_f(w,v)\cdot g_{\deg}(x+w, y+v)\cdot g_{\deg}(x+w, y+v) - \mu_g^2(x,y), \tag{7}$$

$$\mu_g(x,y) = \sum_{w=-10}^{10}\sum_{v=-10}^{10} g_f(w,v)\cdot g_{\deg}(x+w, y+v),$$

$g_{\deg}$ is a feature value of the eighty feature points*(x,y)* from the filtered image. Moreover, w and v reflect the filter size from feature point*(x,y)*. If filter size is too large, an overlapping can occur among nearby feature points and thereby may lose the essential attributes. If filter size is too small, there may not be enough features to be captured. We determined the optimum filter size to be 11×11 by experiment. Note that $g_f(w,v)$ denotes a convolution of mean-filter and Gaussian-filter. We can reduce computational load by designing a novel filter as $g_f(w,v)$. Eq. 8 denotes mean-filter having mean values in all coordinates.

$$M = \frac{1}{N}\sum_{i=-S_w}^{S_w}\sum_{j=-S_v}^{S_v} P(i,j) \tag{8}$$

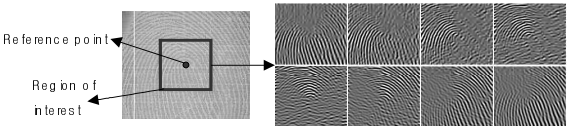
In here, M is the mean value of the neighboring region for each pixel, S_w and S_v are the filter size along the x-axis and y-axis, $P(i,j)$ is the value in pixel*(i,j)*, and N is the total number of pixels in the neighboring region. Moreover, the Gaussian filter(Eq. 9) is to weight for each feature point as ridges and to reduce weight for part as valley. Namely, we can obtain more accurate feature point by weighting for feature point.

$$G = e^{-\frac{r^2}{\sigma^2}}, \quad r = \sqrt{(x-size)^2 + (y-size)^2} \tag{9}$$

where, G is a Gaussian filter, (x,y) is the coordinate, size is the size of Gaussian filter. The Gaussian filter size was set to 18 by experiments. The σ value denotes the width of the Gaussian filter. If σ is too large, the filter would be more robust to noise, but would more likely smooth the image to the extent that the ridge and valley details in the fingerprint may get lost. If σ is too small, the filter would not be effective in removing the noise. The value was determined empirically and was set at 36. Note that $g_f(w,v)$ is defined as a convolution between the mean filter and the Gaussian filter.

$$g_f(w,v) = G * M \tag{10}$$

Note also that μ_g denotes convolution between Gabor-filtered image and $g_f(w,v)$ filter.



(a) Region of interest (b) Gabor-filtered normalization image

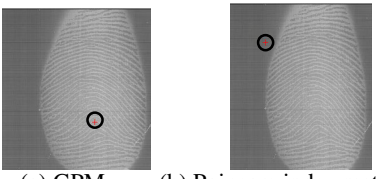
Fig. 10. Gabor-filtered normalized image of region of interest

4 Experimental Result

To evaluate the performance of our verification system, we made experiment as the following. The first series of experiments is to evaluate accuracy of reference point detection. We obtained each 6 images from the 20 same persons. So, a total of 120 fingerprint images are used. In here, we decided reference point position of one image having the best quality to the database. Then, we computed distance of the others and the reference point position. Table 1 shows average of distance for each subject. Particularly, Poincare has a defect for arch type. Fig. 11 shows this example.

Table 1. Average of distance for each subject between GPM and Poincare index method

	Loop	Double loop	Whorl	Arch
GPM	5.12(pixel)	6.39(pixel)	6.14(pixel)	6.08(pixel)
Poincare	7.08(pixel)	2.78(pixel)	5.39(pixel)	More than 20(pixel)



(a) GPM (b) Poincare index method

Fig. 11. Reference point detection example of GPM and Poincare index for arch type

The second series of experiments is to test the verification rate of our system. We took several fingerprint images from each subject and registered one image having the best quality to the database. This process was repeated for all 23 test subjects, and in the process a total of 211 fingerprint images were taken. In here, we made experiment under general condition and two separate noise conditions(noise due to variation in brightness and salt and pepper noise). Table. 2-4 shows these examples.

The third series of experiments is to evaluate the performance of GPM method as a function of the Parzen window size. Fig. 12 shows performance of the GPM method using 4 different sizes of the Parzen window. From this comparison, 50x50 window was adopted as the most optimum size for the verification process.

Table 2. Verification rate of a novel method(above) and Poincare index(below) under general environment

GPM	Loop	Double loop	Whorl	Arch	Total	Verification rate
FRR(%)	5.8	10.8	15.2	13.7	10.9	99.53
FAR(%)	0	0	0	0	0	
Poincare	Loop	Double loop	Whorl	Arch	Total	Verification rate
FRR(%)	14.7	7.8	8.7	62.7	23.5	98.95
FAR(%)	0	0	0	0	0	

Table 3. Verification rate of a novel method(above) and Poincare index (below) under brightness variation noise

GPM	Loop	Double loop	Whorl	Arch	Total	Verification rate
FRR(%)	10.3	23.9	19.6	27.5	19.4	99.16
FAR(%)	0	0	0	0	0	
Poincare	Loop	Double loop	Whorl	Arch	Total	Verification rate
FRR(%)	19.1	10.9	15.2	66.7	28.0	98.8
FAR(%)	0	0	0	0	0	

Table 4. Verification rate of a novel method(above) and Poincare index (below) under salt and pepper noise

GPM	Loop	Double loop	Whorl	Arch	Total	Verification rate
FRR(%)	14.7	32.6	19.6	41.2	26.1	98.87
FAR(%)	0	0	0	0	0	
Poincare	Loop	Double loop	Whorl	Arch	Total	Verification rate
FRR(%)	19.1	13.0	15.2	56.9	26.1	98.87
FAR(%)	0	0	0	0	0	

5 Conclusion and Future Works

We developed an on-line fingerprint verification system which operates in three steps: reference point detection, feature extraction, and matching. From the preliminary results, it has been shown that the proposed GPM method yielded superior verification performance compared to other conventional verification methods such as the Poincare algorithm. It has shown improved performance over the conventional method in a low noise environment, and it demonstrated more robust performance in a high noise environment such as the one caused by variations in image brightness.It was also shown that the new method requires less verification time compared to the conventional method. These are due to the fact that the method uses directional probability distribution information only, thus requiring no detailed preprocessing and

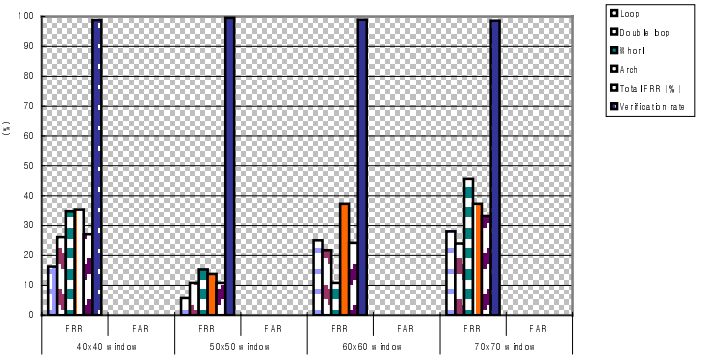


Fig. 12. Verification rate of the GPM method using 4 Parzen window sizes

making it less sensitive to the precise image events. Moreover, the most significant performance improvement attributed to the method is in the case of verifications against the arch type fingerprint images. Large reductions in FRR were noted consistently on all three cases of the comparisons against the Poincare algorithm. This improvement is due mainly to the GPM's ability to locate a more repeatable reference point from fingerprint images of the same subject. Our future work will be focused on the enhancement of the verification performance by improving the present algorithm under various noise environments. Because the preprocessing procedure is important in researching a more robust algorithm under noise environments, we will direct our future research towards local processing rather than global processing about filters used for preprocessing.

References

1. A. K. Jain, L. Hong, and R. Bolle, "On-Line Fingerprint Verification," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 4, pp. 302–313, April 1997.

2. Lin Hong, Yifei Wan, Anil Jain, “Fingerprint Image Enhancement: Algorithm and Performance Evaluation” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 777–789, August, 1998.
3. Kalle Karu and Anil K. Jain, ”Fingerprint Classification,” in Pattern Recognition, vol. 29, no. 3, pp. 389–404, 1996.
4. Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Classification, John Wiley & Sons, 2001.
5. Y. Hamamoto, S. Uchimura, M. Watanabe, T. Yasuda, Y. Mitani and S. Tomita, "A Gabor filter-based method for recognizing handwritten numerals," in Pattern Recognition, Vol. 31, No. 4, pp. 395–400, 1998
6. Anil. K. Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti, "Filterbank-based Fingerprint Matching," IEEE Transactions on Image Processing, vol. 9, no. 5, pp. 846–859, May 2000
7. Anil K. Jain, Salil Prabhakar, and Lin Hong, "A Multichannel Approach to Fingerprint Classification," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 4, pp. 348–359, April 1999
8. Anil K. Jain, Lin Hong, Sharath Pankanti, and Ruud Bolle, "An Identity-Authentication System Using Fingerprints," in Proceedings of the IEEE, vol. 85, no. 9, pp. 1365–1204, September 1997.

GPD-Based State Modification by Weighted Linear Loss Function

Taehee Kwon and Hanseok Ko

Department of Electronics Engineering, Korea University
5 ka-1, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
thkwon@ispl.korea.ac.kr, hsko@korea.ac.kr

Abstract. This paper proposes an effective loss function to assign the HMM state weight based on the MCE method. This is to remedy the performance limitation inherent in the traditional maximum likelihood method, which adjusts parameters to maximize the likelihood of training HMM. If minimum classification error method is used to minimize the error rate for training data, then the local optimum point can be achieved and the recognition performance can be achieved. However, if the amount of data used in the MCE training is too small, there can be a risk of overfitting to the training data. In this paper, we propose state modification by weighted linear loss function to overcome overfitting to training data. Representative experiments confirm this postulation and show the improvement in error rate when applied.

1 Introduction

Hidden Markov modeling (HMM) has become prevalent in speech recognition for capturing the acoustic models. It is statistically based and links a recognition task to the distribution estimation. The most commonly used method out of these distribution estimation methods is the maximum likelihood (ML) estimation. However, we lack the complete knowledge on the form of data distribution and training data is always inadequate in dealing with speech recognition. Usually the performance of a recognizer is normally defined by its expected recognition error rate and an optimal recognizer is the one that achieves the least expected recognition error rate. In this perspective, we can propose the state modification to exploit the discriminant information latent in the state likelihood without improved feature extraction, improved acoustic resolution. However, if the ML estimated HMMs have the high performance to some extent and the number of recognition errors used in the minimum classification error (MCE) training is small, the overfitting problem occurs as we train iteratively using the training data. Therefore, the MCE training not for the full training set but for misrecognized training set results in overfitting to small misrecognized training set. In this circumstance, there's a need to increase the number of errors by adjusting the misclassification measure. If we define a new loss function and train HMMs based on this loss function, then we can expect that overtraining can be prevented and the recognition error rate is reduced by producing increased number of recognition errors and adapting the state weights.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1100–1108, 2003.
© Springer-Verlag Berlin Heidelberg 2003

This paper begins in section 2 with a brief review of the MCE method. We then discuss in section 3 the state modification based on the MCE method. We discuss the newly proposed loss function in section 4 and discuss experimental results about state modification using four different types of loss functions in section 5. Finally, conclusion is described in section 6.

2 MCE Training

The aim of MCE training is to correctly discriminate the observations of an HMM for the best recognition results and not necessarily to try and find the best model for the distributions of the data. This section will briefly describe and discuss the MCE algorithm [1][2][3][4][5]. We shall also define an optimization criterion, which provides a reasonable estimate of the error probability.

In the traditional HMM-based speech recognizer, the discriminant function for class i is defined by the following equation for pattern classification

$$g_i(\mathbf{X};\Lambda)=\sum_{t=1}^{T}\left[\log a^{(i)}_{\bar{q}_{t-1}\bar{q}_t}+\log b^{(i)}_{\bar{q}_t}(\mathbf{x}_t)\right]+\log \pi^{(i)}_{\bar{q}_0} \tag{1}$$

where Λ is a set of classifier parameters, $\mathbf{X}$ is an observation sequence, $\bar{\mathbf{q}}=(\bar{q}_0,\bar{q}_1,\cdots,\bar{q}_T)$ is the optimal state sequence that maximizes the joint state-observation function for class i, a_{ij} denotes the probability of transition from state i to state j, and $b_j(\mathbf{x}_t)$ denotes the probability density function of observing $\mathbf{x}_t$ at state j. In a continuous multivariate mixture Gaussian HMM, the state output distribution is as follows.

$$b_j(\mathbf{x}_t)=\sum_{m=1}^{M} c_{jm}\, N(\mathbf{x}_t;\boldsymbol{\mu}_{jm},\boldsymbol{\Sigma}_{jm}) \tag{2}$$

where $N(\cdot)$ denotes a multivariate Gaussian density, $\boldsymbol{\mu}_{jm}$ is the mean vector in state j, mixture m and $\boldsymbol{\Sigma}_{jm}$ is the covariance matrix in state j, mixture m.

For input utterance, the decision rule is used. For an input utterance $\mathbf{X}$, the class C_i is decided as

$$C(\mathbf{X})=C_i \quad \text{if } i=\arg\max_j g_j(\mathbf{X};\Lambda) \tag{3}$$

where $g_j(\mathbf{X};\Lambda)$ is the discriminant function of the input utterance or observation sequence $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n)$ for the j th model.

In first, it is necessary to express the operational decision rule (3) in a functional form. A class misclassification measure which is a continuous function of the classifier parameters Λ and attemps to emulate the decision rule is therefore defined.

$$d_i(\mathbf{X};\Lambda) = -g_i(\mathbf{X};\Lambda) + \log\left[\frac{1}{N}\sum_{j=1, j\neq i}^{N} \exp\left[g_j(\mathbf{X};\Lambda)\eta\right]\right]^{\frac{1}{\eta}} \tag{4}$$

where η is a positive constant and N is the number of N-best competing classes. For an i th class utterance $\mathbf{X}$, $d_i(\mathbf{X}) > 0$ implies misclassification and $d_i(\mathbf{X}) \leq 0$ means correct classification.

The complete loss function is defined in terms of the misclassification measure using a smooth zero-one function.

$$l_i(\mathbf{X};\Lambda) = l(d(\mathbf{X};\Lambda)) \tag{5}$$

The smooth zero-one function can be any continuous zero-one function, but is typically the following sigmoid function

$$l(d) = \frac{1}{1+\exp[-rd+\theta]} \tag{6}$$

where θ is usually set zero or slightly smaller than zero and r is a constant. Finally, for any unknown $\mathbf{X}$, the classifier performance is measured by

$$l(\mathbf{X};\Lambda) = \sum_{i=1}^{M} l_i(\mathbf{X};\Lambda)\,1(\mathbf{X}\in C_i) \tag{7}$$

where $1(\cdot)$ is the indicator function.

The optimal classifier parameters are those that minimize the expected loss function. The generalized probabilistic descent (GPD) algorithm is used to minimize the expected loss function. The GPD algorithm is given by

$$\Lambda_{n+1} = \Lambda_n - \varepsilon_n \mathbf{U}_n \nabla l(\mathbf{X};\Lambda)\,|_{\Lambda=\Lambda_n} \tag{8}$$

where $\mathbf{U}$ is a positive definite matrix, ε_n is the learning rate or step size of adaptation, and Λ_n is the classifier parameter set at time n.

The GPD algorithm is an unconstrained optimization technique. But some constraints must be maintained for HMMs so some modifications are required. Instead of using a complicated constrained GPD algorithm, Chou et al [1], applied GPD to transformed HMM parameters. The parameter transformations ensure that there are no constraints in the transformed space where the updates occur. The following HMM constraints should be maintained in the original space.

$$\sum_j a_{ij} = 1 \;\; and \;\; a_{ij} \geq 0, \;\; \sum_k c_{jk} = 1 \;\; and \;\; c_{jk} \geq 0, \;\; \sigma_{jkl} \geq 0 \tag{9}$$

The following parameter transformations should be used before and after parameter adaptation.

$$
\begin{aligned}
&a_{ij} \rightarrow \tilde{a}_{ij} \quad where \;\; a_{ij} = e^{\tilde{a}_{ij}} / \left(\sum_k e^{\tilde{a}_{ik}}\right) \\
&c_{jk} \rightarrow \tilde{c}_{jk} \quad where \;\; c_{jk} = e^{\tilde{c}_{jk}} / \left(\sum_k e^{\tilde{c}_{jk}}\right) \\
&\mu_{jkl} \rightarrow \tilde{\mu}_{jkl} = \mu_{jkl} / \sigma_{jkl} \\
&\sigma_{jkl} \rightarrow \tilde{\sigma}_{jkl} = \log \sigma_{jkl}
\end{aligned}
\tag{10}
$$

3 State Modification

In order to exploit the discriminative information latent in state likelihood, each state have a weight such that each state output score is a product of state output likelihood and state weight. We assume that there are M HMMs as the basic recognition units, and that the total number of states is J in a recognition unit. The basic recognition units consist of phonemes or words.

In the traditional HMMs, the discriminant function for class i is defined by equation (1). In contrast, in the state modified HMMs, the discriminant function for class i is denoted by the following equation (11)

$$
g_i(\mathbf{X};\Lambda) = \sum_{t=1}^{T} \left[\log a^{(i)}_{\bar{q}_{t-1}\bar{q}_t} + w^{(i)}_{\bar{q}_t} \log b^{(i)}_{\bar{q}_t}(\mathbf{x}_t)\right] + \log \pi^{(i)}_{\bar{q}_0} \tag{11}
$$

where $w^{(i)}_{\bar{q}_t}$ is the weight of a optimal state at time t in class i.

Adaptation of state weights set initially to one is given by previously discussed equations (4) ~ (7) and equation (12).

$$
w^{(i)}_j(n+1) = w^{(i)}_j(n) - \varepsilon_n \frac{\partial}{\partial w^{(i)}_j} l(\mathbf{X};\Lambda)\Big|_{w^{(i)}_j = w^{(i)}_j(n)} \tag{12}
$$

Similar to equation (9), the state weight constraints should be maintained in the original space. In other words, the state weights for a speech recognition unit are constrained by equation (13)

$$
\sum_{j=1}^{J} w_j = J, \;\; 0 < w_j < J \tag{13}
$$

where w_j is a state weight for state j in a HMM and J is the number of states in a HMM.

Finally, parameter transformations are given by the following equations.

$$
w_j \rightarrow \tilde{w}_j \quad where \;\; w_j = e^{\tilde{w}_j} / \left(\sum_k e^{\tilde{w}_k}\right) \tag{14}
$$

where $\tilde{w}_j$ is the transformed state weight of w_j.

In the following sections, we propose four different kinds of loss functions and discuss experimental results of applying those loss functions for state modification.

4 Linear Loss Function

In general, we only consider misclassified speech in the MCE training. However, if the amount of training data used in MCE training is too small and we just focus on this misrecognized training data, then the overfitting to training data occurs and there exists the difference between training and testing set error rates. So there is a need to modify the smooth zero-one loss function that we should minimize into an alternative form. For this purpose, we propose alternative loss functions. For the first time, we can modify the misclassification measure by adding a weighted likelihood of correct class to the misclassification measure [6]. This modified misclassification measure is inserted into a sigmoid function to produce the sigmoid zero-one loss function. In the next time, we can consider a misclassification measure as a loss function to produce the linear loss function. By using this linear loss function, we can increase the gradient associated with a loss function for correct string by a uniform factor k while not affecting the gradient associated with a loss function for incorrect string as shown in Equation (16).

$$d_i(\mathbf{X};\Lambda) = -g_i(\mathbf{X};\Lambda) + \log\left[\frac{1}{N}\sum_{j=1, j\neq i}^{N} \exp\left[g_j(\mathbf{X};\Lambda)\eta\right]\right]^{\frac{1}{\eta}} \tag{15}$$

$$\begin{aligned}\tilde{d}_i(\mathbf{X};\Lambda) &= d_i(\mathbf{X};\Lambda) - k \cdot g_i(\mathbf{X};\Lambda) \\ &= -(1+k)\cdot g_i(\mathbf{X};\Lambda) + \log\left[\frac{1}{N}\sum_{j=1, j\neq i}^{N} \exp\left[g_j(\mathbf{X};\Lambda)\eta\right]\right]^{\frac{1}{\eta}}\end{aligned} \tag{16}$$

As a result of modified misclassification measure, another loss functions are sigmoid zero-one loss function where a modified misclassification measure is inserted into a sigmoid function, weighted linear loss function that is exactly the same as a misclassification measure (16).
The weighted linear loss function will tend to reinforce correct strings efficiently while still penalizing errors such that this weighted linear loss function can reduce overfitting to training data by adjusting the value of k .

5 Experiments

To investigate the performance of the proposed method in speech recognition, we used the Korean isolated digit database recorded in a silent room. The isolated digit database consists of utterances uttered by 500 speakers (250 male and 250 females). Utterances from 400 speakers (200 males and 200 females) were used as training data

and those from 100 speakers (50 males and 50 females) were used as testing data. Each speech sampled by 16 KHz sampling rate was down sampled into 11.025 KHz rate for implementing a speech recognizer on a DSP board. We extract 13-order mel-frequency cepstral coefficients (MFCCs), 13-order delta coefficients, 13-order acceleration coefficients including log energy. As a result, an observation consists of a 3 stream, 39-order feature vector. We chose the phoneme unit as a recognition unit and produced a 3-state, 8 mixture HMM for each phoneme unit. All state output distributions consist of 8 mixture, multivariate mixture Gaussian distributions. We constructed a HMM set as baseline system by ML method and then constructed four HMM sets by weighting state output likelihood based on GPD method using different loss functions. First of all, we perform the MCE training of basic parameters to compare performance by state modification to performance by MCE training of basic parameters.

5.1 MCE Training of Basic Parameters

In this section, we performed the MCE training of basic HMM parameters using four loss functions. The basic HMM parameters mean mixture weights, means, standard deviations of each state output distribution. Table 1 shows that training and testing data accuracies vary when the accuracy of training data reached the maximum point. As the number of iterations increases, the accuracy of training data is increased gradually but the accuracy of testing data is not increased efficiently. The aim of MCE training is to correctly discriminate the observations of an HMM for the best recognition results of training and testing data. So, the MCE training of basic parameters is not efficient in this case.

Table 1. Word accuracies (%) with four kinds of loss functions

Data	MLE	SIG	WL+SIG	LIN	WL+LIN
Training	98.40	98.58	98.58	98.58	98.71
Testing	98.00	98.00	98.00	98.00	98.00

SIG : sigmoid loss function, LIN : linear loss function
WL : weighted likelihood of correct class

5.2 MCE Training of State Weights

In this section, we performed the MCE training of HMM state weights using four loss functions. Each state weights are set to one as initial values. To estimate the state weights, we used 3 most confusable strings for computing the misclassification measure, $N = 3$ and set $r = 0.01$, $k = 0.005$.

We performed four experiments based on four different loss functions i.e. sigmoid zero-one loss function, sigmoid zero-one loss function resulting from a modified misclassification measure, a linear loss function, a weighted linear loss function resulting from a modified misclassification measure. Table 2 shows maximum word accuracies of both training and testing data when the word accuracy of testing data reaches the maximum point in the four different experiments.

Table 2. Word accuracies (%) with four kinds of loss functions

Data	MLE	SIG	WL+SIG	LIN	WL+LIN
Training	98.40	98.54	98.54	98.44	98.56
Testing	98.00	98.17	98.58	98.00	98.50

Fig. 1 shows that training and testing data accuracies vary as the number of iterations increases using a sigmoid zero-one loss function resulting from a misclassification measure. As the number of iterations increases, both training and testing data accuracies decrease owing to overfitting to training data. The aim of MCE training is to correctly discriminate the observations of an HMM for the best recognition results. Therefore, this result is not very desirable. However, Fig. 2 shows that the training data accuracy decreases gradually while the testing data accuracy fluctuates significantly as the number of iterations increase.

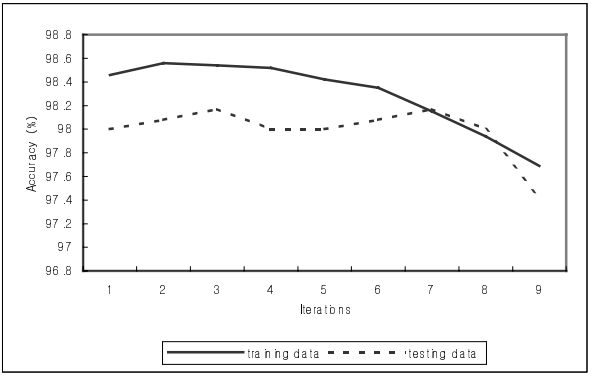


Fig. 1. MCE Training of State Weights (SIG)

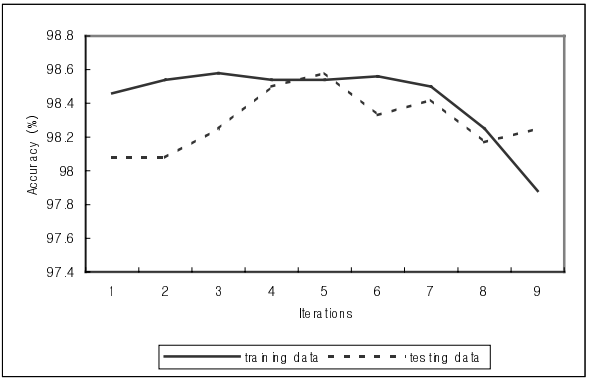


Fig. 2. MCE Training of State Weights (WL+SIG)

Fig. 3 shows that training and testing data accuracies are not invariant nearly as the number of iterations increases using a linear loss. However, Fig. 4 shows that training and testing data accuracies increase consistently at the same time using a weighted linear loss function resulting from the modified misclassification measure as the number of iterations increases. So, this weighted linear loss function with weighted likelihood term is very efficient to overcome overtraining in state modification and the word accuracy is comparable to the ML method without improved feature extraction, improved acoustic resolution.

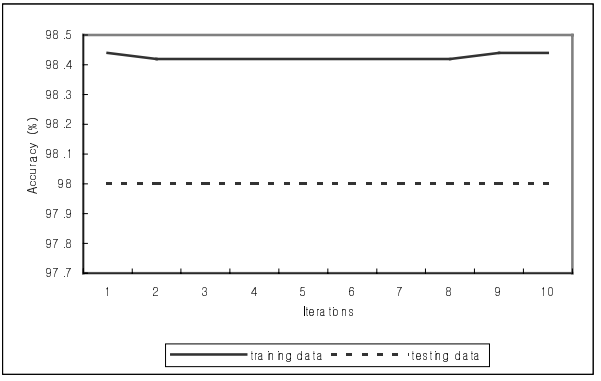


Fig. 3. MCE Training of State Weights (LIN)

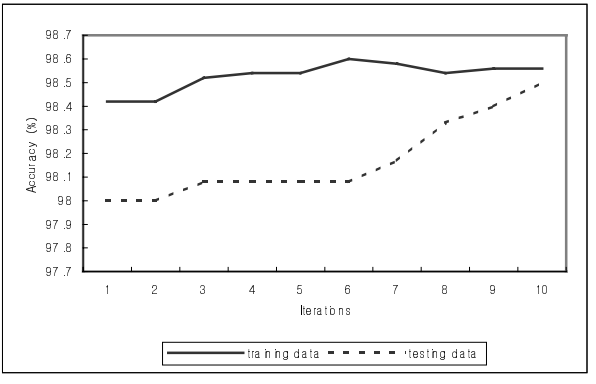


Fig. 4. MCE Training of State Weights (WL+LIN)

It is important to adjust k -values suitably in state modification by weighted linear loss function. Therefore, the performance improvements as the result of adjusting the values of k are shown in Table 3. Table 3 shows word accuracies of training and testing data in the case where the accuracy of testing data reaches the local maximum point.

Table 3. Word accuracies (%) as the result of adjusting the values of k

k	0.001	0.002	0.003	0.004	0.005
Training	98.54	98.63	98.48	98.67	98.56
Testing	98.17	98.25	98.25	98.25	98.50

6 Conclusions

Experimental results showed that the use of weighted linear loss function by weighted likelihood of correct class could improve word accuracies for both training and testing data consistently. It is very important to reduce word accuracies for both training and testing data consistently. Fig. 4 shows the consistency of performance improvement in training and testing data and Table 2 shows that the state modification by weighted linear loss function with weighted likelihood of correct class reduces word error rate by 25 % comparing to ML method. In this result, the state modification by weighted linear loss function is very efficient to discriminate observations.

References

1. X.Huang, et al: Spoken Language Processing, Prentice-Hall PTR, 2001.
2. Biing-Hwang Juang, Wu Chou, Chin-Hui Lee: Minimum Classification Error Rate Methods for Speech Recognition, IEEE Transactions on Speech and Audio Processing, Vol. 5, No. 3, PP. 257–265, May 1997.
3. Wu Chou, Chin-Hui Lee, Biing-Hwang Juang: Minimum Error Rate Training Based on N-best String Models, IEEE ICASSP 93, PP. 652–655, April 1993.
4. Wu Chou: Discriminant-Function-Based Minimum Recognition Error Rate Pattern-Recognition Approach to Speech Recognition, Proceedings of the IEEE, Vol. 88, No. 8, PP. 1201–1223, August 2000.
5. Min-Tau Lin, Andreas Spanias, Philipos Loizou: An Improved Approach to Robust Speech Recognition Using Minimum Error Classification, Speech Communication, Vol.30, PP. 27–36, 2000.
6. Darryl William Purnell and Elizabeth C. Botha: Improved Generalization of MCE Parameter Estimation with Application to Speech Recognition, IEEE Transactions on Speech and Audio Processing, Vol. 10, No. 4, PP. 232–239, May 2002.
7. O.W. Kwon, C.K. Un: Performance of HMM-based Speech Recognizers with Discriminative State-Weights, Speech Communication, Vol. 19, PP. 197–205, June 1996.
8. S. J. Young: The HTK Book 3.0, Cambridge University, 2000.

Spectral Subtraction Using Spectral Harmonics for Robust Speech Recognition in Car Environments

Jounghoon Beh and Hanseok Ko

Departments of Electronics and Computer Engineering, Korea University, 5 ka-1, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
jhbeh@ispl.kore.ac.kr

Abstract. This paper addresses a novel noise-compensation scheme to solve the mismatch problem between training and testing condition for the automatic speech recognition (ASR) system, specifically in car environment. The conventional spectral subtraction schemes rely on the signal-to-noise ratio (SNR) such that attenuation is imposed on that part of the spectrum that appears to have low SNR, and accentuation is made on that part of high SNR. However, since these schemes are based on the postulation that the power spectrum of noise is in general at the lower level in magnitude than that of speech. Therefore, while such postulation is adequate for high SNR environment, it is grossly inadequate for low SNR scenarios such as that of car environment. This paper proposes an efficient spectral subtraction scheme focused specifically to low SNR noisy environment by representing harmonics distinctively in speech spectrum. Representative experiments confirm the superior performance of the proposed method over conventional methods. The experiments are conducted using car noise-corrupted utterances of Aurora2 corpus.

1 Introduction

The mismatch between training and testing condition is a major problem in ASR systems. The techniques to solve this problem can be categorized into two principal approaches. First is the spectral subtractive-type of algorithm performing noise suppression using short-time spectral amplitude, such as spectral subtraction, nonlinear spectral subtraction and Weiner filter. The other is the feature compensation algorithm such as cepstral mean normalization or vector Tayler series. In general, it is well known that spectral subtractive-type algorithm is very simple and efficient especially in stationary noisy environments.

This paper is about a new spectral subtractive-type scheme based on the idea that even though speech is heavily corrupted by noise, the shape of spectral harmonics of speech is well preserved as when speech is not corrupted [1] [2].

The weakness of conventional spectral subtractive-type algorithm is identified and presented in Section 2. The proposed remedial approach is described in Section 3. In Section 4, we show the proposed method's effectiveness over conventional methods with representative experiments using Aurora 2. Concluding remarks are provided in Section 5.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1109–1116, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Spectral Subtractive-Type Algorithm

When speech signal *x(n)* is corrupted by background additive noise *b(n)*, the corrupted speech can be expressed as follows:

$$y(n) = x(n) + b(n) \quad . \tag{1}$$

If speech and noise are assumed to be uncorrelated, in frequency domain, it can be represented as follows:

$$|Y(k)|^2 = |X(k)|^2 + |B(k)|^2 \quad . \tag{2}$$

where *k* is the index of frequency bin.

2.1 Spectral Subtraction

In the case of power spectral subtraction, the short-time power spectrum $|\hat{X}(k)^2|$ of enhanced speech signal can be obtained by subtracting its noise estimate $|\hat{B}(k)^2|$ off from the corrupted speech. Note that every procedure is carried out in frame-by-frame basis [3]. Instead of power spectrum, its magnitude spectrum can be made available. General power spectral subtraction algorithm is as follows.

$$|\hat{X}(k)|^2 = \begin{cases} |Y(k)|^2 \left(1 - \alpha \dfrac{|\hat{B}(k)|^2}{|Y(k)|^2}\right), \\ \quad if \ |Y(k)|^2 - \alpha |\hat{B}(k)|^2 > \beta |Y(k)|^2 \\ \beta |Y(k)|^2, \\ \quad otherwise \end{cases} \quad . \tag{3}$$

With over-subtraction factor α and floor factor β, this algorithm concerns the trade-off between noise reduction and residual noise. Note that the enhanced short-time power spectral amplitude $|\hat{X}(k)^2|$ depends on a posteriori SNR [4]:

$$\mathrm{SNR}_{\mathrm{p}} = |Y(k)|^2 / |\hat{B}(k)|^2 \quad . \tag{4}$$

2.2 Nonlinear Spectral Subtraction

Nonlinear spectral subtraction algorithm is as follows [5]:

$$|\hat{X}(k)|^2 = \begin{cases} |Y(k)|^2 \left(1 - \dfrac{\Phi(k)}{|Y(k)|^2}\right), \\ \quad if \ |Y(k)|^2 - \Phi(k) > \beta |Y(k)|^2 \\ \beta |Y(k)|^2, \\ \quad otherwise \end{cases} \quad . \tag{5}$$

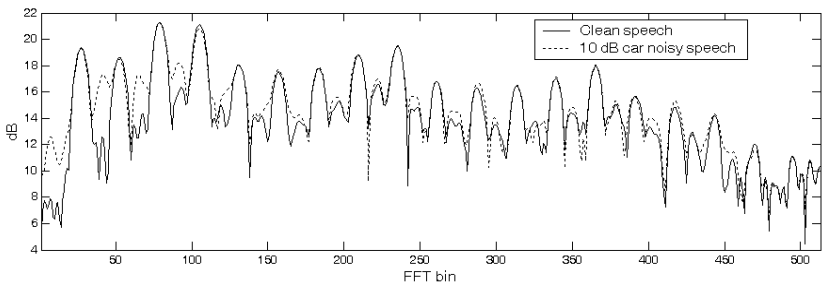


Fig. 1. Example spectrum of a speech frame (pronunciation /oh/ by female)

Function $\Phi(k)$ can be chosen arbitrarily to implement the notion that relatively greater subtraction is applied to the low SNR region of spectrum and less subtraction to the high SNR region.

In general, the spectral amplitude of speech components (e.g. spectral harmonics) is higher than that of noise or the side lobes among harmonics so that those algorithms are rather suitable for reasonably high SNR (about 15~20dB) noisy speech cases. However, it appears that in the case of low SNR environments (especially 0~10dB), such that when the noise level is as close in amplitude as that of speech, there occurs unnecessarily subtracted speech region or less subtracted noisy region in noisy speech spectrum. Consequently, instead of the mundane use of SNR as a measure for subtraction, we need a new and better measure for activating the subtraction procedure. In particular, the subtraction over spectrum requires a more accurate measure than mere SNR in order to apply the subtraction rule, which is selective to speech-dominant region vs. noise-dominant region of spectrum.

3 Proposed Algorithm

3.1 Speech Dominant Region vs. Noise Dominant Region

In speech, it is observed that the voiced speech segment has peaks positioned periodically in spectrum due to the vibration of vocal cords. These points are very critical in speech sound perception. Figure 1 illustrates this phenomenon with a sample spectrum of speech frame capturing the pronunciation /oh/ in one utterance contained in Aurora2 corpus. Note that at the peaks, their amplitudes are far greater than the amplitudes at the points between adjacent small peaks, or side lobes. Also, it is observed that the degree of corruption by the noise in the peaks is not as much as the degree of corruption at the points over the side lobes. From this observation, it can be deduced that in speech spectrum, the speech-dominant regions exist over or near the peaks and the noise-dominant regions exist over or near the side lobes.

The fundamental frequency can be obtained roughly by auto-correlation and then we can find the peak points from the roughly obtained fundamental frequency. For the frequency regions covering the peak points and their vicinity, we apply a small over-subtraction factor and large floor factor. In other regions, we apply a large over-

subtraction factor and small floor factors. The detailed procedure is described in rest of Section 3.

Note that all procedures in the proposed algorithm are done on a frame-by-frame basis.

3.2 Peak Points Detection and Segmentation

It is known that the spectral harmonics are located at the points of multiples of fundamental frequency [6]. First, using autocorrelation function, the indices of local maxima in autocorrelation values are found. Among those indices, we can find the fundamental frequency f_0 by taking the reciprocal of the index that represents the maximum value. Secondly, by applying the nonlinear smoothing method [6], f_0 is modified to better reflect the true fundamental frequency. This algorithm works well even in high degree noisy environment. Autocorrelation function is expressed as follows [7]:

$$\phi(\tau) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n+\tau) \quad . \tag{6}$$

We then establish the index as k_0 for the frequency bin that corresponds to f_0. Using k_0, the peak points (harmonics) h_l are determined as $h_1 = k_0$, $h_2=2k_0$, $h_3=3k_0$, ..., $h_L=Lk_0$ where L is the index of the last component of harmonics. Using this procedure, we accomplish determining all harmonics in the spectrum of input frame, which are assumed to be speech-dominant region.

For the purpose of implementing the proposed scheme (section 3.4), frequency axis is divided into several non-overlapping bands in the following form.

$$\begin{aligned} &[1 \quad h_1], \quad \left[h_1 \quad h_1+\frac{k_0}{2}\right], \quad \left[h_1+\frac{k_0}{2} \quad h_2\right], \quad \left[h_2 \quad h_2+\frac{k_0}{2}\right], \quad \cdots\cdots \\ &, \quad \left[h_{L-1}-\frac{k_0}{2} \quad h_{L-1}+\frac{k_0}{2}\right], \quad \left[h_{L-1}+\frac{k_0}{2} \quad h_L\right], \quad \left[h_L \quad \frac{FFT\ order}{2}+1\right] \quad . \end{aligned} \tag{7}$$

3.3 Voice Activity Detection (VAD) and Noise Estimation

Note that in each input frame, whether it is the speech frame or not, the fundamental frequency is calculated. As a result, the value of autocorrelation is also calculated at each frame inevitably. Since this value is appeared to be a more effective measure to distinguish speech region from non-speech region than mundane energy measure in experiments, so we used this value instead of energy for VAD. Then, two separate procedures are employed for robust detection of speech vs. non-speech region. First, we take a logarithm to this value, and then a smoothing procedure is carried out. If this value is greater than the mean value of autocorrelation obtained during the first 5 frames by 0.3, the relevant frame is decided as 'speech frame'. Secondly, for frames determined as 'speech frame', if the detected fundamental frequency is not in 50~400Hz, then it is concluded as 'non-speech frame' once again. It is well known

that 50~400Hz is considered an appropriate fundamental frequency range for human voice.

Input speech is coded into 32ms frames, with a frame-shift of 16ms. Then short-time FFT is applied. We performed 256 points FFT analysis on the relevant input frame.

For the noise estimation, we assume that any starting input speech is followed by a silence or background noise segment corresponding to 5 frames (96 ms). In that duration, mean of short-time spectral amplitude at each frequency bin are calculated. Then, if input frame is determined as 'non-speech frame', the estimated noise spectral magnitude is updated as follows;

$$|B_i(k)|^2 = \lambda |B_{i-1}(k)|^2 + (1-\lambda)|B_i(k)|^2, \quad 0.65 \le \lambda \le 0.998 \quad . \tag{8}$$

where i is the frame index and k is the index of FFT bins and we use λ=0.75.

This procedure is for the purpose of applying different subtraction rule to the speech frame and the non-speech frame. It is because of the 'non-speech frame', whose harmonics components cannot be determined. Consequently, the proposed scheme is inappropriate to 'non-speech frames'.

3.4 Spectral Subtraction

Based on the result of VAD procedure, we apply different subtraction rule.

Speech Frame. In order to implement the proposed scheme, we designed following simple linear function.

- If $k \in [1 \quad h_1]$ *or* $k \in \left[h_{l-1} + \frac{k_0}{2} \quad h_l\right]$, $l = 2,3,4,...,L$, then

$$\gamma(k) = \frac{\alpha_{\min} - \alpha_{MAX}}{h_l - \left(h_{l-1} + \frac{k_0}{2}\right)}(h_l - k) + \alpha_{MAX} \quad . \tag{9}$$

$$\delta(k) = \frac{\beta_{MAX} - \beta_{\min}}{h_l - \left(h_{l-1} + \frac{k_0}{2}\right)}(h_l - k) + \beta_{\min} \quad . \tag{10}$$

- If $k \in \left[h_l \quad h_l + \frac{k_0}{2}\right]$ *or* $k \in \left[h_L \quad \frac{FFT\ order}{2} + 1\right]$, $l = 1,2,3,...,L-1$, then

$$\gamma(k) = \frac{\alpha_{MAX} - \alpha_{\min}}{\left(h_l + \frac{k_0}{2}\right) - h_l}(h_l - k) + \alpha_{\min} \quad . \tag{11}$$

$$\delta(k) = \frac{\beta_{\min} - \beta_{MAX}}{\left(h_l + \frac{k_0}{2}\right) - h_l}(h_l - k) + \beta_{MAX} \quad . \tag{12}$$

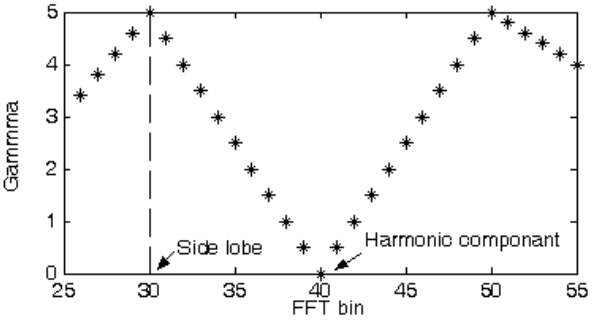


Fig. 2. An illustration about implementing over-subtraction factor to FFT bins

$\gamma(k)$ applies the minimum over-subtraction factor to each harmonic component and the maximum over-subtraction factor to the middle point at each components. Then, over-subtraction factors of points exist in the interval between those points are interpolated linearly. Figure.2. illustrates a shape of $\gamma(k)$ which of fundamental frequency is about 156Hz when sampling rate is 8kHz and 1,024 points FFT analysis is applied.

On the contrary, $\delta(k)$ applies the maximum floor factor to each harmonic component and the minimum floor factor to the middle point at each component. Then, other floor factors between those points are generated by linear interpolation.

The proposed subtraction rule can be represented as follows:

$$|\hat{X}(k)|^2 = \begin{cases} |Y(k)|^2 \left(1 - \gamma(k) \dfrac{|\hat{B}(k)|^2}{|Y(k)|^2}\right), \\ \quad if \ |Y(k)|^2 - \gamma(k)|\hat{B}(k)|^2 > \delta(k)|Y(k)|^2 \\ \delta(k)|Y(k)|^2, \\ \quad otherwise \end{cases} . \tag{13}$$

The values of parameters used are as follows:

$$\alpha_{MAX} = 5, \alpha_{\min} = 2, \beta_{MAX} = 0.2, \beta_{\min} = 0.05 \quad . \tag{14}$$

Non-speech Frame. We applied the subtraction rule as same as conventional method using Eq. (1) with α_{MAX}, β_{min} in Eq. (14).

4 Experimental Results

4.1 Experimental Conditions

In these experiments, utterances corrupted by car noise among Aurora2 corpus are used. For comparison, spectral subtraction algorithm [3] and nonlinear spectral subtraction algorithm [5] are evaluated. Throughout all experiments, the VAD procedure and the noise estimation procedure are the same as the proposed method. Finally, for each algorithm, the enhanced speech signals are recovered by overlap-and-add man-

ner followed by inverse FFT on frame-by-frame basis. Then, all performances are evaluated.

In Table 1, the meanings of the each abbreviation are as follows:

- 'SS': general power spectral subtraction.
- 'SS_D': general power spectral subtraction + different subtraction rule.
- 'NSS': nonlinear spectral subtraction.
- 'NSS_D': nonlinear spectral subtraction + different subtraction rule.

Experiments 'SS_D', 'NSS_D' are evaluated with different subtractions rule as it is applied in the proposed method. That is, if the input frame is decided as 'non-speech frame', we applied the rule in section 3.4.2 which is the conventional subtraction rule Eq. (1) with the over-subtraction factor α_{MAX} and floor factor β_{min}.

4.2 Experimental Results

'Avg' denotes to the mean value of word accuracy over 0dB~20dB. From the Table, the proposed method is seen effective at low SNR cases. The average word accuracy of the proposed method also shows that it is superior over other spectral subtraction approaches.

Table 1. Word Accuracy (%)

SNR	Baseline	SS	SS_D	NSS	NSS_D	Proposed
clean	99.02	98.78	98.78	98.75	98.75	98.81
20dB	97.97	98.27	98.51	98.18	98.21	98.36
15dB	94.24	97.08	97.55	97.26	97.20	97.52
10dB	78.17	91.59	94.15	93.95	93.83	94.66
5dB	42.59	79.39	81.78	84.61	84.49	85.00
0dB	14.67	31.26	50.58	54.64	54.49	56.13
-5dB	9.25	10.47	17.78	17.95	17.86	20.52
Avg.	69.79	79.52	84.51	85.73	85.64	86.33

5 Conclusions

This paper proposes an efficient spectral subtraction scheme focused to specifically low SNR noisy environments by distinguishing the speech-dominant segment from the noise-dominant segment in speech spectrum. Consequently we let the shape of the spectral harmonics be preserved more clearly in noisy environments, which are very critical in speech sound perception. Representative experiments confirm the superior performance of the proposed method to conventional methods. In particular, the proposed method is seen effective at low SNR cases (SNR < 10 dB). The average word accuracy of the proposed method also shows that it is superior to other recently intro-

duced approaches. The experiments are conducted using car noise-corrupted utterances of Aurora2 corpus.

References

1. Jensen, J., Hansen, J.: Speech Enhancement Using a Con-strained Iterative Sinusoidal Model, IEEE Transactions on Speech and Audio Processing, Vol. 9, No. 7, Oct (2001) 731–740
2. Ealey, D., Kellher, H., Pearce, D.: Harmonic tunneling: track-ing non-stationary noises during speech, Eurospeech 2001(2001) 437–440
3. Berouti, M., Schwartz, R., Makhoul, J.: Enhancement of speech corrupted by additive noise, Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing, April (1979) 208–211
4. Virag, N.: Single Channel Speech Enhancement Based on Masking Properties of the Human Auditory System, IEEE Transactions on Speech and Audio Processing, Vol. 7, No. 2, Mar (1999) 126–137
5. Lockwood, P., Boudy, J.: Experiments with a Nonlinear Spectral Subtractor(NSS), Hidden Markov Models and the pro-jection, for robust speech recognition in cars, Speech Communication, Vol. 11 (1992) 215–228,
6. Hess, W.: Pitch Determination of Speech Signals, Springer-Verlag Berlin Heidelberg New York Tokyo (1983)
7. Rabiner, L., Schafer, R.: Digital Processing of Speech Signals, Prentice-Hall (1978)
8. Boll, S.F.: Suppression of Acoustic Noise in Speech Using Spectral Subtraction, IEEE Transaction on Acoustics, Speech and Signal Processing, Vol.27, No.2, April (1979) 113–120

A Workflow Management and Grid Computing Approach to Molecular Simulation-Based Bio/Nano Experiments

Karpjoo Jeong[1], Dongwook Kim[1], Moon Hae Kim[1],
Suntae Hwang[2], Seunho Jung[3], Youngho Lim[4], and Sangsan Lee[5]

[1] College of Information and Communication, Konkuk University, Seoul, Korea
{jeongk, dwkim, mhkim}@konkuk.ac.kr
[2] Department of Computer Science, Kookmin University, Seoul, Korea
sthwang@kookmin.ac.kr
[3] Department of Microbial Engineering, Konkuk University, Seoul, Korea
shjung@konkuk.ac.kr
[4] Department of Applied Biology & Chemistry, Konkuk University, Seoul, Korea
yoongho@konkuk.ac.kr
[5] Supercomputing Center, Korea Institute of Science and Technology Information, Daejon, Korea
sslee@hpcnet.ne.kr

Abstract. In this paper, we propose an approach to molecular simulation-based experiments which combines workflow management and grid computing techniques to address both the computing issue due to the challenging computation requirement from molecular simulation and the management issue due to distributed, heterogeneous computing platforms. We present a workflow management system customized for such experiments and explain how this workflow system can be integrated with computational grids.

Keywords: Molecular Simulation, Grid Computing, Workflow Management

1 Motivations

1.1 Molecular Simulation-Based Analysis of Bio/Nano Materials

Molecular simulation is an effective approach to computing and analyzing behaviours of biologically or chemically relevant molecular structures (e.g., proteins, DNA, RNA) easily by software tools such as `CHARMM` and `GAUSSIAN`[2, 9] without going through long, complicated, labor-intensive real experiments. In spite of such apparent advantage, molecular simulation has not been widely used because it has a challenging requirement for computation which can be satisfied only by expensive parallel computers.

Compuational grids which aim at combining computing resources from a number of organizations into large scale parallel/distributed computing platforms give us an economical solution to address the challenging computing issue

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1117–1126, 2003.
© Springer-Verlag Berlin Heidelberg 2003

of molecular simulation[5, 6, 8]. However, in order to make grid computing techniques effective for molecular simulation-based experiments, the following issues must be addressed:

- *Single System Image*. Compuational grids are, in nature, wide-area heterogeneous distributed systems. Complicated issues involved in heterogeneous distributed computing must be hidden from scientists whose knowledge about distributed computing is limited. That is, the single system image is needed.
- *Workflow Management*. Molecular simulation-based experiments are composed of various complicated tasks such as molecular structure modelling, model verfication, simulation progress monitoring, molecular simulation, simulation result verfication. They form a complex workflow in which there are inter-task dependencies and whose entire execution may take a couple of months. Managing those tasks properly is already a big burden on scientists and computational grids enable much more concurrent experiments and create much more management burdens on scientists. Therefore, the workflow mangement support is crucial for effective molecular simulation-based experiments on computational grids.

1.2 Brief Review of Conventional Workflow Systems

Workflow is used to express a complex process as a set of interconnected, smaller, less complicated component tasks[11]. In various areas such as industrial and administrative process management and design processes, workflow has successfully been applied. Technically, workflow management implies the automated coordination, control, and inter-task communication.

A workflow instance is an activity involving the coordinated execution of multiple tasks performed by different processing entities[15]. These tasks could be manual, automated, either created specifically for the purpose of the workflow application being developed, or possibly already existing as legacy programs[14]. A workflow process is an automated organizational process involving both human (manual) and automated tasks.

There are many commercial workflow management systems and research prototypes developed for various applications[18]. These systems are either customized for specific application domains or designed as general purpose systems. Both classes of workflow systems fail to satisfy requirements for molecular simulation-based experiments. Apparently, the former class is not suitable and the latter class requires furthermore customization for such experiments.

In addition, these workflow systems are not designed to work with computational grids. In the grid computing area, there are currently efforts on application-specific groupware systems, but major focuses are still on general purpose middleware systems[5, 6]. In [1], Buyya, Branson, Giddy, and Abramson proposed a World Wide Grid-based virtual laboratory system for drug designs. Although it is aimed at molecular simulation, this system is aimed at processing a large number of independent docking tasks by computational grids and assumes

no human interventions. However, our work assumes both human interventions and the dependency among molecular simulation tasks.

There are research efforts on scientific workflow systems[4, 14, 17, 20]. Although they address various issues involved in scientific workflows and biological research processes, they do not address those issues specifically involved in molecular simulation-based experiments. or furthermore issues on grid computing at all.

For these reasons, workflow systems customized for molecular simulation-based experiments and designed to support computational grids must be needed in order to make molecular simulation a widely-applicable effective technique for various bio/nano research work.

2 Workflow Model for Molecular Simulation-Based Experiments

We propose a workflow model for the molecular simulation-based experiment. The model is designed to be simple because the fundamental structure of the experimental process is not complicated and this simple design makes the system easy to use. Instead of one of existing workflow systems, we decided to develop a custom workflow system because most scientists need to use conventional simulation software tools (i.e., *legacy programs* widely applied and these tools must be integrated in a seamless fashion. Furthermore, we aim at design our workflow system to support grid computing.

The workflow model is defined to consist of:

- a set of *Programs*. We assume simulation tools to be registered in the system. The management and control information about each software are stored in the workflow system:
  - `Program ID`. We assume a system-wide unique name for each program.
  - `Interaction type`. A program can be run as a background process or requires GUI interactions with the user. The latter class of programs must be run locally or requires local GUI emulation facilities for remote execution (e.g., thin client features).
  - `Invocation Parameters`. They can be default values or must be required at invocation time.
  - `I/O file specification` . File names, file types (e.g., file extensions) and file content viewers are specified.
- a set of *Tasks*. We assume each task to be associated with a program which is pre-registered in the workflow system. That is, the task is defined as the function of the program. The information about each task is:
  - `Task ID`. We assume a task to have a system-wide unique name.
  - `Program ID`. The name of the program to execute is specified.
  - `Invocation parameter values`. Values for invocation parameters are given.

  - `Activation Type`. According to the activation condition, tasks are grouped into two classes: *automatically-started* and *manually-started*. The former class of tasks are is immediately initiated as soon as they are ready (e.g., all tasks prerequisite to the task are finished). When a task in the manually-started class is ready, the workflow system sends the user a notification (e.g., an email message or a popup message in the client GUI, *etc.*) and waits for him or her to ask for invocation.
  - `Monitoring Flag`, `Monitoring Program ID` and `Output File Names`. If this flag is on, then the workflow system shows the user the progress of the current task. Currently, we simply assume the progress of the program to be available as appended data in output files. In addition, we assume a monitoring program which is registered. This monitoring program periodically read newly appended data in specified output files and visualize them.
- a set of *Task Dependencies*.
  In our model, tasks are inter-connected and outputs from one task are used as inputs for another task. We call this a *task dependency*. A task can have multiple parent nodes. A task can be started only after all the parent tasks are finished.
  In addition to the execution order, a task dependency defines the *mapping information* between the output files of the parent task and the input files of the child task.

We define a *workflow instance* to be a workflow definition for a certain experiment. A workflow instance consists of a set of tasks and a set of task dependencies.

3 Workflow Management System for Molecular Simulation-Based Experiments

The workflow management system consists of:

- `Workflow Engine`. The workflow engine manages all the running workflow instances created by a number of users. The engine schedules tasks for each workflow instance.
- `Workflow Client` (GUI). The workflow client provides the user with a GUI that allows the user to create, monitor, and guide workflow instances for experiments. This client is written in JAVA and can be invoked on any platform.
- `Workflow Instance Databases`. All the workflow instances are managed by the central repository. Updates to workflow instances are transactional and therefore resilient to failure.
- `Task Template Repository`. Tasks frequently used are saved as templates in the repository. The user can retrieve and update templates for new experiments.

- `Program Repository`. Simulation tools are registered in the repository. Tasks are basically an execution of a certain simulation software and when the user defines tasks, he or she retrieves info about programs from the program respository and uses it to update task templates.
- `Program Adaptor`. Simulation programs require input files and produce output files. In order to make the workflow system independent of specific simulation programs, a program adaptor for each simulation program is built to allow the workflow system to manipulate input and output files in an uniform way.
- `Interface to Computational Grids`. The workflow system can delegate the execution job of molecular simulation software to a computation grid in a user-transparent way. The workflow system is designed to interoperate with computational grids through the Globus middleware.

4 Implementation Status and Experiments

4.1 Workflow System Implementation

Implementing groupware systems like a workflow system is challenging because those systems are required to support heterogeneous platforms and fault tolerance. Without the support for those features, groupware systems are not able to integrate a variety of legacy systems or may lose the entire intermediate work on a single failure. In order to address these issues, we use a fault-tolerant distributed computing system called Persistent Linda [12] as an implementation computing system.

The Linda model which Persistent Linda is based on allows arbitrary processes on distributed heterogeneous platforms to communicate and sychronize via virtual shared memory called tuple space[3]. Tuple space is a bag of tuples which are almost equivalent to database tuples in terms of how to access them. Due to ease of coordination and the support for heterogeneous platforms, Linda has been used for various groupware systems[7, 19, 21]. Persistent Linda extends the Linda model to support transactional persistence.

The workflow engine (in fact, a Persistent Linda application system) maintains workflow instances (i.e., tasks and task dependencies for an experiment) in the persistent tuple space and keep track of which tasks are ready to run according to task dependencies. For ready tasks that require an user notification, it first informs the user which tasks are ready to run. For those which can be automatically started, it immediately invokes them.

In Linda, program invocation is executed as tuple space access operations. Figure 1 illustrates how the workflow engine works. The workflow engine creates an invocation tuple for a ready task. Then, the program adaptor for the program required by the task retrieves the tuple and invokes the program. The program adaptor is designed to handle invocation details about the program.

In the workflow system, a simple shared file system is implemented on Persistent Linda. This shared file system is used to manage I/O files for simulation programs in a simple way. In this file system, metadata about I/O files is stored

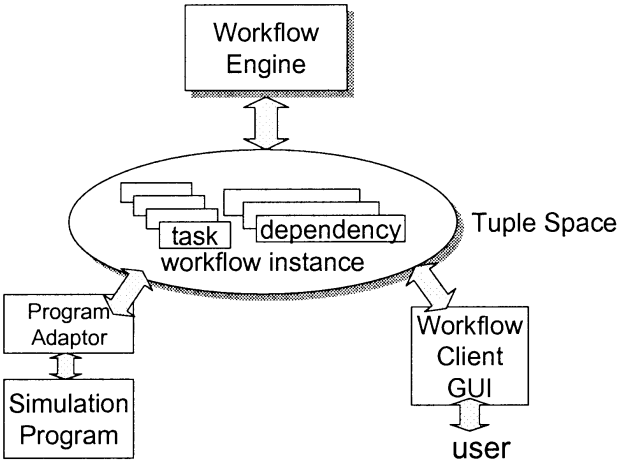


Fig. 1. How the Workflow Engine Works

in tuple space, but real data is managed as files. Program adaptors access I/O files through tuple space.

The workflow client is implemented in JAVA so that it can be run on any platform. The client displays the state of an workflow instance, gives notifications, and allows the user to examine intermediate files and to terminate tasks.

There are several molecular simulation software packages[2, 9]. Currently, our implementation of the workflow system is focused on the `CHARMM` software packages and its relevant tools. Therefore, current program adaptors are developed for those tools. `CHARMM` is one of the most popular molecular simulation package that uses the `CHARMM` force field to model the energetics, forces and dynamics of biologogical molecules by the classical method of integrating Newton's equations of motion.

For `CHARMM`, crucial input files are `inp`, `psf`, `crd` files and important output files are `out`, `trj` files. The program adaptor for `CHARMM` is designed to handle those files automatically.

4.2 Extensions to Computational Grids

In the design described in Section 4.1, the workflow engine directly interacts with program adaptors. In this section, we present how the workflow system supports computational grids.

In the grid computing-based design, the workflow engine and program adaptors can be located in different organizations which require authentication for remote communication. In such wide area environments, the workflow engine interacts with program adaptors via the *grid gateway system*. The grid gateway system consists of:

- `Gateway Agent`. The gateway agent plays the role of the delegate of a program adaptor. It receives an invocation request from the workflow engine, passes the request to the gateway server and waits for results from the server. When it receives results, then it sends them to the workflow system.
- `Gateway Server`. The gateway server manages program adaptors and waits for invocation requests from gateway agents on a number of organizations. When the server receives a request for the execution of a program, it invokes the corresponding program adaptor and waits for the adaptor to return results. When it receives results, it sends them back to the requesting agent.
- `Authentication Server`. For security, gateway agents and servers must use authentication keys from the same authentication server (`Globus CA server`).

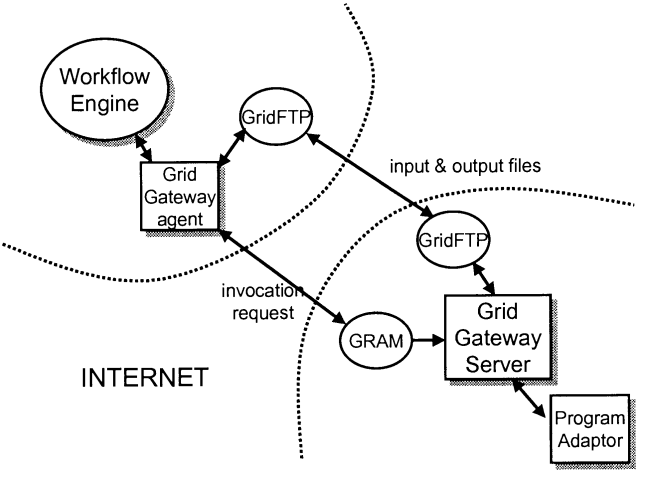


Fig. 2. Design of Computational Grid Support

Figure 2 illustrates how the grid gateway system works with the workflow system. Gateway agents and servers communicate by Globus toolkits such as `GRAM`, `GridFTP`, `Globus CA`, and `MDS`. They are implemented in JAVA in order to support heterogeneous platforms.

4.3 Experiment: Analyzing Cyclo-Oligosacchrarides for Chiral Discrimination

Our workflow system was applied for a real experiment for the chiral discrimination of R- and S-propranolol by β-cyclodextrin[13]. In this experiment, we performed computer aided molecular modeling studies such as Monte Carlo and molecular dynamics simulations.

We used CHARMM as a molecular simulation package, a Linux cluster with 32 nodes as compute server for simulation, and an SGI Octane workstation as platform for modeling, visualization, and verification. Our workflow system provided the user with the single system view. The system automatically invoked CHARMM jobs on remote nodes in the Linux cluster and tranfered input and output files among computers in a user-transparent way.

Figure 3 and 4 shows molecular models and their docking results, respectively.

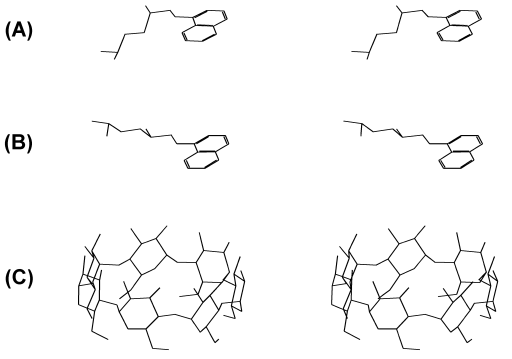


Fig. 3. Stereoview of molecular models used in the MC simulations. (A) (*R*)-propranolol, (B) (*S*)-propranolol, (C) β-CD

.

5 Conclusions and Future Work

In this paper, we assert that the approach combining workflow management and grid computing is an effective way to support large scale molecular simulation-based bio/nano experiments. We present a workflow system customized for those experiments and show how the system is integrated with computational grids in a user-transparent way.

Our experience with a real experiment for the chiral discrimination of *R*- and *S*-propranolol by β-cyclodextrin demonstrated that the workflow system enabled scientists to run many jobs on a Linux cluster concurrently without worrying about complicated distributed computing details.

The current implementation is targeted at the CHARMM molecular simulation package and its related tools. The implementation of the support for other simulation packages such as GAUSSIAN is underway. We are currently planning on

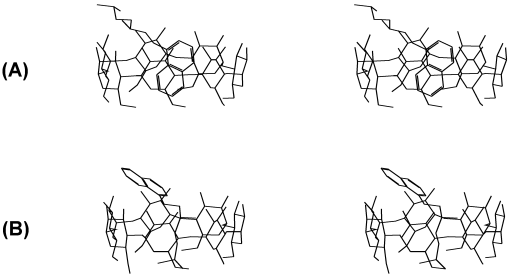


Fig. 4. Stereoview of lowest-energy configurations of the inclusion complexes of both enantiomers of propranolol and b-CD in MC docking simulations. (A) (R)-propranolol-β-CD complex, (B) (S)-propranolol-β-CD complex

large scale experiments (building a database for chiral materials) on computational grids (i.e., Globus environments) in the near future.

Acknowledgements.

This wok was partially supported by MIC (Ministry of Information and Communication) through National Grid Infrastructure Implementation Project of KISTI (Korea Institute of Science and Technology Information). It was also in part supported by University IT Research Center Project.

References

1. Buyya, R., Branson, K., Giddy, J., Abramson,D.: The Virtual Laboratory: A Toolset for Utilising the World-Wide Grid to Design Drugs, Proceedings of 2nd IEEE International Symposium on Cluster Computing and the Grid, May 2002, Berlin, Germany.
2. Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S., Karplus, M.: CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, Journal of Computational Chemistry, Vol. 4, 1983.
3. Carriero, N. and Gelernter, D.: How to Write Parallel Programs, The MIT Press, 1990.
4. Chin, G., Schuchardt, K., Myers, J., and Gracio, D.: Participatory Workflow Analysis: Unveiling Scientific Research Processes with Scientists, Proc. of the 6th Biennial Participatory Design Conference, 2000.

5. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, Vol 15., No. 3, 2001.
6. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services, Architecture for Distributed Systems Integration, Jan. 2002.
7. Freeman, E., Hupfer, S., and Arnold, K.: JavaSpaces Principles, Patterns and Practice, Addison Wesley, 1999.
8. Frenkel, D. and Smit, B.: Understanding Molecular Simulation, Academic Press, 2002.
9. The Official Gaussian Home Page: http://www.gaussian.com/.
10. Goble, C.: The low down on e-science and grids for biology, Comparative and Functional Genomics, Vol. 2, pp. 365-370, 2001.
11. James, H., Hawick, K., and Coddington, P.: An Environment for Workflow Applications on Wide-Area Distributed Systems, Hawaii International Conference on System Sciences, 2001.
12. Jeong, K., Talla, S., Wyckoff, P., and Shasha, D.: An Approach to Fault Tolerant Parallel Processing on Intermittently Idle, Heterogeneous Workstations, Proc. the 27th International Symposium on Fault-Tolerant Computing, Jun. 1997.
13. Kim, H., Jeong, K., Lee, S. and Jung, S.: Molecular Modeling of the Chiral Recognition of Propranolol Enantiomers by a β-Cyclodextrin, Bulletin of Korean Chemical Society, Vol. 24, No. 1, 2003.
14. Kochut, K., Arnold, J., Sheth, A., Miller, J., Kraemer, E., Arpinar, B., Cardoso, J.: IntelliGEN: A Distributed Workflow System for Discovering Protein-Protein Interactions, International Journal on Distributed and Parallel Databases, Special Issue on Bioinformatics, 2002.
15. Krishnakumar, N. and Sheth, A.: Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations, Distributed and Parallel Database Journal, Vol. 3, No. 2, 1995.
16. Natrajan, A., Crowley, M., Wilkins-Diehr, N., Humphrey, M., Fox, A. and Grimshaw, A.: Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion, Proceeding of the HPDC Conference, San Francisco, CA, USA, Aug 7-9, 2001.
17. Peleg, M., Yeh, I., and Altman, R.: Modeling biological processes using Workflow and Petri Net models, Bioinformatics, vol. 18, pp. 825-837, 2002.
18. Riempp, G.: Wide-Area Workflow Management, Springer-Verlag, 1998.
19. Sun Microsystems, Jini Specifications, Available from Sun Microsystems WWW Site (http://java.sun.com/products/javaspaces), 1998.
20. Weske, M., Vossen, G., and Medeiros, C.: Scientific Workflow Management: WASA Architecture and Applications, Architecture and Applications, Fachbericht Angewandte Mathematik und Informatik 03/96-I, Universitat Munster, 1996.
21. Wyckoff, P., McLaughry, S., Lehman, T., Ford, D.: TSpaces, IBM System Journal, 1998.

Simulation Infrastructure for Modeling Large Scale Neural Systems

Charles C. Peck, James Kozloski, A. Ravishankar Rao, and Guillermo A. Cecchi

IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
{cpeck, kozloski, ravirao, gcecchi}@us.ibm.com

Abstract. This paper describes the Large-scale Edge Node Simulator, a problem solving environment for the implementation of large scale models of neural systems. This work was motivated by the absence of adequate modeling tools for this domain. The object-oriented Large-scale Edge Node Simulator was developed after a rigorous requirements analysis for this class of simulations. An example use of this environment for a complex neural simulation of cortical plasticity is presented. It is shown that the Large-scale Edge Node Simulator is capable of meeting the challenges of simulating large scale models of neural systems.

1 Introduction

This paper describes the Large-scale Edge Node Simulator, a problem solving environment for the implementation of large scale models of neural systems.

Neural systems are highly complex, structured, and can be viewed from many levels of abstraction. For example, the human brain is composed of roughly 100 billion neurons interconnected by 100 trillion synapses. These fundamental elements are assembled into approximately 850 anatomically identifiable structures [1], such as the cerebellum and the visual cortex, each consisting of specific classes of neurons interconnected in stereotyped patterns. These structures have patterned connections to and from other structures. Finally, these networks of structures interact with the sensors and muscles of the body to generate behavior that is appropriate for a particular environmental context.

Since the earliest experimental studies of neural dynamics [2], simulations have provided value to the understanding of neural systems. Simulations can guide experiments by providing a conceptual framework within which to interpret data, form hypotheses,and allow exploration of phenomena that are not directly accessible. For instance, simulations have been used to model epileptic seizures for clinical purposes [3]. At a higher level of abstraction, large scale simulations can provide a testbed for exploring the fundamental computational principles of global brain function. The value and impact of simulations is expected to increase dramatically with the continuing improvement of computational performance and the full exploitation of recently developed experimental

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1127–1136, 2003.
© Springer-Verlag Berlin Heidelberg 2003

techniques, such as PET and fMRI, multi-electrode recordings from brain areas, and optical recordings.

Adequate tools for large scale neural simulations do not presently exist. Currently available simulation environments are either general purpose or tailored to a specific class of neural models. General purpose tools lack support for neural system-specific modeling, such as simple methods for generating complex network topologies and global architectures. Conversely, environments like Neuron™ and Genesis™ provide tools for detailed compartmental modeling of individual neurons with few capabilities for models at other levels of abstraction. Furthermore, the low abstraction level of compartmental modeling makes large-scale simulations of neural systems computationally intractable.

The remainder of the paper presents the specific challenges of large scale neural system modeling and their technological implications, a description of the Large-scale Edge Node Simulator and how this environment solves these challenges, an example application of the environment, and a brief conclusion.

2 Challenges of Large Scale Modeling of Neural Systems

Large scale modeling of neural systems imposes several key technical challenges, including extensibility, scalability, and interoperability. In addition, challenges specific to each phase of the experimental cycle must also be satisfied.

Extensibility is required to enable the introduction and execution of arbitrary models. Furthermore, the simulation must not constrain these models to operate at a specific level of abstraction. As examples, one may desire to model: the visual cortex as a collection of thousands of minicolumn model instances, a minicolumn as a collection of about 100 neuron model instances, or a neuron as a collection of hundreds of compartmental model instances.

Scalability is required to support simulations with complexity spanning many orders of magnitude. For example, a simulation at the level of neurons and synapses might involve a few hundred neurons and thousands of synapses, or it might require $> 10^6$ neurons and $> 10^8$ synapses to model a large structure, such as the hippocampus.

Interoperability of models is required to support the needs of the neuroscientific community. These needs include exchanging and refining models, communicating experimental phenomena, developing higher level models based on models developed by others, and replicating results.

Conducting a complex experiment takes a considerable amount of time from conception through data analysis. While the design and implementation of most simulation systems emphasizes run-time execution performance, experience shows that the time required for setup and initialization of complex simulations exceeds that for run-time execution by many orders of magnitude. The same is true for data collection and analysis. To reduce the time required for a simulation experiment, the simulation environment must therefore expedite each of the three phases of the experimental lifecycle (i.e., setup and initialization, run-

time execution, and data collection and analysis). The remainder of this section addresses the specific challenges in each of the three phases.

2.1 Setup and Initialization

To be most efficient, the manner in which the simulation is setup and initialized should be well matched to the manner in which the experiment designer conceptualizes large scale neural systems. In particular, there should be support for instantiating multiple types of models and for establishing complex patterns of connectivity. This latter requirement is especially important for maintaining scalability, as point-to-point specification of connections would not be possible for experiments consisting of large numbers (possibly millions) of interconnected model instances.

The simulation environment must also support the parameterization of the model instances. This challenge is exacerbated by the large numbers of instances, that they may be instantiated from arbitrary models, and that the parameters for each instance may vary or even depend upon the configuration context.

In addition to parameterizing the instances of models, the simulation environment must also support parameterization of connections or relationships among models. For example, the effect one neuron has on another is dependent on where the synapse occurs on the postsynaptic neuron. This position information, which may be parameterized, is not a property of either neuron, rather it is a property of the relationship between them.

2.2 Run-Time Execution

Computational efficiency is critical for large scale simulations. The run-time environment must effectively utilize all available computational resources, such as the CPUs and system memory. In addition, the run-time environment should support both time-step and discrete time event update modes to allow the selection of the most appropriate mode for a particular model. For example, since neuron models are continuously and frequently updated they should use the time-step mode. In contrast, models of environmental stimuli may use fewer resources with the higher overhead, but far less frequent, discrete time event updates.

The run-time environment must also support control of computation sequencing at each time step. This control enables integration of heterogeneous models and greater parallelism. Such control would enable, for example, the ability to execute all synapse model instances in parallel prior to executing any neuron model instances. Once all synapse instances have been executed, the neuron model instances could then be executed, also in parallel.

2.3 Data Collection and Analysis

Data collection and analysis is specific to the particular experiment being performed, and must be easy, efficient, involve minimal code development, and maximize the reuse of third party data analysis tools, such as data visualization and

statistical analysis packages. For ease and efficiency, data collection and analysis setup should be done in the same way as simulation setup and initialization. Furthermore, it should be possible to extract specific data from individual model instances or desired collections of instances.

The simulation environment must also support a rich set of data collection and analysis operations. For example, it must be possible to perform data collection every time step, when a predicate is satisfied, or when a user interactively indicates. It must also be possible to capture time-series of variables of interest. Finally, the simulation environment should provide interactive access to arbitrary data while a simulation is running.

3 A Large-Scale Neural Simulation Infrastructure

3.1 Overview

This section presents a detailed description of the Large-scale Edge Node Simulator approach for satisfying each of the requirements for large-scale simulations of neural systems.

3.2 Modeling Architecture

The primary design philosophy for the Large-scale Edge Node Simulator was to allow experiment designers to express their designs in a manner consistent with their conceptualizations, to allow model developers to write their models for maximum computational efficiency, and to bridge the gap between simulation specification and model development with technology.

This philosophy is illustrated in Figure 1. The "Simulation Design" refers to the experimenter's conceptualization of the experiment. The "Domain Level" refers to a specification using the terminology and knowledge framework provided by neuroscience. The "Network Level" refers to the technology bridge and the "Computational Level" refers to the level at which model development is performed.

A description of how each of these levels is implemented is given below.

Domain level. To allow the simulation designer to create domain level models without requiring simulation system expertise, the Large-scale Edge Node Simulator includes a specification language. This language allows domain level modeling to proceed using familiar terms, and for domain level simulation designers to access, parameterize, and compose existing models automatically based on a high level specification.

Network level. To bridge the gap between domain level modeling and its mapping onto the computational infrastructure, the network level generates an isomorphic description of the simulation in the form of an edge-node graph.

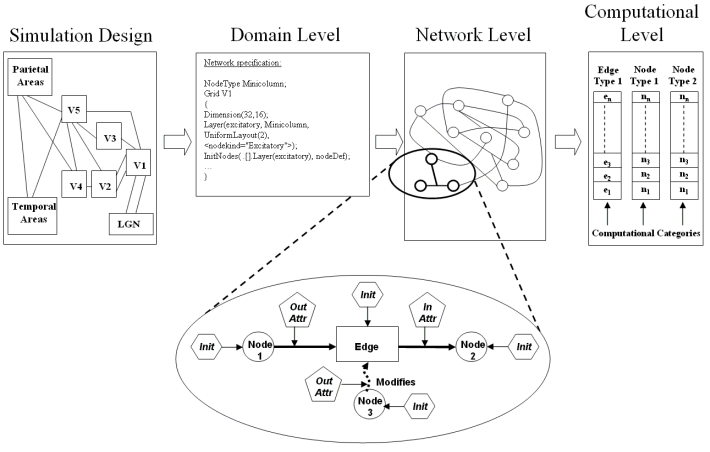


Fig. 1. Summary of design philosophy for the problem solving environment.

Both domain level simulation designers and model code developers can address network elements directly through this graph description. The graph is fundamentally constrained in that nodes connect directionally to an arbitrary number of edges while edges connect directionally to precisely two nodes.

To a simulation designer, edges and nodes represent network elements that are invoked within a specification using domain level names (eg., neuron, synapse, minicolumn). To a code developer, these same network elements are computational models that support standard interfaces for initialization, execution, browsing, and data collection and analysis.

Connections within the edge-node graph are traversed through hierarchical collections of node repertoires and edge connection sets. Some aspects of this hierarchy have been proposed previously in other simulation system designs [4]. There are two types of repertoires. Composite repertoires are composed of other repertoires, and grid repertoires consist of a multidimensional array nodes. Grids provide access to nodes according to their grid coordinate system and/or their model type. Grids have arbitrary dimensionality and may support an arbitrary number of nodes at each grid coordinate. Finally, each repertoire provides access to all edges that are entirely contained within it.

Computational level. To provide a basis for model development, the Large-scale Edge Node Simulator provides standard API's for model initialization, parallelization, memory management, execution, browsing, and data collection. These APIs assume a uniprocessor or shared memory multiprocessor system. Model developers can use these APIs to efficiently exploit the available computational resources.

3.3 System Architecture

As shown in Figure 2, the Large-scale Edge Node Simulator possesses data stores containing specification files and model libraries, and a simulation infrastructure that includes an initializer and simulation kernel. The initializer consists of a parser for reading specification files and generating the simulation implementation and a loader for dynamically loading the specified models. The simulation kernel owns and executes computational categories, which themselves contain executable models, and the edge-node graph description of the simulation. In addition, a Java-based data browser and simulation control system, and distributed third-party tools for data collection and analysis are depicted running on systems that form dynamic links to the simulation to perform these functions.

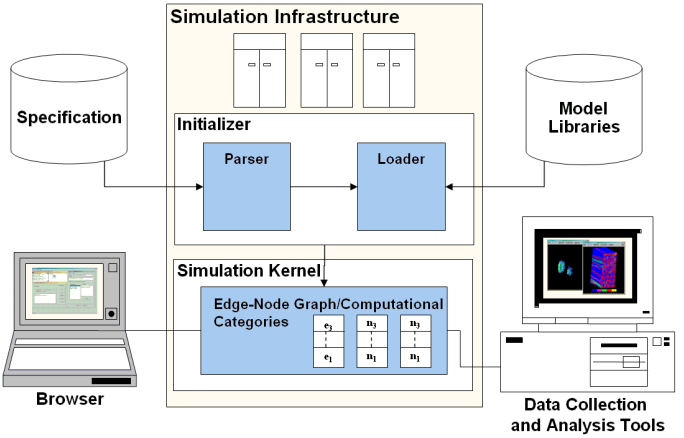


Fig. 2. Architectural design of problem solving environment.

Within this system, data about the simulation originates in the specification files and model libraries. The specification is read by the parser, which, after dynamically loading the specified models and initialization algorithms, generates instances of edges and nodes. These instances are then composed into an edge-node graph using the specified connection algorithms. Nodes and edges, their connectivity, and the relationships between them, are then parameterized algorithmically. Both browsing systems and data collection and analysis tools access the graph through the network level hierarchy.

3.4 Initialization Solution

Specification language. The specification language allows the setup and initialization of large-scale simulations of neural systems to proceed using the terms with which the simulation designer conceptualizes these systems. The language

is based on a generic programming paradigm, is similar to C++ in its grammar, and uses Standard Template Library syntax. Functor (function objects), variables, and scripts are each specifiable using the language. Multiple types of models can be readily specified for instantiation, then parameterized and connected together in complex patterns using functors. Relationships between models are parameterized separately from the models themselves. The specification language also supports setup and initialization of data collection and analysis.

Loader. For extensibility, all components (eg., nodes, edges, and functors) of the simulation are dynamically loaded from model libraries. The simulation system provides a means to parameterize loaded models easily from within a specification. To support efficient data exchange between arbitrary models, an interface negotiation protocol has been provided that allows the initializer to verify that specified connections are supported, and to establish a direct link between data objects once, prior to simulation run-time.

3.5 Run-Time Solution and Simulation Kernel

Thread-pools of CPU-bound kernel threads. Thread pools are available to model code developers as they design and implement computational categories for high performance. Threads are managed by the simulation engine, and bound to all available CPUs in a shared memory system.

Synchronous and asynchronous event queues. To achieve high performance, the simulation engine supports two simulation modes for model computation: time-step and discrete-event. These modes are integrated by the simulation engine through the use of both synchronous and asynchronous event queues. Code developers can therefore choose to implement models using the more efficient mode of operation, and simulation designers can specify the use of these models arbitrarily within a simulation.

Arbitrary phase ordering. The simulation infrastructure allows model code developers to arbitrarily control the sequencing of their model computations through assignment of computational phases during initialization. During implementation, developers assign computational categories their relative computational phases (e.g., *compute edges before nodes*), which are then made absolute when models are loaded during initialization.

3.6 Data Collection and Analysis Solution

The simulation infrastructure creates a easy and efficient means to access, collect, and analyze arbitrary data from within a simulation.

Composable Collection and Analysis Tools. The operations of data collection and analysis are implemented as composable, reusable elements within the simulation infrastructure. Data collection objects are specified, parameterized, and composed using domain modeling terms from within a specification. These objects collect arbitrary data from arbitrary simulation components or collections of components. In addition, the data collection and analysis solution supports an arbitrary, composable triggering scheme, such that data can be collected continuously, over any specified time interval relative to a simulation event, or upon a user command.

Network, Data, and Analysis Browser. Users can access all simulation elements with a Java-based network, data, and analysis browser. The browser allows a user to navigate the edge-node graph, select data for collection and analysis, and invoke arbitrary analysis tools. Tools can be created from prespecified types, or can be dynamically loaded from tool libraries.

Integrated Open Source Data Visualization Tool. Third party, open source data visualization is fully integrated within the simulation infrastructure. The visualization tool, DataExplorer™, employs a socket interface to access active data collection and analysis tools within the simulation. Sockets can be created during initialization or dynamically, then directed to DataExplorer™ for real-time data display. Because data collection and analysis tools can access arbitrary data, the entire simulation is accessible for visualization.

4 Simulation Example

4.1 Rationale and Simulation Overview

In order to validate our simulation environment, we implemented a previously published neural simulation [5], which modeled results from experiments on the plasticity of connections in somatosensory cortex [6].

The somatosensory cortex is organized into regions, such that each hand digit maps to a specific region. Neurons within each region are organized into tight clusters called neuronal groups, such that synaptic weights between neurons within a neuronal group is significantly higher than weights between neurons in different groups. This group structure is hypothesized to arise as a consequence of self-organization through synaptic changes driven by the sensory input.

In our simulation, the somatosensory cortex receives inputs from four digits. The digits were modeled using two receptor sheets, derived from a grid of size 32x16. Each receptor sheet projects to a patch of cortex, overlaid on a grid of size 32x16. The simulation includes three classes of neurons and two classes of synaptic connections, for a total of 1500 neurons and 170,000 synaptic connections. The simulation also includes rules for synaptic modification, and specific patterns of anatomical connectivity between the network elements. The simulation was run on a 4-processor IBM RS/6000 machine running AIX Version 4.3.2. The Large-scale Edge Node Simulator was integrated with a third party data analysis tool, the IBM DataExplorer™ data visualization system.

4.2 Specification Example

The simulation was implemented in our environment using the specification language in about 50 lines. The following excerpt shows the ability of the language to capture complex configurations in a compact way.

Cortical simulation specification excerpt:

```
 1  NodeType ExcCell;
 2  NodeType InhCell;
 3  Grid Cortex
 4  {
 5     Dimension(32,16);
 6     Layer(excitatory, ExcCell, UniformLayout(2), <kind="Exc">);
 7     InitNodes(.[].Layer(excitatory), nodeDefault);
 8     Layer(inhibitory, InhCell, UniformLayout(1), <kind="Inh">);
 9     InitNodes(.[].Layer(inhibitory), nodeDefault);
10     list<int> e2e = {1,8,8};
11     list<int> e2i = {1,4,8,24,32};
12     list<int> i2e = {1,8,8};
13     list<int> i2i = {0};
14     centerSurround(.[],.[], e2e, e2i, i2e, i2i);
15  };
```

In this specification, two node types are declared "`ExcCell`" and "`InhCell`" (lines 1,2). Next a grid repertoire "`Cortex`" is declared (line 3), its size and dimensionality specified (line 5), its layers declared with a uniform distribution of nodes (lines 6,8), and its nodes initialized with the default parameter sets (lines 7,9). Finally, a functor, "`centerSurround`," is employed that connects the `Cortex`'s own "`excitatory`" and "`inhibitory`" layers (specified as "`.[]`" in the first two arguments of `centerSurround`) using a geometric pattern specified in four list arguments (lines 10–13).

4.3 Simulation Results

Figure 3 shows the output from the DataExplorer™ module of our simulation after running this experiment. The left panel shows the pattern of neural activation in response to a single stimulus in both the inhibitory (leftmost) and excitatory (rightmost) layers of the cortical sheet model. The right panel shows the connections from the receptor sheet to the somatosensory cortex, after repeated stimulation of the digits. The synaptic strength of the connections is color-coded. The emergence of neuronal groups can be observed as clusters of high synaptic strength, in agreement with previous results [15].

5 Conclusion

This paper was motivated by the lack of adequate problem solving environments for implementing large scale simulations of neural systems. Consequently, this

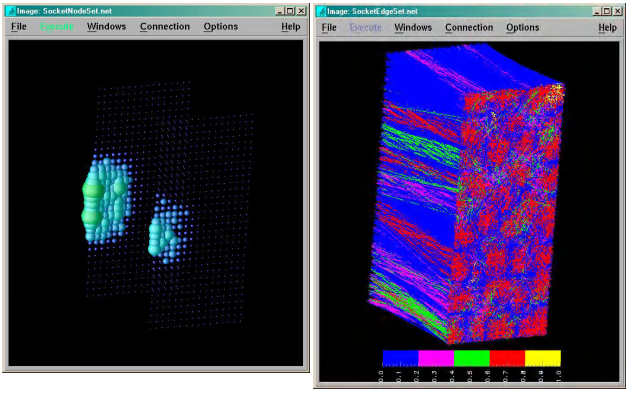


Fig. 3. Results of test simulation visualized with DataExplorer™.

paper has identified the challenges associated with meeting the needs of this domain. Based on these challenges, the Large-scale Edge Node Simulator environment was designed and implemented. It has been demonstrated that the environment is able to implement, execute, and analyze large scale, complex simulations of neural systems. Finally, the has shown that it is possible to meet the challenges of large scale simulations of neural systems and that the Large-scale Edge Node Simulator has done so.

Acknowledgements. The authors would like to acknowledge the contributions of Bill Arnold in the software development process of the Large-scale Edge Node Simulator.

References

1. See http://rprcsgi.rprc.washington.edu/neuronames/index1.html
2. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117:500–544 (1952).
3. Schiff, S.J.: Forecasting brain storms, Nat. Med. 4(10):1117–8 (1998).
4. Reeke, G.N., Edelman, G. M.: Selective Networks and Recognition Automata, Annals New York Academy of Sciences. 426:181–201 (1984).
5. Pearson, J.C., Finkel, L.H., Edelman, G.H.: Plasticity in the organization of adult cerebral cortical maps: a computer simulation based on neuronal group selection, J. Neuroscience, 7(12): 4209–4233, December 1987.
6. Merzenich, M.M., Kaas, J.H., Wall, J.T., Neson, R.J., Sur, M., Felleman, D.J.: Topographic reorganization of somatosensory cortical ares 3b and 1 in adult monkeys following restricted deafferentation, Neuroscience 10:639–665, 1983.

Monotone Iterative Methods and Schwarz Methods for Nonlinear Parabolic PDE with Time Delay

Daoud S. Daoud

Department of Mathematics, Eastern Mediterranean Univ., Famaqusta, North Cyprus - Mersin 10 Turkey
daoud.daoud@emu.edu.tr

Abstract. Its well known that the Schwarz alternating method proved to be feasible and powerful approach to solve elliptic or parabolic PDEs over multi overlapped subdomains. Recently Schwarz method proved to be very effective method when embedded within a well established solution methods such as monotone iterative method(or the method of lower and upper solution method). In this work we present the proofs of convergence of additive and multiplicative Schwarz alternating method for non linear parabolic equation where the reaction function involves a time delay function.

1 Introduction

In several applications its useful and even essential to decompose the domain of the PDE into several subdomains overlapped and non overlapped. The concept of decomposing the domain has been utilized in connections with the numerical approximation of the problem and the parallel processing algorithms. The earliest known domain decomposition method was presented by H.A. Schwarz in 1870.

The classical alternating Schwarz method have been utilized efficiently to solve the PDE over several subdomains and then the method has gained its popularity specially after the developments of the parallel computer architecture.

Recently the method of lower and upper solution and its associated monotone iterative method have been considered to develop analytical and numerical solution algorithms. The method has found to be a powerful tool for treatment of several type of PDEs like elliptic , time dependent parabolic problem see Pao [7, 8], and time dependent parabolic problem which involves time delay reaction function see the 'works by X-Lu [2, 3](The method is also called monotone iterative method).

On the other hand the method of lower and upper solution has also been considered with the use of the Schwarz iterative method see Lui [4]-[6], Lui studied the use of the monotone type of iterative methods in solving non linear Elliptic[4], and Parabolic equations with Schwarz methods [6].

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1137–1148, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The solution of the time delay problem by the lower and upper solution method and its associated monotone iterative method also has been studied numerically and analytically by Feng and Lu [1], and Lu [2, 3] and the references in Pao [7].

In Pao [7] presented a comprehensive study and complete bibliography for the monotone iterative method and the application for the time delay problem.

This article is an extension of the analysis by Lui [6] to study the Schwarz method with monotone iteration to solve the time dependent parabolic problem where the reaction function involves a time delay function such that the considered type of problem will fulfill the analysis by the monotone method see [7].

In section 3 we present some preliminaries notations and the definition of the model problem and in section 4 we present the proof for the multiplicative and the additive Schwarz method when the reaction function involves a monotone non decreasing time delay function. In section 5 we present the proofs of the Schwarz methods when the reaction function involves a monotone non increasing time delay function and the conclusion in section 6.

2 Preliminaries

Consider the following type of parabolic boundary value problem where the reaction function involves a time delay function given by

$$u_t - \Delta u = f(t,x,u(t,x)) + g(t,x,u(t-r,x)) \quad \text{on} \quad D = [0,T] \times \Omega, \tag{1}$$

with

$$u(t,x) = h \quad \text{on } S, \text{ and } u(t,x) = \eta_0(t,x) \text{ on } D_{(-r)},$$

and $D_{(-r)} = [-r,0] \times \Omega$ where Ω is assumed to be a bounded domain in $\mathbf{R}^N$. We assume that f, g are Hölder continuous and $\eta_0(t,x)$ is also Hölder continuous on $D_{(-r)}$. We also assume that Ω is decomposed into m, $m \geq 2$, subdomains with smooth boundaries such that $\Omega = \Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_m$.
We are interested in the solution of the parabolic PDE (1) for the time interval $[-r,T]$ for some fixed positive T.

In this area a couple of well known examples of the equation (1) are the model problems known as the diffusive logistics equation and Fisher's diffusive equation in population genetics [7]. Define $D_i = (0,T) \times \Omega_i$, $S = (0,T) \times \partial\Omega$, $S_i = (0,T) \times \Omega_i$, $D_{(-r)} = [-r,0] \times \Omega$, and $D_{i,(-r)} = [-r,0] \times \Omega_i$.

Let X denote the space of functions in $C(\overline{E_T})$, $E_T = [-r,T] \times \Omega$, which are continuously differentiable in time and twice differentiable in Ω, and we are interested in the solutions of PDE's in this space. The pair of functions $(\underline{u}, \overline{u})$ is said to be ordered if $\underline{u} \leq \overline{u}$ in E_T.

Let $\mathcal{A}$ be the sector of smooth functions defined by

$$\mathcal{A} = \{u \in X : \underline{u} \leq u \leq \overline{u} \text{ on } \overline{D}\}.$$

In addition suppose there exist some non negative continuous functions $\underline{c}$, and $\overline{c}$ defined on $\overline{D}$ by

$$\underline{c}(t,x) = sup\{-f_u(t,x,u); \underline{u} \leq \overline{u}\}, \text{ and}$$
$$\overline{c}(t,x) = sup\{f_u(t,x,u); \underline{u} \leq \overline{u}\}$$

such that

$$\underline{c}(u-v) \leq (f(t,x,u)+g(t,x,u(t-r,x)))-(f(t,x,v)+g(t,x,v(t-r,x))) \leq \overline{c}(u-v)$$

for any $v \leq u \in \mathcal{A}$.

The function g in (1) is either a monotone nondecreasing or monotone nonincreasing function in η for η in the sector of upper and lower solutions [7]. An essential theorems in this study are the following maximum principles theorems

Theorem 1 (Maximum Principle Theorem). *Let $w \in X$ satisfy*

$$w_t - \Delta w + \underline{c} \geq 0 \text{ on } D, w \geq 0 \text{ on } S, \text{ and } w(0,x) \geq 0 \text{ on } \Omega.$$

Then $w \geq 0$ on $\overline{D}$.

In cases when the functions are less smooth or the domain is with non smooth boundary a generalized maximum principle theorem is also required see [6], given by the following generalized form.

Theorem 2 (Generalized Maximum Principle). *Suppose Ψ is an open set in $\mathbf{R}^N$ and $\Psi_T = (-r,T) \times \Psi$, let $w \in H^1(-r,T; H^1(\Psi) \cap C(\overline{\Psi}))$ and satisfy*

$$\int_{\Psi_T} (\nabla w.\nabla\phi + w_t\phi + \underline{c}w\phi) \geq 0, \text{ for non negative } \phi \in H^1(-r,T; H^1(\Psi) \cap C(\overline{\Psi}))$$

and $w(0) \geq 0$ on Ψ. Then $w \geq 0$ on $\overline{\Psi_T}$.

The study in this article is divided into two parts, the first part section 4 is dealing with the case when g is a monotone nonincreasing function and the second part section 5 is dealing with the case when g is a monotone nondecreasing function.

3 Monotone Nonincreasing Function

This section investigate the application of Schwarz methods to the parabolic PDEs (1), where the reaction function involves a monotone nonincreasing time delay function $g(t,x,u(t-r,x))$. In this case the generated sequence of lower and upper solutions are coupled pair of sequence and are defined as follows.

Definition 1. *Let $g(.\eta)$ be a monotone nonincreasing in η. Then a pair of function $\underline{u}, \overline{u} \in X$ are called coupled upper and lower solutions of PDE (1), if $\overline{u} \geq \underline{u}$ in X, and they satisfy the inequalities;*

$$\overline{u}_t - \Delta\overline{u} \geq f(\overline{u}) + g(\underline{u}(t-r)) \text{ and}$$
$$\underline{u}_t - \Delta\underline{u} \leq f(\underline{u}) + g(\overline{u}(t-r)) \text{ in } D_T$$

with $\underline{u} < h < \overline{u}$ on S, and $\underline{u} \leq \eta_0(t,x) \leq \overline{u}$ on $D_{(-r)}$.

For a given pair of coupled upper and lower solutions $\overline{u}$, $\underline{u}$ and using $\overline{u}^{(0)} = \overline{u}$, and $\underline{u}^{(0)} = \underline{u}$ we construct two sequences of $\{\overline{u}^{(k)}, \underline{u}^{(k)}\}$ from the following iterative process

$$\begin{aligned} \overline{u}_t^{(k)} - \Delta\overline{u}^{(k)} + \underline{c}\,\overline{u}^{(k)} &= \underline{c}\,\overline{u}^{(k-1)} + f(\overline{u}^{(k-1)}) + g(\underline{u}^{(k-1)}) \\ \underline{u}_t^{(k)} - \Delta\underline{u}^{(k)} + \underline{c}\,\underline{u}^{(k)} &= \underline{c}\,\underline{u}^{(k-1)} + f(\underline{u}^{(k-1)}) + g(\overline{u}^{(k-1)}) \end{aligned} \tag{2}$$

subject to the following boundary and initial conditions

$$\overline{u}^{(k)} = h, \quad \underline{u}^{(k)} = h \text{ on } S, \quad \overline{u}^{(k)} = \eta_0(t,x), \quad \text{and} \quad \underline{u}^{(k)} = \eta_0(t,x) \text{ on } D_{(-r)}.$$

The generated sequence of pair of solutions $\{\underline{u}^{(k)}, \overline{u}^{(k)}\}$ are well defined and monotone. Furthermore the generated sequences convergence monotically to the unique solution of (1) when $g_u \leq 0$ (Theorem 8.2 in [7]).

For the above definition of sequences of pair of solutions corresponding to the monotone nonincreasing function g we present the convergence of sequence(pair) of solutions generated from the Multiplicative Schwarz method by the following theorem;

Theorem 3. *Let* $\overline{u}^{(0)} = \overline{u}^{(-1/2)} = \overline{u}$ *and* $\underline{u}^{(0)} = \underline{u}^{(-1/2)} = \underline{u}$ *on* $\overline{D}$, *with* $\underline{u} = h$, and $\overline{u} = h$ *on* S, *and* $\overline{u} = \underline{u} = \eta_0$ *on* $D_{(-r)}$.
The Schwarz sequence for $n \geq 0$, *is defined by*

$$\begin{aligned} (\partial_t - \Delta + \underline{c})\,\overline{u}^{(n+1/2)} &= f(\overline{u}^{(n-1/2)}) + \underline{c}\,\overline{u}^{(n-1/2)} + g(\underline{u}^{(n-1/2)}) \\ (\partial_t - \Delta + \underline{c})\,\underline{u}^{(n+1/2)} &= f(\underline{u}^{(n-1/2)}) + \underline{c}\,\underline{u}^{(n-1/2)} + g(\overline{u}^{(n-1/2)}) \end{aligned} \quad \text{on } D_1, \tag{3}$$

with

$$\begin{aligned} \overline{u}^{(n+1/2)} &= \overline{u}^{(n)} \\ \underline{u}^{(n+1/2)} &= \underline{u}^{(n)} \end{aligned} \quad \text{on } S_1, \text{ and } \begin{aligned} \overline{u}^{(n+1/2)} &= \eta_0 \\ \underline{u}^{(n+1/2)} &= \eta_0 \end{aligned} \quad \text{on } D_{1,(-r)},$$

and

$$\begin{aligned} (\partial_t - \Delta + \underline{c})\,\overline{u}^{(n+1)} &= f(\overline{u}^{(n)}) + \underline{c}\,\overline{u}^{(n)} + g(\underline{u}^{(n)}) \\ (\partial_t - \Delta + \underline{c})\,\underline{u}^{(n+1)} &= f(\underline{u}^{(n)}) + \underline{c}\,\underline{u}^{(n)} + g(\overline{u}^{(n)}) \end{aligned} \quad \text{on } D_2, \tag{4}$$

$$\begin{aligned} \overline{u}^{(n+1)} &= \overline{u}^{(n+1/2)} \\ \underline{u}^{(n+1)} &= \underline{u}^{(n+1/2)} \end{aligned} \quad \text{on } S_2, \text{ and } \begin{aligned} \overline{u}^{(n+1)} &= \eta_0 \\ \underline{u}^{(n+1)} &= \eta_0 \end{aligned} \quad \text{on } D_{2,(-r)}.$$

Here $\underline{u}^{(n+1/2)}$ and $\overline{u}^{(n+1/2)}$ *are defined as* $\underline{u}^{(n)}$ and $\overline{u}^{(n)}$ *on* $\overline{D}\backslash\overline{D}_1$ *respectively, and* $\underline{u}^{(n+1)}$ and $\overline{u}^{(n+1)}$ *are defined as* $\underline{u}^{(n+1/2)}$ and $\overline{u}^{(n+1/2)}$ *on* $\overline{D}\backslash\overline{D}_2$, *respectively. Then* $\underline{u}^{(n+i/2)} \to u$ *and* $\overline{u}^{(n+i/2)} \to u$ *on* D_i , $i = 1, 2$, *where* u *is the solution of (1) in* $\mathcal{A}$.

Proof. To show the convergence of the lower and upper solution sequence to the solution u on D_i, we start to prove that the pair of coupled lower and upper sequence is monotone *i.e.* to show that

$$\underline{u} \leq \underline{u}^{(n-1/2)} \leq \underline{u}^{(n)} \leq \underline{u}^{(n+1/2)} \leq \overline{u}^{(n+1/2)} \leq \overline{u}^{(n)} \leq \overline{u}^{(n-1/2)} \leq \overline{u} \text{ on } \overline{D}, \quad (5)$$

For $n = 0$, consider the sequence on D_1 for $\overline{u}^{(1/2)}$ and the upper solution $\overline{u}^{(0)}$

$$\begin{aligned} (\partial_t - \Delta + \underline{c})\overline{u}^{(1/2)} &= f(\overline{u}^{(0)}) + \underline{c}\overline{u}^{(0)} + g(\underline{u}^{(0)}) \\ (\partial_t - \Delta + \underline{c})\overline{u}^{(0)} &\geq f(\overline{u}^{(0)}) + \underline{c}\overline{u}^{(0)} + g(\underline{u}^{(0)}) \end{aligned} \quad (6)$$

On subtraction of the above two results we obtain

$$(\partial_t - \Delta + \underline{c})(\overline{u}^{(0)} - \overline{u}^{(1/2)}) \geq 0.$$

Since $\overline{u}^{(0)} - \overline{u}^{(1/2)} = 0$ on S_1, and $\overline{u}^{(0)} - \overline{u}^{(1/2)} = \overline{u}^{(0)} - \eta_0(t,x) \geq 0$ on $D_{1,(-r)}$, which implies that $\overline{u}^{(0)} - \eta_0(0,x) \geq 0$ on Ω.
Then by the maximum principle theorem 1 we conclude that $\overline{u}^{(0)} \geq \overline{u}^{(1/2)}$ on $\overline{D_1}$, which also hold on $\overline{D}$.
For the lower solution the proof is similar to the above and it follows by reversing the second inequality in(6), concluding that $\underline{u}^{(0)} \leq \underline{u}^{(1/2)}$. To proceed in the proof of (5) it requires the proof of the following pair of couple of inequalities

$$\begin{aligned} \overline{u}^{(n+1/2)} &\leq \overline{u}^{(n-1/2)} \\ \underline{u}^{(n-1/2)} &\leq \underline{u}^{(n+1/2)} \end{aligned} \quad (7)$$

on D_1, and on D_2 the following inequalities

$$\begin{aligned} \overline{u}^{(n+1)} &\leq \overline{u}^{(n)} \\ \underline{u}^{(n)} &\leq \underline{u}^{(n+1)}. \end{aligned} \quad (8)$$

On the subdomain D_1 for $n = 0$ and for both of the inequalities in (7) they follows from the definition.

On D_2 for $n = 0$ consider the defining equation of $\overline{u}^{(1)}$

$$(\partial_t - \Delta + \underline{c})\overline{u}^{(1)} = \underline{c}\overline{u}^{(0)} + f(\overline{u}^{(0)} + g(\underline{u}^{(0)}),$$

and the equation for $\overline{u}^{(0)}$

$$(\partial_t - \Delta + \underline{c})\overline{u}^{(0)} \geq \underline{c}\overline{u}^{(0)} + f(\overline{u}^{(0)}) + g(\underline{u}^{(0)}).$$

From the subtraction the above relations we obtain;

$$(\partial_t - \Delta + \underline{c})(\overline{u}^{(0)} - \overline{u}^{(1)}) \geq 0$$

on D_2 with $\overline{u}^{(0)} - \overline{u}^{(1)} = 0$ on S_2, and $\overline{u}^{(0)} - \overline{u}^{(1)} \geq 0$ on $D_{2,(-r)}$, then by the maximum principle theorem it follows that $\overline{u}^{(0)} \geq \overline{u}^{(1)}$.

For the lower solution sequence over D_2 it follows similarly by considering the defining equations for $\underline{u}^{(1)}$ and the equation for the lower solution sequence $\underline{u}^{(0)}$.

Assume that the inequalities (7,8) are holds for n, to prove they holds for n replaced by $(n+1)$, and we firstly consider the case of (7).

Consider (3) for $\overline{u}^{(n+3/2)}$, and $\overline{u}^{(n+1/2)}$

$$(\partial_t - \Delta + \underline{c})\overline{u}^{(n+3/2)} = f(\overline{u}^{(n+1/2)}) + \underline{c}\overline{u}^{(n+1/2)} + g(\underline{u}^{(n+1/2)})$$
$$(\partial_t - \Delta + \underline{c})\overline{u}^{(n+1/2)} = f(\overline{u}^{(n-1/2)}) + \underline{c}\overline{u}^{(n-1/2)} + g(\underline{u}^{(n-1/2)}).$$

On subtraction the above equalities with the consideration of the definition of $\underline{c}$, $g_u \leq 0$ and the induction assumption, it follows that;

$$(\partial_t - \Delta + \underline{c})(\overline{u}^{(n+1/2)} - \overline{u}^{(n+3/2)}) =$$
$$(f_u + \underline{c})(\overline{u}^{(n-1/2)} - \overline{u}^{(n+1/2)}) - g_u(\underline{u}^{(n+1/2)} - \underline{u}^{(n-1/2)}) \geq 0,$$

Since $\overline{u}^{(n+3/2)} = \overline{u}^{(n+1)}$ on S_1, $\overline{u}^{(n+1)} \leq \overline{u}^{(n)}$ by induction hypothesis and $\overline{u}^{(n)} = \overline{u}^{(n+1/2)}$ on S_1, then it follows that $(\overline{u}^{(n+1/2)} - \overline{u}^{(n+3/2)}) \geq 0$ on S_1, and $(\overline{u}^{(n+1/2)} - \overline{u}^{(n+3/2)}) = 0$ on $D_{1,(-r)}$.

Therefore by the maximum principle theorem 1 it follows that $\overline{u}^{(n+1/2)} - \overline{u}^{(n+3/2)} \geq 0$ on $\overline{D_1}$ concluding that $\overline{u}^{(n+1/2)} \geq \overline{u}^{(n+3/2)}$ on $\overline{D_1}$.
To prove $\underline{u}^{(n+1/2)} \leq \underline{u}^{(n+3/2)}$. Consider (3) for $\underline{u}^{(n+1/2)}$ and $\underline{u}^{(n+3/2)}$

$$(\partial_t - \Delta + \underline{c})\underline{u}^{(n+3/2)} = f(\underline{u}^{(n+1/2)} + \underline{c}\ \underline{u}^{(n+1/2)} + g(\overline{u}^{(n+1/2)})$$
$$(\partial_t - \Delta + \underline{c})\underline{u}^{(n+1/2)} = f(\underline{u}^{(n-1/2)} + \underline{c}\ \underline{u}^{(n-1/2)} + g(\overline{u}^{(n-1/2)}),$$

by subtracting the above two equations we obtain

$$(\partial_t - \Delta + \underline{c})(\underline{u}^{(n+3/2)} - \underline{u}^{(n+1/2)}) =$$
$$(f_u + \underline{c})(\underline{u}^{(n+1/2)} - \underline{u}^{(n-1/2)}) - g_u(\overline{u}^{(n-1/2)} - \overline{u}^{(n+1/2)}) \geq 0.$$

The last inequality follows from the induction assumptions, the definition of $\underline{c}$, and $g_u \leq 0$.

Since $\underline{u}^{(n+3/2)} = \underline{u}^{(n+1)}$ on S_1, $\underline{u}^{(n+1)} \geq \underline{u}^{(n)}$ by induction hypothesis , and $\underline{u}^{(n)} = \underline{u}^{(n+1/2)}$ on S_1, then $\underline{u}^{(n+3/2)} - \underline{u}^{(n+1/2)} \geq 0$ on S_1, and $\underline{u}^{(n+3/2)} - \underline{u}^{(n+1/2)} = 0$ on $D_{1,(-r)}$, then it follows that $\underline{u}^{(n+3/2)} - \underline{u}^{(n+1/2)} = 0$ on Ω_1. By the maximum principle theorem 1 implies that $\underline{u}^{(n+3/2)} - \underline{u}^{(n+1/2)} \geq 0$ on $\overline{D_1}$, and then concluding that $\underline{u}^{(n+3/2)} \geq \underline{u}^{(n+1/2)}$ on $\overline{D_1}$.

Similarly we could prove the inequalities (8) for D_2 when n replaced by $n+1$.

Then assume (5)is hold for n and to prove it is true for n replaced by $n+1$.

On $D_1 \setminus D_2$ we have $\underline{u}^{(n+1)} = \underline{u}^{(n+1/2)}$ on S_2 by definition and on $D_1 \cap D_2$ subtract the defining equations for $\underline{u}^{(n+1/2)}$ and $\underline{u}^{(n+1)}$ to obtain

$$(\partial_t - \Delta + \underline{c})(\underline{u}^{(n+1)} - \underline{u}^{(n+1/2)}) = (f_u + \underline{c})(\underline{u}^{(n)} - \underline{u}^{(n-1/2)}) - g_u(\overline{u}^{(n-1/2)} - \overline{u}^{(n)}) \geq 0.$$

The above inequality hold by the definition of $\underline{c}$, $g_u \leq 0$, and the induction hypothesis. In case that $D_1 \cap D_2$ is not smooth we can multiply the above inequality by anon negative function $\phi \in H^1(-r, T; H_0^1(\Omega_1 \cap \Omega_2))$ and integrate by parts to obtain

$$\int_{D_1 \cap D_2} (\underline{u}^{(n+1)} - \underline{u}^{(n+1/2)})_t \phi + \nabla(\underline{u}^{(n+1)} - \underline{u}^{(n+1/2)}).\nabla\phi + \underline{c}(\underline{u}^{(n+1)} - \underline{u}^{(n+1/2)})\phi \geq 0.$$

On $S_1 \cap D_2$ we have $\underline{u}^{(n+1)} \geq \underline{u}^n = \underline{u}^{(n+1/2)}$ by (7), and by the definition, and on $S_2 \cap D_1$ we have $\underline{u}^{(n+1)} = \underline{u}^{(n+1/2)}$. Hence $\underline{u}^{(n+1)} - \underline{u}^{(n+1/2)} \geq 0$ on $D \cap (S_1 \cup S_2)$, and also by definition $\underline{u}^{(n+1)}(t,x) - \underline{u}^{(n+1/2)}(t,x) = 0$ on $D_{1,(-r)} \cap D_{2,(-r)}$ (*i.e.* $\underline{u}^{(n+1)}(0,x) - \underline{u}^{(n+1/2)}(0,x) = 0$ on $\Omega_1 \cap \Omega_2$).

By the generalized maximum principle theorem 2 we conclude that $\underline{u}^{(n+1)} \geq \underline{u}^{(n+1/2)}$ on $\overline{D_1 \cap D_2}$. On $D_2 \setminus D_1$ we have $\underline{u}^{(n+1/2)} = \underline{u}^{(n)} \leq \underline{u}^{(n+1)}$, then we have $\underline{u}^{(n+1)} \geq \underline{u}^{(n+1/2)}$ on $\overline{D}$.

Following the same way we can prove that $\overline{u}^{(n+3/2)} \leq \overline{u}^{(n+1)}$ on $\overline{D}$.

We proved that the sequence of solutions are bounded above and below therefore the limits are well defined and exist on $\overline{D}$, given by;

$$\lim_{n\to\infty} \underline{u}^{(n+1/2)} = \lim_{n\to\infty} \overline{u}^{(n+1/2)} = u_1, \quad \text{and} \quad \lim_{n\to\infty} \underline{u}^{(n)} = \lim_{n\to\infty} \overline{u}^{(n)} = u_2.$$

Then by theorem 8.2 in Pao [7] it follows that u_1 and u_2 satisfies the PDE on D_1 and on D_2 respectively and its easily to conclude that $u_1 = u_2$ on $\overline{D}$, and that is completes the proof of the theorem.

The monotone non increasing additive Schwarz method leads to the following convergence property of the pair coupled lower and upper solution sequence, given by the following theorem.

Theorem 4. *Let* $\underline{u}^{(0)} = \underline{u}_i^{(0)} = \underline{u}$, and $\overline{u}^{(0)} = \overline{u}_i^{(0)} = \overline{u}$ *on* $\overline{D}$, *for* $i = 1, \ldots, m$ *with* $\underline{u} = h$, and $\overline{u} = h$ *on* S. *The additive Schwarz sequence for the pair of coupled upper and lower solutions* ($n \geq 1$) *is defined by;*

$$\begin{aligned} (\partial_t - \Delta + \underline{c})\underline{u}_i^{(n)} &= f(\underline{u}_i^{(n-1)}) + \underline{c}\ \underline{u}_i^{(n-1)} + g(\overline{u}_i^{(n-1)}) \\ (\partial_t - \Delta + \underline{c})\overline{u}_i^{(n)} &= f(\overline{u}_i^{(n-1)}) + \underline{c}\ \overline{u}_i^{(n-1)} + g(\underline{u}_i^{(n-1)}) \end{aligned} \quad \text{on } D_i, \tag{9}$$

subject to;

$$\underline{u}_i^{(n)} = \underline{u}^{(n-1)}, \text{ and } \overline{u}_i^{(n)} = \overline{u}^{(n-1)},$$

on S_i, *and*

$$\underline{u}_i^{(n)}(t,x) = \eta_0(t,x), \text{ and } \overline{u}_i^{(n)}(t,x) = \eta_0(t,x)$$

on $D_{i,(-r)}$.
Here $\underline{u}_i^{(n)} = \underline{u}^{(n-1)}$, and $\overline{u}_i^{(n)} = \overline{u}^{(n-1)}$, *on* $\overline{D} \setminus \overline{D_i}$, *and*

$$\underline{u}^{(n)}(t,x) = \max_i \{\underline{u}_i^{(n)}(t,x)\}, \text{ and} \tag{10}$$

$$\overline{u}^{(n)}(t,x) = \min_i \{\overline{u}_i^{(n)}(t,x)\}, \text{ for} (t,x) \in \overline{D}. \tag{11}$$

$\underline{u}_i^{(n)} \to u$, and $\overline{u}_i^{(n)} \to u$, *on* D_i, $i = i, \ldots, m$ *where* u *is the solution of (1) in* $\mathcal{A}$.

The proof of theorem 4 relies on few lemmas, we will presents the required lemmas with their proofs firstly and the proof of theorem 4 will be presented afterwards.

Lemma 1. *Let g be a monotone nonincreasing function given by definition 1 and let $\{\underline{u}^n, \overline{u}^n\}$ be a sequence of lower and upper solution satisfying $\underline{u}^n \leq \underline{u}^{n+1} \leq \overline{u}^{n+1} \leq \overline{u}^n$ on $D_i \subset D$. If condition (10) in theorem 4 holds and if the following PDE*

$$(\partial_t - \Delta + \underline{c})\underline{h}^{(n)} = f(\underline{u}^{(n-1)}) + \underline{c}\ \underline{u}^{(n-1)} + g(\overline{u}^{(n-1)}) \tag{12}$$

holds on D_i, and subject to

$$\underline{h}^{(n)} = \underline{u}^{(n)} \text{ on } S_i, \text{ and } \underline{h}^{(n)}(t,x) = \eta_0(t,x) \text{ on } D_{i,(-r)}, \quad \text{for} n \geq 0.$$

such that $\underline{u}^{(-1)} = \underline{u}^{(0)}$. Then

$$\underline{h}^{(n+1)} \geq \underline{h}^{(n)} \text{ on } D_i, \tag{13}$$

and

$$\underline{h}^{(n)} \geq \underline{u}^{(n)} \text{ on } D_{i,(-r)}. \tag{14}$$

Lemma 2. *Let g be a monotone nonincreasing function given by definition 1 and let $\{\underline{u}^n, \overline{u}^n\}$ be a sequence of lower and upper solution satisfying $\underline{u}^n \leq \underline{u}^{n+1} \leq \overline{u}^{n+1} \leq \overline{u}^n$ on $D_i \subset D$. If condition (11) in theorem 4 holds and if the following PDE*

$$(\partial_t - \Delta + \underline{c})\overline{h}^{(n)} = f(\overline{u}^{(n-1)}) + \overline{c}\ \overline{u}^{(n-1)} + g(\underline{u}^{(n-1)}) \tag{15}$$

holds on D_i, and subject to

$$\overline{h}^{(n)} = \overline{u}^{(n)} \text{ on } S_i, \text{ and } \overline{h}^{(n)}(t,x) = \eta_0(t,x) \text{ on } D_{i,(-r)}, \quad \text{for} n \geq 0.$$

such that $\overline{u}^{(-1)} = \overline{u}^{(0)}$. Then

$$\overline{h}^{(n+1)} \geq \overline{h}^{(n)} \text{ on } D_i, \tag{16}$$

and

$$\overline{h}^{(n)} \geq \overline{u}^{(n)} \text{ on } D_{i,(-r)}. \tag{17}$$

The other necessary lemma for the proof of theorem 4 is the following lemma.

Lemma 3. *Let the hypothesis in theorem 4 hold. Then over each subdomain $D_i \subset D$ the following integrals satisfied,*
1-

$$\int_{D_i} \nabla \underline{u}^{(n)}.\nabla\phi + \underline{u}_t^{(n)}\phi + \underline{c}\underline{u}^{(n)}\phi \leq \left(\int_{D_i} f(\underline{u}^{(n-1)}) + \underline{c}\ \underline{u}^{(n-1)} + g(\overline{u}^{(n-1)})\right)\phi \tag{18}$$

for all nonnegative $\phi \in H^1(-r,T;H_0^1(\Omega_i))$.
2-

$$\int_{D_i} \nabla \overline{u}^{(n)}.\nabla\phi + \overline{u}_t^{(n)}\phi + \underline{c}\,\overline{u}^{(n)}\phi \geq \int_{D_i} (f(\overline{u}^{(n-1)}) + \underline{c}\overline{u}^{(n-1)} + g(\underline{u}^{(n-1)}))\phi \quad (19)$$

for all non negative $\phi \in H^1(-r,T;H_0^1(\Omega_i) \cap C(\overline{\Omega}))$.

In the following we will present the proof of theorem 4.

Proof. For the proof of theorem 4, it requires to justify the following monotone properties of the coupled pair of lower and upper solution,

$$\underline{u} \leq \underline{u}_i^{(n)} \leq \underline{u}_i^{(n+1)} \leq \overline{u}_i^{(n+1)} \leq \overline{u}_i^{(n)} \leq \overline{u} \quad \text{on } D_i, \quad (20)$$

$$\underline{u} \leq \underline{u}^{(n)} \leq \underline{u}^{(n+1)} \leq \overline{u}^{(n+1)} \leq \overline{u}^{(n)} \leq \overline{u} \quad \text{on } \overline{D}, \quad (21)$$

and

$$\underline{u}^{(n)} \leq \underline{u}_i^{(n+1)} \leq \overline{u}_i^{(n+1)} \leq \overline{u}^{(n)} \text{ on } \overline{D}. \quad (22)$$

The proof of (20, 21) is by induction and it is easy to show that they holds for $n = 0$ which follows from the definition.

Assume that (20,21) are holds for n, to prove they are true for n replaced by $n+1$, firstly is to show that (20) satisfies the following inequalities

$$\underline{u} \leq \underline{u}_i^{(n+1)} \leq \underline{u}_i^{(n+2)} \leq \overline{u}_i^{(n+2)} \leq \overline{u}_i^{(n+1)} \leq \overline{u} \quad \text{on } D_i.$$

Consider the iteration given by (9) for $\underline{u}_i^{(n+1)}$ and $\underline{u}_i^{(n+2)}$, respectively

$$\begin{aligned}(\partial_t - \Delta + \underline{c})\underline{u}_i^{(n+1)} &= f(\underline{u}_i^{(n)}) + \underline{c}\,\underline{u}_i^{(n)} + g(\overline{u}_i^{(n)}) \\ (\partial_t - \Delta + \underline{c})\underline{u}_i^{(n+2)} &= f(\underline{u}_i^{(n+1)}) + \underline{c}\,\underline{u}_i^{(n+1)} + g(\overline{u}_i^{(n+1)}).\end{aligned}$$

On subtraction of the above equations we have

$$(\partial_t - \Delta + \underline{c})(\underline{u}_i^{(n+2)} - \underline{u}_i^{(n+1)}) = (\underline{c} + f_u)(\underline{u}_i^{(n+1)} - \underline{u}_i^{(n)}) - g_u(\overline{u}_i^{(n)} - \overline{u}_i^{(n+1)}) \geq 0.$$

The above follows from induction assumption and $g_u \leq 0$.

Since $(\underline{u}_i^{(n+2)} - \underline{u}_i^{(n+1)}) = (\underline{u}^{(n+1)} - \underline{u}^{(n)}) \geq 0$ on S_i and also we have $(\underline{u}_i^{(n+2)} - \underline{u}_i^{(n+1)}) = 0$ on $D_{i,(-r)}$.

Then $(\underline{u}_i^{(n+2)} - \underline{u}_i^{(n+1)}) = 0$ on Ω_i, and then by the maximum principle theorem concluding that $\underline{u}_i^{(n+2)} - \underline{u}_i^{(n+1)} \geq 0$ on $\overline{D_i}$ ($\underline{u}_i^{(n+2)} \geq \underline{u}_i^{(n+1)}$ on $\overline{D_i}$). Similarly we can prove $\overline{u}_i^{(n+1)} \geq \overline{u}_i^{(n+2)}$ on $\overline{D}_i$, then (20) holds for any n.

To prove (21) for n replaced by $n+1$, and $(t,x) \in \overline{D}$. For the lower solution sequence there exist an integer i, $1 \leq i \leq m$ such that $\underline{u}^{(n+1)}(t,x) = \underline{u}_i^{(n+1)} \leq \underline{u}_i^{(n+2)} \leq \underline{u}^{(n+2)} \leq \overline{u}^{(n+2)}$.
The above inequalities follows by definition of $u^{(n+1)}$ and the upper and lower solution. For the upper solution sequence, there exist an integer k, $1 \leq k \leq m$ such that

$$\overline{u}^{(n+1)} = \overline{u}_k^{(n+1)} \geq \overline{u}_k^{(n+2)} \geq \overline{u}^{(n+1)}.$$

Therefore

$$\underline{u} \le \underline{u}^{(n)} \le \underline{u}^{(n+1)} \le \overline{u}^{(n+1)} \le \overline{u}^{(n)} \le \overline{u}, \quad \text{on } \overline{D}.$$

The proof of inequality(22) is by induction as well. For $n = 0$, the inequality follows from the definition. Assume (22) holds with n replaced by $(n-1)$, *i.e.*

$$\underline{u}^{(n-1)} \le \underline{u}_i^{(n)} \le \overline{u}_i^{(n)} \le \overline{u}^{(n-1)} \text{ on } \overline{D}.$$

To prove it holds for n, *i.e.* $\underline{u}^{(n)} \le \underline{u}_i^{(n+1)} \le \overline{u}_i^{(n+1)} \le \overline{u}^{(n)}$. Consider the first inequality of lemma 3 given by;

$$\int_{D_i} \nabla \underline{u}^{(n)}.\nabla\phi + \underline{u}_t^{(n)}\phi + \underline{c}\underline{u}^{(n)}\phi \le (\int_{D_i} f(\underline{u}^{(n-1)}) + \underline{c}\,\underline{u}^{(n-1)} + g(\overline{u}^{(n-1)}))\phi \quad (23)$$

for all nonnegative $\phi \in H^1(-r,T;H_0^1(\Omega_i))$.

Next multiply the defining equation for $\underline{u}_i^{(n+1)}$, by a nonnegative function $\phi \in H^1(-r,T;H_0^1(\Omega_i))$ to obtain

$$\int_{D_i} \nabla \underline{u}_i^{(n+1)}.\nabla\phi + (\underline{u}_i^{(n+1)})_t\phi + \underline{c}\,\underline{u}_i^{(n+1)}\phi = \int_{\Omega_i} (f(\underline{u}_i^{(n)}) + \underline{c}\,\underline{u}_i^{(n)} + g(\overline{u}_i^{(n)}))\phi$$

Subtract (23) from the last equation to obtain;

$$\begin{aligned} &\textstyle\int_{D_i} \nabla(\underline{u}_i^{(n+1)} - \underline{u}^{(n)}).\nabla\phi + (\underline{u}_i^{(n+1)} - \underline{u}^{(n)})_t\phi + \underline{c}(\underline{u}_i^{(n+1)} - \underline{u}^{(n)})\phi \ge \\ &\textstyle\int_{D_i} ((f_u + \underline{c})(\underline{u}_i^{(n)} - \underline{u}^{(n-1)}) - g_u(\overline{u}_i^{(n)} - \overline{u}^{(n-1)}))\phi \ge 0 \end{aligned}$$

The last inequality follows from the induction assumption, the definition of $\underline{c}$, and $g_u \le 0$.

On S_i we have $\underline{u}_i^{(n+1)} = \underline{u}^{(n)}$ and on $D_{i,(-r)}$, we have $\underline{u}_i^{(n+1)} - \underline{u}^{(n)} = 0$, therefore by the generalized maximum principle we get $\underline{u}_i^{(n+1)} \ge \underline{u}^{(n)}$ on $\overline{D_i}$ and of course this inequality is also hold on $\overline{D}$.

To prove $\overline{u}^{(n)} \ge \overline{u}_i^{(n+1)}$ we consider the second inequality of lemma 3 given by ;

$$\int_{D_i} \nabla \overline{u}^{(n)}.\nabla\phi + \overline{u}_t^{(n)}\phi + \underline{c}\overline{u}^{(n)}\phi \ge \int_{D_i} f(\overline{u}^{(n-1)}) + \underline{c}\overline{u}^{(n-1)} + g(\underline{u}^{(n-1)}) \quad (24)$$

for all nonnegative $\phi \in H^1(-r,T;H_0^1(\Omega_i))$.

Next multiply the defining equation for $\overline{u}_i^{(n+1)}$, by anon negative function $\phi \in H^1(-r,T;H_0^1(\Omega_i))$ to obtain

$$\int_{D_i} \nabla \overline{u}_i^{(n+1)}.\nabla\phi + (\overline{u}_i^{(n+1)})_t\phi + \underline{c}\overline{u}_i^{(n+1)}\phi = \int_{\Omega_i} (f(\overline{u}_i^{(n)}) + \underline{c}\overline{u}_i^{(n)} + g(\underline{u}_i^{(n)}))\phi.$$

Subtract the last equation from (24), we obtain

$$\begin{aligned} &\textstyle\int_{D_i} \nabla(\overline{u}^{(n)} - \overline{u}_i^{(n+1)}).\nabla\phi + (\overline{u}^{(n)} - \overline{u}_i^{(n+1)})_t\phi + \underline{c}(\overline{u}^{(n)} - \overline{u}_i^{(n+1)})\phi \ge \\ &\textstyle\int_{D_i} ((f_u + \underline{c})(\overline{u}^{(n-1)} - \overline{u}_i^{(n)}) - g_u(\underline{u}^{(n-1)} - \underline{u}_i^{(n)}))\phi \ge 0. \end{aligned}$$

The above inequality follows from induction assumption, the definition of $\underline{c}$, and $g_u \leq 0$ Since $\overline{u}_i^{(n+1)} = \overline{u}^{(n)}$ on S_i, and $\overline{u}_i^{(n+1)} - \overline{u}^{(n)} = 0$ on $D_{i,(-r)}$, therefore by the generalized maximum principle theorem 1 we obtain

$$\overline{u}^{(n)} \geq \overline{u}_i^{(n+1)},$$

on $\overline{D_i}$ and its also holds on $\overline{D}$.

4 Monotone Non Decreasing Function

Definition 2. *Let $g(.,\eta)$ be a monotone non decreasing function in η. A function $\underline{u} \in X$ is called a lower solution of (1), if it satisfies the inequality*

$$\underline{u}_t - \Delta \underline{u} - f(t,x,\underline{u}) - g(t,x,\underline{u}(t-r,x)) \leq 0 \text{ on } D, \tag{25}$$

$\underline{u} \leq h$ on S, and $\underline{u}(0,x) \leq \eta_0(t,x)$ on $D_{(-r)}$. Similarly $\overline{u}$ is called an upper solution if it satisfies the reversed inequalities in(25).

For a given pair of ordered upper and lower solutions with $u^{(0)} = \overline{u}$ and $u^{(0)} = \underline{u}$ as two independent initial iterates for the lower and upper sequences and the respective sequences from the iteration process is given by

$$u_t^n - \Delta u^n + \underline{c} u^n = \underline{c} u^{(n-1)} + f(t,x,u^{(n-1)}) + g(t,x,u^{(n-1)}(t-r,x)) \text{ on} D, \tag{26}$$

with

$$u^n = h, \text{ on } S, \text{ and } u^n = \eta_0(t,x) \text{ on} D_{-r},$$

to produce the independent monotone sequences denoted by $\{\overline{u}^{(n)}\}$, and $\{\underline{u}^{(n)}\}$, respectively. The functions f and g are Hölder continuous functions, and $g_\eta \geq 0$, for $\eta \in < \underline{u}, \overline{u} > \in \mathcal{A}$.

The following theorem present the convergence of a multiplicative Schwarz sequences for the PDE (1) over two subdomains.

Theorem 5. *Let $u^{(0)} = u^{(-1/2)} = \underline{u}$ on $\overline{D}$, with $\underline{u} = h$ on S. Define the Schwarz sequence for $(n \geq 0)$ as follows ;*

$$\begin{aligned}
(\partial_t - \Delta + \underline{c}) u^{(n+1/2)} &= \underline{c} u^{(n-1/2)} + f(u^{(n-1/2)}) + g(u^{(n-1/2)}(t-r)) \text{ on } D_1, \\
u^{(n+1/2)} &= u^{(n)} \text{ on } S_1 \\
u^{(n+1/2)} &= \eta_0(t,x) \text{ on } D_{1,(-r)}
\end{aligned}$$

and

$$\begin{aligned}
(\partial_t - \Delta + \underline{c}) u^{(n+1)} &= \underline{c} u^{(n)} + f(u^{(n)}) + g(u^{(n)}(t-r)) \text{ on } D_2 \\
u^{(n+1)} &= u^{(n+1/2)} \text{ on } S_2 \\
u^{(n+1)} &= \eta_0(t,x) \text{ on } D_{2,(-r)}
\end{aligned}$$

Here $u^{(n+1/2)}$ is defined as $u^{(n)}$ on $\overline{D} \backslash \overline{D_1}$ and $u^{(n+1)}$ is defined as $u^{(n+1/2)}$ on $\overline{D} \backslash \overline{D_2}$.

Then $u^{(n+i/2)} \to u$ on D_i, $i = 1,2$ where u is the solution of (1) in $\mathcal{A}$

The proof outlines of theorem 5 is similar to the proof given by Lui [6] in the case of parabolic equation without time delay.

The other type of Schwarz iterative method to consider is the additive type of Schwarz iterative method which posses high parallelism than the multiplicative type. The following theorem presents the general case for m-subdomains.

Theorem 6. *Let* $u^{(0)} = u_i^{(0)} = \underline{u}$ *on* $\overline{D}$, for $i = 1, \ldots, m$ *with* $\underline{u} = h$ *on* S. *Define the additive Schwarz sequence by* $(n \geq 1)$

$$(\partial_t - \Delta + \underline{c})u_i^{(n)} = f(u_i^{(n-1)}) + \underline{c}u_i^{(n-1)} + g(u_i^{(n-1)}) \quad \text{on} \quad D_i$$

subject to

$$u_i^{(n)} = u^{(n-1)} \text{ on } S_i, \quad u_i^{(n)} = \eta_0(t,x) \text{ on } D_{i,(-r)}, \ i = 1, \ldots, m.$$

Here, $u_i^{(n)}$ *is defined as* $u^{(n-1)}$ *on* $\overline{D} \setminus \overline{D}_i$, *and*

$$u^{(n)}(t,x) = \max_{1 \leq i \leq m} u_i^{(n)}(t,x), \quad (t,x) \in \overline{D}.$$

Then $u_i^{(n)} \to u$ *on* D_i, $i = 1, \ldots, m$, *where* u *is the solution of (1) in* $\mathcal{A}$.

5 Conclusion

In this work we studied the applicability of the monotone iterative method and for the parabolic equation with reaction function involves a time delay function and we proved the convergence of the generated sequence of approximations to the exact solution.

References

1. W. Feng and X. Lu. Harmless delays for permanence in a class of population models with diffusion effects, *J.Math. Anal. Appl.*, 206:547-566, 1997.
2. X. Lu. Monotone method and convergence acceleartion for finite difference solutions of pa problems with time delays. *Numer. Methods for Part. Diff. Eqs.*11:591-602, 1995.
3. X. Lu. Combined methods for numerical solutions of parabolic problemwith time delays, *Appl. Math. Comput.* 89:213:254, 1994.
4. S. H. Lui. On monotone and Schwarz alternating method for non linear elliptic pdes. *M2AN*35:1-15, 2001.
5. S. H. Lui. On linear monotone and Schwarz method for nonlinear elliptic pdes. *Numer Math.* to appear.
6. S. H. Lui. On monotone iteration and Schwarz methods for nonlinear parabolic pdes. *Numer Math.* to appear.
7. C. V. Pao. *Nonlinear Parabolic and Elliptic Equations*, Plenum Press, New york, 1992.
8. C. V. Pao. Monotone iterations for numerical solutions of reaction-diffusion-convection equations with time delay. *Numer. Methods for part. Diff Eqs.* 14:339-351, 1998.

Author Index

Abad, Alberto I-858
Abawajy, Jemal H. IV-223, IV-242
Abe, Steffen III-883
Abraham, Ajith III-169
Acosta-Elias, Jesús II-275
Aguilar-Ruiz, Jesús S. II-766
Ahn, JinHo III-504
Akerkar, S. IV-319
Al-Dabbagh, Majid III-90
Aladjev, Victor I-711
Alahakoon, Damminda III-151
Albada, G. Dick van I-284, I-679
Albano, Giovannina I-754
Alber, M. III-39
Albertì, Margarita II-324
Albrecht, Andreas II-575
Alda, Witold I-669
Aldworth, Zane IV-337
Alique, A. I-351, II-812
Alique, J.R. I-351, II-812
Allemand, J.N.C. II-963
Almeida, Nuno II-267
Altas, Irfan I-691, III-422
Álvárez, Luis M. I-581, I-659, II-1079
Alves, Albano II-1022
Amodio, Pierluigi II-139
Amorim, Antonio II-267
Anai, Hirokazu I-828
Ananikyan, Nerses I-85
Anderson, Matthew III-422
Angelis, Filippo De II-331
Anido, Luis I-581, I-659, II-1079
Antonuccio-Delogu, Vincenzo II-1032
Appelbe, Bill III-781
Aprà, Edoardo IV-168
Argentini, R. I-987
Arickx, Frans I-474
Artoli, Abdel Monim M. I-997, I-1034
Ashrafi, Mafruz Zaman IV-978
Askarian Abyaneh, Hossein III-90
Athanassoulis, Gerassimos A. II-223
Aydin, Galip IV-393

Babuška, Ivo IV-385
Bacquet, P. III-514
Bagchi, Susmit II-1012
Bagirov, A.M. III-20, III-592
Bajuelos, António Leslie I-869
Bakker, A.F. I-987
Bała, Piotr II-307
Balas, Lale II-195, II-205
Baldridge, Kim K. IV-111
Baliś, Bartosz I-214
Balk, Igor III-1074
Balsoy, Ozgur IV-393
Bandyopadhyay, Subhajyoti IV-365
Bang, Sung yoon IV-911
Bang, Young-Cheol II-678, III-495, IV-518
Banks, Jasmine III-11
Barczak, Andre L.C. IV-1050
Barnes, Mark IV-930
Barron, John M. IV-365
Barseghyan, A.R. II-545
Batista, Rodolfo B. II-981
Batiuk, Richard III-80
Batten, Lynn IV-1040
Becciani, Ugo II-1032
Beco, Stefano II-1052
Bedford, Anthony III-179
Bedingfield, Susan E. IV-868
Beek, J.K.L. van I-649
Beh, Jounghoon IV-1109
Bekker, Henk III-70
Belanzoni, Paola II-376
Beletskyy, Volodymyr II-925
Beliakov, Gleb III-592, IV-1040
Bellucci, D. II-341
Beloussov, L.V. IV-698
Benedyczak, Krzysztof II-307
Benkner, S. III-612
Benninger, Kathy IV-33
Beran, Gregory J.O. IV-96
Berg, Jur P. van den III-70
Bernsdorf, J. I-1054
Bespamyatnikh, Sergei I-533
Beylkin, Gregory IV-103
Beynon, Michael IV-355
Bharathi, B. Suman III-1005
Billeter, Martin I-15

Birthelmer, Horst I-925
Bischof, Christian H. II-888
Bistarelli, Stefano I-811
Black, Gary IV-122
Blais, J.A.R. II-610
Blom, Joke G. IV-688
Blum, Carol IV-988
Boderie, P.M.A. I-649
Boek, Edo S. II-668
Boffi, Giandomenico I-811
Bogdanski, Maciej I-255
Bogg, Paul IV-859
Bonifacio, Anna II-748
Borgosz, Jan I-591, II-713, II-731
Boronat, Pablo II-935
Botana, Francisco I-801, I-879
Boucher, Charles I-125
Boukhanovsky, Alexander V. I-125, II-213, II-223, II-234
Bourchtein, Andrei I-25
Bournez, Olivier III-367
Brandstätter, Wilhelm I-1062
Branson, Kim III-739
Briant, A.J. I-1024
Broeckhove, Jan I-474
Brown, Ross III-995
Browne, James C. IV-385
Brun, O. III-514
Brunst, Holger II-285
Bryant, Steven IV-355
Bubak, Marian I-207, I-214, I-284
Bücker, H. Martin II-888
Bukatov, Alexander Alexeyevich II-945
Bulatov, Michael V. II-516
Buono, Nicoletta Del II-111
Burbano, Daniel III-602
Burns, John I-75
Burq, Shoaib III-739
Burrage, Kevin III-693
Buyya, Rajkumar III-739, III-1123
Bylaska, Eric J. IV-168
Byun, Yanga III-190

Čada, M. III-788
Caeiro, Manuel I-581
Cai, Lizhi I-225
Cai, Yang III-1005, III-1045, III-1064
Cai, Yuanlong I-457
Calle, Marcos IV-968
Campbell, P. III-979
Carmichael, Gregory R. IV-269
Carretero, Jesús IV-248
Carrington, Laura III-926
Carvalho, Luís Alfredo Vidal de I-511
Catalyurek, Umit IV-355
Catlin, Ann C. IV-739
Cavagnat, Dominique II-350
Cavalieri, Salvatore III-1083
Cebrat, Stanisław I-343, II-650
Cecchi, Guillermo A. IV-1127
Cepulkauskas, Algimantas ˇ I-747
Cerco, Carl III-80
Chae, Kijoon IV-535
Chae, Song-hwa IV-1060
Chai, Tianfeng IV-269
Chang, Robert IV-375
Chang, Shun-Chyi IV-1079
Chang, Yun Seok I-378, II-695, II-955
Chantrapornchai, Chantana IV-940
Chapman, Barbara IV-667
Chaturvedi, Alok R. IV-365
Chaturvedi, R. III-39
Chee, Clinton IV-637
Chen, Huajun III-1143
Chen, Li III-773
Chen, Max II-90
Chen, Ning II-34
Chen, Xiao-jian II-84
Chen, Zizhong III-665
Cheung, Yen IV-839
Chew, Paul IV-299
Chi, Sung Do IV-809
Chino, Masamichi III-120
Chiu, Steve IV-230
Cho, Gihwan I-194
Cho, Ki Hyung IV-524
Cho, Sok-Pal IV-545
Cho, Sun Young II-695, II-955
Cho, Tae Ho IV-809
Cho, Youngsong I-887
Choi, DaeKyu IV-443
Choi, Inchul IV-213
Choi, Jonghyun IV-473
Choi, Joong Sup III-474
Choi, Mun-Young II-685, IV-920
Choi, Sung Chune III-474
Choi, Sung Jin IV-958
Choi, Won-Hyuck III-453
Choi, YoungIl III-504
Choi, YoungSik IV-491

Cholvi, Vicente II-935, IV-453
Chong, Kihun II-703
Choo, Hyunseung I-466, II-879, III-463, III-495, IV-443, IV-518
Chopard, Bastien I-947, I-1044
Choudhary, Alok IV-230
Chrisochoides, Nikos IV-299
Chun, Seok-Ju IV-1030
Chung, Chin Hyun I-295, I-305, I-314
Chung, Ilyong I-147
Chung, Jin-wook IV-576
Chung, Kwang Sik III-286
Chung, Kyungchun II-695
Chung, Tai M. IV-463
Churilov, L. III-20
Cickovski, T. III-39
Cole, Martin J. IV-279
Collins, Michael A. IV-159
Côme, Guy-Marie III-367
Conraud, Valérie III-367
Cotofrei, Paul I-323
Cox, Simon J. III-297, IV-420
Crisci, Gino M. I-333
Cyganek, Bogusław I-591, II-713, II-721, II-731

D'Auria, Bernardo I-754
D'Orazio, Annunziata I-977
Daescu, Dacian N. IV-269
Daivis, Peter J. III-275
Daly, John IV-3
Daniilidis, Kostas IV-33
Dantas, M.A.R. II-963
Daoud, Daoud S. IV-1137
Datta, Amitava I-915, III-100, III-1054
Debbabi, M. III-750
Demkowicz, Leszek F. IV-385
Demmel, James W. III-705
Deng, Dafu II-822
Deng, Qian-ni IV-413
Deng, Xiaotie II-3, II-34
Denisikhin, Sergey II-565
Der, Uwe I-245
DeSouza, Jayant III-760
Dhaene, Tom II-741
Dhooge, Annick I-701
Díez, Fernando I-791
Dimitrov, Alexander G. III-612, IV-337
Dimitrov, Rossen III-307
Ding, Jingbo I-225, II-971
Diniz, Pedro C. III-916
Domínguez, Juan J. IV-968
Donchyts, Gennadiy I-35
Dong, Fangpeng III-1133
Dongarra, Jack J. III-233, III-665, IV-53, IV-759
Donnellan, Andrea III-827, III-855
Douglas, Craig C. IV-279
Dudek, Mirosław R. I-343, II-650
Dudkiewicz, Małgorzata I-343, II-650
Dufour, Frédéric III-837
Duke, Brian J. IV-132
Dunietz, Barry IV-96
Dupuis, A. I-1024
Dupuis, Michel IV-168
Dutka, Łukasz I-629
Dyskin, Arcady V. III-845, III-873
Dziurzanski, P. II-758

Edwards, Ron III-169
Eerola, P. I-264
Efendiev, Yalchin IV-279
Eijkhout, Victor IV-759
Ekelöf, T. I-264
El-Sayed, Mazen III-109
Ellert, M. I-264
Eltekov, Vitaly I-764
Emerich dos Santos, Luís Orlando I-967, I-1007
Emerson, Andrew II-316
Engelbrecht, G. III-612
Ercan, M. Fikret III-548
Eres, Hakki IV-420
Ergenç, Tanıl I-691
Erhioui, M. III-750
Escobar, José María I-568
Eshleman, Keith N. III-30
Evangelinos, Constantinos IV-375
Evangelisti, Stefano II-316
Ewing, Richard IV-279
Exposto, José II-1022

Facin, Paulo Cesar I-1007
Falaleev, Michael V. II-523
Fang, Yong II-10
Fann, George I. IV-103
Faray Melo, Renata Cristina II-981
Farhat, Charbel IV-309
Fatyanova, Olga II-446
Fei, Han II-267

Fellows, Geoff III-422
Fernández, Manuel J. I-659
Ferrer-Troyano, Francisco J. II-766
Ferro, Daniela II-1032
Filipová, Blanka II-585
Flitman, Andrew IV-930
Forgacs, G. III-39
Fox, Geoffrey III-827, III-863, IV-393, IV-403
Francois, D. I-417
Freeman, Justin III-781
Frolova, Julia II-629
Frutos Escrig, David de II-774
Fuchs, Henry IV-33
Fung, Yu-Fai III-548
Funika, Włodzimierz I-214
Fursov, Vladimir II-463
Furuno, Akiko III-120

Gabriel, Edgar I-235, II-285
Galant, Adam I-45
Galis, Alex III-1103
Gálvez, Akemi I-848, I-897
Gao, Dayong II-784
Gao, Wen III-537, IV-829
Gao, Xiaofeng III-926, III-966
García, Félix IV-248
Garcia, J.M. III-514
Garcke, Jochen III-683
Gashkov, Igor I-737, II-1042
Gava, Frédéric I-155
Gavrilova, M.L. IV-718
Gedeon, Tomas IV-337
Geelkerken, Bob III-1015
Gelas, Jean-Patrick III-1103
Germaná, Angela II-1032
Gervasi, Osvaldo II-297, II-324
Gevorkyan, A.S. II-545
Gheller, Claudio II-1032
Ghulghazaryan, Ruben I-85
Ghulyan, A.V. II-545
Gils, J.A.G. van I-649
Gimenez, Judit IV-181
Giorgi, Giacomo II-331
Glazier, J.A. III-39
Goguen, Joseph IV-1020
Gong, Yili III-1133
González-Yuste, José María I-568
Gopalsamy, S. IV-299
Gorbunov, Vladimir K. II-483, II-492
Gorbunova, Ekaterina O. II-794
Gordon, Mark S. IV-75
Gorokhov, Oleg II-436
Gottschling, Peter II-575
Govaerts, Willy I-701
Grabovsky, V.I. IV-698
Gracio, Debbie IV-122
Gradalski, Przemyslaw I-361
Granat, Robert A. III-892
Gray, Mike IV-355
Greenberg, Jerry P. IV-111
Gregorio, Salvatore Di I-333
Grimwood, Daniel J. IV-142
Gu, Yonggen II-3
Gülçat, Ü. III-622
Guerrero, F. IV-886
Guo, Jingzhi II-802
Gyulassy, Attila III-705

Ha, Kyeoung-Ju II-841
Haber, Rodolfo E. I-351, II-812
Halavati, Ramin I-601
Han, Dong-Soo III-213
Han, Kyungsook III-130, III-140, III-190
Han, Namshik III-140
Han, Sukwoo IV-626
Han, SunYoung II-917, III-485
Han, Xiaojun III-537
Han, Zongfen II-822
Hanan, Jim IV-708
Hang, Cheng Wei IV-839
Hansen, Christian IV-355
Hansen, J.R. I-264
Harrison, Robert J. IV-103
Hastings, Shannon IV-355
Hayashi, Yukio II-257
Hayryan, Edik I-54
Hayryan, Shura I-54
Head-Gordon, Martin IV-96
Heber, Gerd IV-299
Hegland, Markus III-683
Hentschel, G. III-39
Herrera, José II-1069
Higuchi, Kenji III-120
Hirabayashi, Miki I-1044
Hirao, Kimihiko IV-84
Hirata, So IV-84, IV-168
Hoefsloot, H.C.J. I-1034
Hoekstra, Alfons G. I-639, I-997, I-1034

Hoff, Jos van 't I-997
Hoheisel, Andreas I-245
Hoisie, Adolfy III-905
Hong, Choong Seon IV-789
Hong, Daniel Won-Kyu IV-789
Hong, Sung-Soo IV-586
Hong, Yongmiao II-90
Houstis, Elias N. IV-309, IV-739
Hu, Chin-Kun I-54
Hu, Qiying II-28, II-64
Huang, C. III-39
Huang, Chang III-1143
Huang, Lican III-316
Huang, Yan III-254, IV-430
Huber, Valentina II-307
Huh, Eui-Nam IV-482
Huntoon, Gwendolyn IV-33
Hwang, Il-Sun IV-576
Hwang, Jun I-295, I-305, I-314, III-401
Hwang, Suntae IV-1117
Hyun, Hojae III-485

Ibănescu, Liliana III-367
Igel, Heiner III-807
Iglesias, Andrés I-848, I-897, I-935
Iizuka, Mikio III-773
Imasaki, Kenji IV-242
Inamuro, Takaji I-1015
Ince, Huseyin IV-289
Ingraffea, Tony IV-299
Ionescu, Dorin-Cezar III-807
Islas, A.L. II-101
Ito, K. II-784
Ivanov, Sergey V. II-234
Izaguirre, J.A. III-39, III-50
Izworski, Antoni I-427
Izydorczyk, Adam I-361

Jaekel, U. I-1054
Jammalamadaka, R. IV-319
Janicki, Aleksander I-361
Jayatilaka, Dylan IV-142
Jee, Jeung-gyu IV-545
Jeffrey, Stephen IV-679
Jensen, Iwan III-203
Jeong, Chang-Sung I-610
Jeong, Chang-Won IV-877
Jeong, Karpjoo IV-1117
Jesús Cruz Guzmán, José de II-832
Ji, Yongchang I-437
Jiao, Zhouan IV-420
Jiménez, J.E. II-812
Jin, Hai II-822
Jin, Haoqiang IV-181
Jin, Min-Jung IV-535
Jin, Sung Ho III-286
Jin, Ying IV-393
Jo, Sung-Up I-610
Johnson, Chris R. IV-279
Johnson, Gregory S. III-946
Johnson, Martin J. IV-1050
Johnsson, S. Lennart IV-729
Jonassen, Tore M. I-85
Jones, Greg IV-279
Joo, Kyung-Soo II-685, IV-911, IV-920
Joo, Su-Chong III-401, IV-877
Joppich, Wolfgang II-161
Jorba, Josep IV-191
Jost, Gabriele IV-181
Jung, Jin-Won III-652
Jung, Seunho IV-1117
Jung, Soon Young III-286

Kaandorp, Jaap A. I-639, IV-688
Kalé, Laxmikant V. III-760, IV-23
Kamiński, Marcin I-521
Kamikawa, Naoki I-492
Kamiya, Muneaki IV-84
Kanada, Yasumasa III-581
Kandhai, D. I-1034
Kantardgi, Igor II-188
Karasözen, Bülent II-533
Karlik, Bekir I-371
Karniadakis, George Em IV-346
Kashevarova, Tamara II-1088
Kasprzak, Andrzej IV-799
Kawahara, Kenji II-247
Kawashima, Yukio IV-84
Kazemi Karegar, Hossein III-90
Keane, Andy IV-420
Keller, Rainer I-235
Kelshikar, Nikhil IV-33
Kerbyson, Darren J. III-905, II-936
Ketari, L. III-750
Keulen, Wilco I-125
Kiat, Deniss Teng Tai IV-930
Kim, Bo Yeon I-378, II-695, II-955
Kim, Deok-Soo I-887
Kim, Dong S. IV-463
Kim, Dong-kyoo IV-1060

Kim, Donguk I-887
Kim, Dongwook IV-1117
Kim, Eui Sun IV-994
Kim, Hae-Jin III-213
Kim, Hong-Soog III-213
Kim, Hongjin IV-555
Kim, Hyun I-887
Kim, Hyun-Ki IV-877
Kim, Hyung-Jun I-610
Kim, HyunSook III-445
Kim, Hyuntaek II-695
Kim, Hyunwoo III-140
Kim, Jai-Hoon III-652
Kim, Jeom goo IV-596
Kim, Jin Ok I-295, I-305, I-314, I-619
Kim, Jong-Seok III-527
Kim, Jongoh IV-473
Kim, Jongwoo II-695
Kim, Jung-Sun III-453
Kim, Kee-cheon II-917
Kim, Keechon III-485
Kim, Kuinam J. IV-555, IV-626
Kim, Kwang H. IV-463
Kim, Min Jun III-652
Kim, Moon Hae IV-1117
Kim, Myung Il IV-500
Kim, Nam Kyun II-636, II-644
Kim, Sang-Kil IV-586
Kim, Seongyeol I-147
Kim, Shin-Dug IV-819
Kim, Sung Jo IV-500
Kim, Tae-Kyung IV-463
Kim, Tai-hoon IV-605
Kim, Wonil IV-1060
Kim, Yoon Hyuk IV-771
Kim, Young-Sung IV-626
Kinouchi, Y. II-784
Kirchner, Hélène III-367
Kiryukhin, Ilia I-125
Kitowski, Jacek I-629
Kivva, Sergey II-619
Klein, Clementine III-1005
Klein, William III-827
Klein-Seetharaman, Judith III-1005
Kłopotek, Mieczysław A. I-397
Knight, Doyle IV-329
Knüpfer, Andreas III-956
Ko, A. III-50
Ko, Hanseok IV-1089, IV-1100, IV-1109
Kobayashi, Takuya III-120
Kochmar, John IV-13
Koivisto, Hannu II-907
Kommers, Piet III-1015
Kondratenko, Yuri V. II-794
Kondratiev, Alexey II-446
Konopleva, Irina II-533
Konstantinov, A. I-264
Kónya, B. I-264
Kopylov, A. II-507
Korczyński, Waldemar II-832
Korobitsin, Victor II-629
Koshigoe, Hideyuki III-388
Kosiedowski, Michal I-255, II-1052
Kou, Gang II-54
Kouh, Hoon-Joon IV-849
Kovalchick, L. III-377
Kowalczuk, Maria I-343, II-650
Kozera, Ryszard III-100
Kozloski, James IV-1127
Kramer, William T.C. III-560
Kranzlmüller, Dieter IV-201
Krasnikov, S.D. II-500
Kremenetsky, Mark IV-657
Krishnamurthy, E.V. III-333, IV-779
Krogstad, Harald E. II-213, II-223
Krul, Tommy IV-688
Ku, Kyo-Min II-841
Kudo, Makoto III-581
Kulikov, Gennady Y. I-540
Kulikova, Maria V. II-417, II-427
Kulvietienė, Regina I-747
Kulvietis, Genadijus I-747
Kumar, Sameer IV-23
Kunst, Eelco III-1015
Kuonen, Pierre I-165, I-175
Kupczyk, Miroslaw II-1052
Kurc, Tahsin IV-355
Kuroda, Hisayasu III-581
Kutner, Ryszard I-45, I-407
Kuznetsov, E.B. II-500, II-507
Kuznetsov, Yuri A. I-701
Kwok, Terence IV-886, IV-968
Kwok, Wai Yip IV-43
Kwon, Oh-Young IV-819
Kwon, Taehee IV-1100
Kwon, Yong-won I-610

Łabaj, Paweł I-669
Labarta, Jesus IV-181

Laganà, Antonio II-297, II-324, II-341, II-357
Lago, N. Faginas II-357
Lai, K.K. II-10
Lange, Marc II-285
Langella, Steve IV-355
Langer, Malgorzata II-636, II-644
Lari-Lavassani, Ali I-95
Larsen, Scott IV-33
Lau, H.C.W. III-432
Lavrenov, Igor V. II-179
Lazarov, Raytcho IV-279
Leal, Katia II-1069
Lechenault, Frederic IV-309
Lee, Bo Kyoung IV-958
Lee, Bum Ro I-295, I-305
Lee, Byung-Kwan IV-510
Lee, ByungSun III-504
Lee, Chee Wai IV-23
Lee, Dae-Won III-286
Lee, Dong Chun IV-605, IV-614
Lee, Dongkyu III-130
Lee, DongWoo III-233
Lee, Gi-Sung IV-545
Lee, Hwa Min III-286
Lee, Hwang-Jik IV-819
Lee, Hyeong-Ok III-527
Lee, Hyung-Woo IV-1069
Lee, Im-Yeong IV-1069
Lee, Jong-Hee IV-566
Lee, Jongchan IV-555
Lee, Ju-Hong IV-1030
Lee, Keun-Wang IV-566
Lee, Myungsun IV-576
Lee, Okbin I-147
Lee, Sangho I-147
Lee, Sangsan IV-1117
Lee, Sangyoun IV-491
Lee, Seok Jae IV-524
Lee, Seok-Lyong IV-1030
Lee, Sukyoung III-445
Lee, Tae-Seung III-453
Lee, W.B. III-495
Lee, Weontae III-652
Lee, Won Gyu III-286
Lee, Woongjae I-295, I-305, I-314
Lee, Youn-Joo IV-819
Lefèvre, Laurent III-1103
Lencastre, Maria II-555
Lendasse, A. I-417
Lermusiaux, Pierre F.J. IV-375
Lesoinne, Michel IV-309
Lespade, Laure II-350
Leszczynski, Jacek S. I-105
Lewoc, Jozef B. I-427
Li, Ben II-20
Li, Ru-zhong II-84
Li, Wei III-1133
Li, Xiaobo I-447
Li, Xun I-95
Liao, Wei-keng IV-230
Lichwala, Rafal II-1052
Liechti, Kenneth M. IV-385
Lim, Jaehwan IV-596
Lim, JongSeon IV-911, IV-920
Lim, Kieran F. IV-1040
Lim, Suk Hyun III-60
Lim, Youngho IV-1117
Lin, Hong I-437
Lin, Nancy P. IV-950
Lin, Whe Dar II-1059
Lindner, Peggy I-235
Linker, Lewis C. III-30, III-80
Lisik, Zbigniew II-636, II-644
Liu, Jianyong II-64
Llamas, Martín I-581, II-1079
Loginov, Boris II-533
Lomazova, Irina A. II-851
Lopatoukhin, Leonid J. II-213, II-223
Lopez, Luciano II-111
Loulergue, Frédéric I-155, III-223
Lowe, C.P. I-987
Lowe, Nick I-915, III-1054
Lozano, S. IV-968
Lu, Guibin II-28
Lu, Xin-da IV-413
Lucor, Didier IV-346
Luengo, F. I-935
Luke, Edward IV-299
Luo, Jianqiang III-264
Luque, Emilio IV-191
Luszczek, Piotr III-665
Lutoshkin, Igor V. II-483
Luursema, Jan-Maarten III-1015
Lynch, Robert E. I-437
Lyzenga, Gregory III-855

Macal, C. III-979
Mackiewicz, Dorota I-343, II-650
Mackiewicz, Paweł I-343, II-650

Madina, Duraid IV-896
Madsen, Henrik I-482
Magalhaes Alves de Melo, Alba Cristina II-981
Magoni, Damien III-1093
Majerowski, Andrzej I-45
Makinouchi, A. III-798
Malawski, Maciej I-207
Malony, Allen D. IV-53, IV-749
Manuali, Carlo II-297
Margalef, Tomàs IV-191
Mărginean, Flaviu Adrian III-644
Marinescu, Dan C. I-437
Marques, Fábio I-869
Marroquín Alonso, Olga II-774
Martino, Beniamino Di II-748
Martino, Ryan IV-355
Martoyan, Gagik A. II-593, II-600
Martynenko, Serguei I-550
Mascagni, Michael I-63
Mastroserio, C. II-149
Mathis, Mark M. III-905
Matsuhisa, Takashi II-74
Matsumoto, H. I-1054
Matsumoto, Nobuko III-120
Matthey, T. III-50
May, David III-781
Mayer-Patel, Ketan IV-33
Mazurek, Cezary I-255
Mazzia, Annamaria I-560
McCracken, Michael O. IV-749
Mehta, Shailendra IV-365
Melnik, Roderick V.N. III-343
Melnikoff, Steve III-739
Melotti, Maura II-1032
Memnonov, Vladimir II-565
Menasalvas, Ernestina II-1069
Merks, Roeland I-639
Messom, Chris H. IV-1050
Meyer, Norbert II-1052
Michisaka, Koshi I-492
Michopoulos, John IV-309
Mieszkowicz-Rolka, Alicja I-115
Mikic, Fernando A. I-581, I-659, II-1079
Miller, John P. IV-337
Min, SungGi III-504
Minasyan, Seyran H. II-593
Mirković, Dragan IV-729
Mohades, Ali II-861
Mohamed, Khaireel A. III-100
Mohr, Jonathan I-447
Monforte, Salvatore III-1083
Monteil, T. III-514
Montenegro, Rafael I-568
Montero, Gustavo I-568
Moore, Shirley IV-53
Mora, Peter III-798, III-817, III-883
Morajko, Anna IV-191
Morajko, Oleg IV-191
Moresi, Louis III-781, III-788, III-837
Moriyón, Roberto I-791
Mota, Sonia IV-1011
Mou, Xuanqin I-457
Mucci, Philip IV-53
Mühlhaus, Hans-B. III-788, III-837, III-873
Müller, Matthias S. I-235, II-285
Mulligan, Jane IV-33
Mumey, Brendan IV-337
Mun, Youngsong III-463, IV-473, IV-482, IV-518

Nadarajan, R. IV-1000
Nagai, Haruyasu III-120
Nagashima, Umpei III-244
Nagel, Wolfgang E. II-285
Nakajima, Kengo III-773
Nakajima, Takahito IV-84
Nakamura, Hisashi III-773
Nakano, Haruyuki IV-84
Nakao, Yoshihide IV-84
Nam, Young Jin IV-257
Nam, Young-Kwang IV-1020
Narayanan, Sivaramakrishnan IV-355
Naumann, Uwe II-575
Naumov, Lev II-869
Navarro-Moldes, Leandro II-275
Neto, Joaquim IV-299
Neves, Armando G.M. I-838
Neřiva, Pavel II-585
Newman, S.A. III-39
Ng, Patrick K.L. III-412
Ng, Vincent T.Y. III-412
Nguyen, Hung P. I-947
Nguyen, Tuan-Anh I-165, I-175
Nielsen, J.L. I-264
Nielsen, Ole III-683
Niewiadomska-Szynkiewicz, Ewa II-991
Nishikawa, Takeshi III-244

Nishizawa, K. I-1054
No, Byung-gyu IV-605
Noël, Alfred G. I-728
Noh, Sam H. II-703
North, M. III-979
Nowicka, Aleksandra I-343, II-650
Nowoczynski, Paul IV-13
Núñez, Manuel II-658
Nutaro, J. IV-319
Nygaard, Mads II-1012

O'Hare, Gregory III-1035
Oden, J. Tinsley IV-385
Özhan, Erdal II-195, II-205
Ofengeim, Dmitry K. I-3
Ogata, Takeshi I-1015
Ogino, Fumimaru I-1015
Oguz, C. III-548
Oh, Am-Suk IV-510
Oh, Eunseuk III-527
Oh, Hae-Seok IV-566
Oh, Sung-Kwun III-401, IV-877
Ohta, Makoto I-1044
Oi, Yuji II-247
Okuda, Hiroshi III-773
Olazábal, Juan M. de I-718
Orekhov, Vladislav I-15
Ortega, Julio IV-1011
Osborne, M.R. III-673
Otani, Takayuki III-120
Ould-Saada, F. I-264
Ožana, Štěpán II-585
Oziewicz, Zbigniew II-832

Pacholczyk, Daniel III-109
Paek, Han Wook I-314
Palak, Bartek II-1052
Palmer, Bruce IV-122
Pan, Yunhe III-316
Parejo, J.A. IV-886
Park, Chanik IV-213, IV-257
Park, Eunhea II-695
Park, Hyoungwoo I-610
Park, Hyung-Woo IV-819
Park, Junbum IV-1089
Park, Jung-Min IV-535
Park, Kwang Suk I-378
Park, Kyung-Lang IV-819
Park, Sung-Hoon I-185, I-387
Park, Sung-Yong IV-819
Parker, Albert E. IV-337
Parker, Jay III-855
Pascal, P. III-514
Passos, L.B.C. II-963
Pasternak, E. III-873
Paternoster, Beatrice II-131
Patrikalakis, Nicholas M. IV-375
Peck, Charles C. IV-1127
Peña, José M. II-1069, IV-248
Peng, Yi II-54
Pérez, María S. IV-248
Peszynka, Malgorzata IV-355
Petrischev, Vyacheslav V. II-492
Petrovic-Lazarevic, Sonja III-169
Pham, Binh III-995
Philippe, B. III-693
Philippi, Paulo Cesar I-967, I-1007
Pierce, Marlon III-863, IV-393, IV-403
Pina, António II-1022
Pingali, Keshav IV-299
Pini, Giorgio I-560
Pivovarov, J. IV-718
Plóciennik, Marcin II-1052
Pokorny, Imrikh I-54
Polak, Natalia I-343
Politi, T. II-121, II-149
Pollack, Lisa IV-168
Poloskov, Igor E. I-781
Pooley, C.M. I-1024
Postma, L. I-649
Pound, Graeme IV-420
Pound, Graeme E. III-297
Provins, D.A. II-610
Puchala, Edward III-3
Puig-Pey, J. I-897
Putti, Mario I-560
Pytliński, Jarosław II-307

Qi, X.T. III-548
Qian, Jia-zhong II-84
Quaas, Johannes II-161
Quintella Mendes, Daniele I-511

Ra, Yoochan IV-596
Racero, J. IV-886
Radhakrishnan, S. II-678
Raefsky, Arthur IV-657
Raj, Ewa II-636, II-644
Ramakrishna, R.S. III-233
Rao, A. Ravishankar IV-1127

Rao, N.S.V. III-495
Rao, Nageswara S.V. II-678
Rasch, Arno II-888
Razzazi, Mohammadreza II-861
Re, Nazzareno II-331
Reddy, Raghurama IV-33
Redl, Christian I-1062
Rees, Jackie IV-365
Reeve, J.S. III-350, III-570
Reilly, Ronan III-1035
Reinefeld, Alexander I-274
Reinhardt, Steve IV-657
Rendell, Alistair IV-648
Renton, Michael IV-708
Resch, Michael M. I-235, II-285
Reuter, Jürgen IV-63
Rice, John R. IV-309, IV-739
Richard, S. III-514
Richman, Michael B. IV-289
Riganelli, Antonio II-324
Rimnac, Axel IV-201
Riquelme, José C. II-766
Robertson, Nathan IV-648
Robles, Víctor IV-248
Roche, Kenneth III-665
Rodionov, Alexey S. II-879
Rodríguez, Eduardo I-568
Rodríguez, Ismael II-658, II-1002
Rodríguez, Judith I-659
Rödel, Steffan III-1015
Rolka, Leszek I-115
Romanova, Olga A. II-523
Rong, Zimei II-350
Rongo, Rocco I-333
Ros, Eduardo IV-1011
Rosi, Marzio II-366, II-376
Rossi, Elda II-316
Rossi, Fabio I-811
Rozhkov, Valentin A. II-213, II-223
Rubio, Fernando II-1002
Rüfenacht, Daniel A. I-1044
Rufino, José II-1022
Rundle, John B. III-827, III-855
Ruskin, Heather J. I-75
Ryan, Clint III-560
Ryu, So-Hyun I-610

Sadovsky, Michael G. II-794
Sahm, Jörg I-925
Salerno, Saverio I-754
Saltz, Joel IV-355
Samokhvalov, Konstantin II-471
San-Juan, Juan Félix I-858
Sandu, Adrian IV-269
Santos, Felix C.G. II-555
Santos, Juan M. II-1079
Sartoretto, Flavio I-560
Sas, Corina III-1035
Saskov, Kirill I-125
Sawant, Vivek IV-33
Saywell, M. III-570
Schintke, Florian I-274
Schmidt, Michael W. IV-75
Schmidt, R. III-612
Schober, C.M. II-101
Schuchardt, Karen IV-122
Schütt, Thorsten I-274
Scibilia, Fabio III-1083
Scott, J. Ray IV-13
Segarra, Juan IV-453
Seguel, Jaime III-602
Sekiguchi, Satoshi III-244
Sekino, Hideo IV-84
Semenov, Alexander II-1088
Semoushin, Innokenti II-407, II-417, II-436, II-1098
Seo, Hee Suk IV-809
Sessanna, Dennis IV-355
Sethumadhavan, Simha III-946
Sgamellotti, Antonio II-331, II-366, II-376
Shakhov, Vladimir V. I-466
Sheka, E.F. II-386
Shende, Sameer IV-53
Shenk, Gary III-80
Shi, Yong II-54
Shih, Alan IV-299
Shin, Byeong-Seok III-60
Shin, Chang-Sun III-401
Shin, Jae Ryong IV-524
Shindin, Sergey K. I-540
Shiri, Mohammad Ebrahim II-861
Shouraki, Saeed Bagheri I-601
Shu, Jiwu II-3
Sidje, Roger B. III-693
Sidorov, Nicholas A. II-523
Simmel, Derek IV-13
Simonov, Nikolai A. I-63
Simpson, D.P. III-350
Sinha, Sudipta IV-33

Skjellum, Anthony III-307
Skorwider, Łukasz II-307
Skowronski, Slawomir I-427
Slim, Chokri III-159
Sloot, Peter M.A. I-125, I-284, I-639, I-679, I-997, I-1034, III-1025
Slusanschi, Emil II-888
Sman, R.G.M. van der I-1072
Smeds, Nils IV-637
Smirnova, O. I-264
Smith, Dayle M. IV-168
Smith, Kate A. III-151, IV-868, IV-886, IV-968, IV-978
Smolarczyk, Kamila I-343
Snavely, Allan III-926, III-966, IV-749
Snel, Ingo III-1005
Snider, Ross IV-337
Sobecki, Janusz II-1107
Soetebier, Ingo I-925
Sommerfield, Jason IV-13
Song, Ha Yoon II-703
Song, JooSeok III-445
Song, Seok Il IV-524
Soni, Bharat IV-299
Sparks, Travis IV-33
Spataro, William I-333
Spooner, Daniel III-936
Sriram, Kotikalapudi III-445
Standish, Russell K. IV-637, IV-896
Stankova, Elena N. II-171
Stefano, Marco Di II-366
Stephanakos, Christos N. II-223
Stodghill, Paul IV-299
Stoffel, Kilian I-323
Stoll, Serge I-947
Stone, Nathan IV-13
Storchi, Loriano II-297
Straatsma, T.P. IV-168
Strazdins, Peter E. III-715
Stredney, Don IV-355
Stuer, Gunther I-474
Su, C.-H. IV-346
Subramani, K. III-377
Succi, Sauro I-957, I-977
Sulistio, Anthony III-1123
Sun, Chengzheng II-802
Sun, Jizhou I-907
Sun, Zhigang I-907
Sung, Yoon Bang II-685
Suresh, B. IV-1000
Suresh, S. IV-903
Surmas, Rodrigo I-967
Sussman, Alan IV-355
Suzuki, Yoshio III-120
Sviridov, Vyacheslav Y. II-492
Świtała, Filip I-407
Szepieniec, Tomasz I-214
Szmidt, Jan II-636, II-644

Tadić, Bosiljka I-136
Takatsuka, Masahiro III-985
Takei, T. I-1054
Takine, Tetsuya II-247
Tan, Guozhen III-537, IV-829
Taniar, David IV-978
Tarantelli, Francesco II-297
Tashlinskii, Alexandr II-456
Tasso, S. II-341
Tavadyan, Levon A. II-593, II-600
Tawbi, N. III-750
Telles Walter, Maria Emília II-981
Ten-no, Seiichiro IV-152
Terada, Hiroaki III-120
Terziev, N. III-612
Thambipillai, Srikanthan III-360
Thomas, G. III-39
Thompson, David IV-299
Thornby, David IV-708
Tiampo, Kristy III-827
Tian, Jinlan II-20
Tichon, Jennifer III-11
Tichy, Walter F. IV-63
Tirado-Ramos, Alfredo I-284, I-679
Tobon, J.E. Monsalve III-592
Todd, B.D. III-275
Tomás, Ana Paula I-869
Tong, Weiqin I-225, II-971
Topa, Paweł I-669
Toro, Francisco de IV-1011
Towles, Herman IV-33
Trabelsi, Abdelwahed III-159
Tracy, Fred III-725
Trafalis, Theodore B. IV-289
Tran, Minh III-1054
Treebushny, Dimitri I-482
Trezentos, Paulo II-267
Tsompanopoulou, Panagiota IV-309
Tsuneda, Takao IV-84
Tsyganova, Julia II-417, II-1098
Tullis, Terry III-855

Tyszka, Jarosław I-669

Ünal, V.Ü. III-622
Ufuktepe, Ünal I-774, I-821
Ugarov, Vladimir II-1098
Umeo, Hiroshi I-492
Urbanic, John IV-33

Valeev, Sultan II-471
Vavasis, Steve IV-299
Verleysen, M. I-417
Verykios, Vassilios S. IV-739
Vieira, Mardoqueu II-555
Vijayan, Prakash IV-903
Villate, Jaime E. II-267
Virtue, P. III-39
Vitillaro, Giuseppe II-297
Vizino, Chad IV-13
Voorhis, Troy Van IV-96
Vuduc, Richard III-705

Wäänänen, A. I-264
Walker, David W. III-254
Walkowiak, Krzysztof I-501
Wang, Guilian IV-1020
Wang, Jiangchun III-264
Wang, Jingcun IV-988
Wang, Jun II-46
Wang, KiCheoul I-194
Wang, Ping III-30, III-80
Wang, Shou-Yang II-10
Wang, Xiaozhe III-151
Wason, Jasmin IV-420
Wassenaar, Tsjerk A. III-70
Wawrzynek, Paul IV-299
Weatherley, Dion III-817
Webb, Darren III-1113
Wedemann, Roseli S. I-511
Wendelborn, Andrew L. III-1113
Weng, Tien-hsiung IV-667
Wertz, V. I-417
Whang, Weiwan II-695
Wheeler, Mary IV-355
Windus, Theresa L. IV-168
Wismüller, Roland I-214
Woch, Marcin I-397
Wolniewicz, Malgorzata I-255
Wolniewicz, Pawel II-1052
Wolter, Nicole III-926
Wong, Christina W.Y. III-432
Woo, Yong-je I-610
Woźniak, Adam II-897
Wozniak, Michal III-635
Wroński, Michał II-307
Wu, Jigang III-360
Wu, Kai-ya II-84
Wu, Zhaohui III-316, III-1143
Wu, ZhiMei II-46

Xia, Xiang-fu IV-829
Xing, H.L. III-798
Xiong, Guangyu II-907
Xiu, Dongbin IV-346
Xu, Fenglian IV-420
Xu, Jiefeng III-1143
Xu, Weixuan II-54
Xu, Zhiwei III-1133
Xue, Gang III-297

Yahyaoui, H. III-750
Yakovlev, Dmitry II-668
Yamagishi, Nobuhiro III-120
Yamamoto, Hiroshi II-247
Yanai, Takeshi IV-84, IV-103
Yanami, Hitoshi I-828
Yang, Chao-Tung IV-1079
Yang, Kun III-1103
Yantır, Ahmet I-821
Yates, Brian F. IV-132
Yazıcı, Ali I-691
Yelick, Katherine A. III-705
Yellowlees, Peter III-11
Yeo, Chee Shin III-1123
Yeomans, J.M. I-1024
Yoo, Hyeong Seon IV-994
Yoo, Jae Soo IV-524
Yoo, Jae-Pil II-917
Yoo, Kee-Young II-841
Yoo, Ki-sung IV-576
Yoo, Weon-Hee IV-849
Yoon, Seok Ung IV-994
Youn, Choonhan III-863, IV-403
Youn, Hee Yong III-474, IV-958
Yu, Hengyong I-457
Yu, Heon Chang III-286
Yue, Wuyi II-64
Yun, Dong-Sik IV-789

Zabulis, Xenophon IV-33
Zając, Katarzyna I-207, I-284, I-679

Zatevakhin, Mikhail A. II-171
Zeephongsekul, Panlop III-179
Zeigler, B.P. IV-319
Zeiser, T. I-1054
Zhang, HuanQiang II-46
Zhang, Jiawan I-907
Zhang, Shensheng III-264
Zhang, Suqin II-20
Zhao, Zhiming I-284, I-679
Zheleznyak, Mark I-35
Zheng, Gengbin IV-23
Zheng, Weimin II-3
Zhioua, S. III-750
Zhmakin, Alexander I. I-3
Zhou, Zhongwu III-275
Zhu, Hong II-34
Zhu, Lin II-20
Zhuravleva, Svetlana II-565
Żmuda, Maciej II-991
Zotsenko, Kate N. III-343
Zudilova, E.V. III-1025