

Lecture Notes in Bioinformatics

3370

Edited by S. Istrail, P. Pevzner, and M. Waterman

Editorial Board: A. Apostolico S. Brunak M. Gelfand
T. Lengauer S. Miyano G. Myers M.-F. Sagot D. Sankoff
R. Shamir T. Speed M. Vingron W. Wong

Subseries of Lecture Notes in Computer Science

Akihiko Konagaya Kenji Satou (Eds.)

Grid Computing in Life Science

First International Workshop on Life Science Grid, LSGRID 2004
Kanazawa, Japan, May 31 – June 1, 2004
Revised Selected and Invited Papers



Springer

Series Editors

Sorin Istrail, Celera Genomics, Applied Biosystems, Rockville, MD, USA
Pavel Pevzner, University of California, San Diego, CA, USA
Michael Waterman, University of Southern California, Los Angeles, CA, USA

Volume Editors

Akihiko Konagaya
RIKEN Genomic Sciences Center
Bioinformatics Group
E216 1-7-22 Suehiro-cho, Tsurumi, Yokohama, 230-0045, Japan
E-mail: konagaya@gsc.riken.jp

Kenji Satou
School of Knowledge Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-1292, Japan
E-mail: ken@jaist.ac.jp

Library of Congress Control Number: 2005921642

CR Subject Classification (1998): H.4, D.4, D.2, F.2, J.3

ISSN 0302-9743

ISBN 3-540-25208-8 Springer 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. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11403326 06/3142 5 4 3 2 1 0

Preface

Researchers in the field of life sciences rely increasingly on information technology to extract and manage relevant knowledge. The complex computational and data management needs of life science research make Grid technologies an attractive support solution. However, many important issues must be addressed before the Life Science Grid becomes commonplace.

The 1st International Life Science Grid Workshop (LSGRID 2004) was held in Kanazawa Japan, May 31–June 1, 2004. This workshop focused on life science applications of grid systems especially for bionetwork research and systems biology which require heterogeneous data integration from genome to phenome, mathematical modeling and simulation from molecular to population levels, and high-performance computing including parallel processing, special hardware and grid computing.

Fruitful discussions took place through 18 oral presentations, including a keynote address and five invited talks, and 16 poster and demonstration presentations in the fields of grid infrastructure for life sciences, systems biology, massive data processing, databases and data grids, grid portals and pipelines for functional annotation, parallel and distributed applications, and life science grid projects. The workshop emphasized the practical aspects of grid technologies in terms of improving grid-enabled data/information/knowledge sharing, high-performance computing, and collaborative projects. There was agreement among the participants that the advancement of grid technologies for life science research requires further concerted actions and promotion of grid applications. We therefore concluded the workshop with the announcement of LSGRID 2005. More information about the workshop is available at: <http://www.lsgrid.org/>

This post proceedings contains the revised versions of the accepted papers of the LSGRID 2004 workshop. Ten regular papers were selected for inclusion in the postproceedings. The papers address the following issues:

- An Integrated System for Distributed Bioinformatics Environment on Grids
- Distributed Cell Biology Simulations with E-Cell System
- The Architectural Design of High-Throughput BLAST Services on OBIGrid
- Heterogeneous Database Federation Using Grid Technology for Drug Discovery Process
- Grid Portal Interface for Interactive Use and Monitoring of High-Throughput Proteome Annotation
- Grid Workflow Software for a High-Throughput Proteome Annotation Pipeline
- Genome-Wide Functional Annotation Environment for *Thermus thermophilus*
- Parallel Artificial Intelligence Hybrid Framework for Protein Classification

- Parallelization of Phylogenetic Tree Inference Using Grid Technologies
- Building a Biodiversity GRID

In addition to the regular papers, the postproceedings includes an invited keynote address by Hideaki Sugawara on:

- Gene Trek in Procaryote Space Powered by a Grid Enviornment

and the following four papers presented in invited talks and posters in LSGRID 2004 (these papers were reviewed by the editors of the postproceedings)

- EMASGRID: an NBBnet Grid Initiative for a Bioinformatics and Computational Biology Services Infrastructure in Malaysia
- Development of a Grid Infrastructure for Functional Genomics
- Mega Process Genetic Algorithm Using Grid MP
- “Gridifying” an Evolutionary Algorithm for Inference of Genetic Networks Using the Improved GOGA Framework and Its Performance Evaluation on OBI Grid

We would like to acknowledge all the Program Committee members and all the additional referees for their work on reviewing the submitted papers. We also wish to thank all the authors and participants of the workshop for contributing to lively dicussions and the exchange of knowledge and experiences on the Life Science Grid. It should also be mentioned that the workshop was independent but closely related to the Life Science Grid Research Group (LSG-RG) of the Global Grid Forum. More than half of the Program Committee members are also active members of the Life Science Grid Research Group. It would be difficult to organize such an international workshop without the continuous efforts of the LSG-RG. Finally, we wish to thank Fumikazu Konishi, Sonoko Endo, Maki Otani, Kyoko Hirukawa, Yuko Watada, Aki Hasegawa and Shigerv Takasaki for their help in organizing this workshop and editing the proceedings.

October 2004

Akihiko Konagaya
Kenji Satou

Organization

LSGRID 2004 was organized by the Special Interest Group on Molecular Bioinformatics (SIGMBI) of the Japanese Society for Artificial Intelligence, the Open Bioinformatics Grid (OBIGrid) Project, and the Japan BioGrid Project.

Executive Committee

Program Chair:	Akihiko Konagaya (RIKEN GSC, Japan)
Organizing Chair:	Kenji Satou (JAIST, Japan)
Exhibition:	Rei Akimoto (Sun Microsystems, Inc., Japan)

Program Committee

Akiyama, Yutaka	(AIST CBRC, Japan)
Ang, Larry	(BII, Singapore)
Angulo, David	(DePaul Univ., USA)
Arzberger, Peter	(UCSD, USA)
Bala, Piotr N.	(Copernicus Univ., Poland)
Goble, Carole	(Univ. of Manchester, UK)
Farazdel, Abbas	(IBM, USA)
Fukuda, Ken'ichiro	(AIST CBRC, Japan)
Himeno, Ryutaro	(RIKEN ACCC, Japan)
Hong, Gilnam	(POSTECH, South Korea)
Kao, Cheng-Yao	(NTU, Taiwan)
Konagaya, Akihiko	(RIKEN GSC, Japan)
Konishi, Fumikazu	(RIKEN GSC, Japan)
Lin, Fang-Pang	(NCHPC, Taiwan)
Luo, Jingchu	(CBI, Peking University, China)
Miyazaki, Satoru	(NIG, Japan)
Nakamura, Haruki	(Osaka Univ., Japan)
Matsuda, Hideo	(Osaka Univ., Japan)
Matsuoka, Satoshi	(TITECH, Japan)
Mohamed, Rahmah	(UKM, Malaysia)
Napis, Suhaimi	(UPM, Malaysia)
Ono, Isao	(Tokushima Univ., Japan)
Palittapongarnpim, Prasit	(BIOTEC, Thailand)
Rodrigo, Allen	(Auckland, New Zealand)
Satou, Kenji	(JAIST, Japan)
See, Simon	(SUN, Singapore)

Sekiguchi, Satoshi	(AIST GTRC, Japan)
Shimojo, Shinji	(Osaka Univ., Japan)
Stevens, Rick	(ANL, USA)
Tan, Tin Wee	(NUS, Singapore)

Sponsoring Institutions

We appreciate financial supports from the following institutes for LSGRID 2004.

- A Grant-in-Aid for Scientific Research on the Priority Area Genome Informatics from the Ministry of Education, Science, Sports and Culture of Japan
- Japanese Society for Artificial Intelligence
- RIKEN Genomic Sciences Center
- The Initiative for Parallel Bioinformatics, Japan

Table of Contents

Life Science Grid

Gene Trek in Procaryote Space Powered by a GRID Environment <i>Hideaki Sugawara</i>	1
An Integrated System for Distributed Bioinformatics Environment on Grids <i>Kenji Satou, Yasuhiko Nakashima, Shin'ichi Tsuji, Xavier Defago, Akihiko Konagaya</i>	8
Distributed Cell Biology Simulations with E-Cell System <i>Masahiro Sugimoto, Kouichi Takahashi, Tomoya Kitayama, Daiki Ito, Masaru Tomita</i>	20
The Architectural Design of High-Throughput BLAST Services on OBIGrid <i>Fumikazu Konishi, Akihiko Konagaya</i>	32
Heterogeneous Database Federation Using Grid Technology for Drug Discovery Process <i>Yukako Tohsato, Takahiro Kosaka, Susumu Date, Shinji Shimojo, Hideo Matsuda</i>	43
Grid Portal Interface for Interactive Use and Monitoring of High-Throughput Proteome Annotation <i>Atif Shahab, Danny Chuon, Toyotaro Suzumua, Wilfred W. Li, Robert W. Byrnes, Kouji Tanaka, Larry Ang, Satoshi Matsuoka, Philip E. Bourne, Mark A. Miller, Peter W. Arzberger</i>	53
Grid Workflow Software for a High-Throughput Proteome Annotation Pipeline <i>Adam Birnbaum, James Hayes, Wilfred W. Li, Mark A. Miller, Peter W. Arzberger, Phililp E. Bourne, Henri Casanova</i>	68
Genome-Wide Functional Annotation Environment for <i>Thermus thermophilus</i> in OBIGrid <i>Akinobu Fukuzaki, Takeshi Nagashima, Kaori Ide, Fumikazu Konishi, Mariko Hatakeyama, Shigeyuki Yokoyama, Seiki Kuramitsu, Akihiko Konagaya</i>	82

Parallel Artificial Intelligence Hybrid Framework for Protein Classification

<i>Martin Chew Wooi Keat, Rosni Abdullah, Rosalina Abdul Salam</i>	92
--	----

Parallelization of Phylogenetic Tree Inference Using Grid Technologies

<i>Yo Yamamoto, Hidemoto Nakada, Hidetoshi Shimodaira, Satoshi Matsuoka</i>	103
---	-----

EMASGRID: An NBBnet Grid Initiative for a Bioinformatics and Computational Biology Services Infrastructure in Malaysia

<i>Mohd Firdaus Raih, Mohd Yunus Sharum, Raja Murzaferi Raja Moktar, Mohd Noor Mat Isa, Ng Lip Kian, Nor Muhammad Mahadi, Rahmah Mohamed</i>	117
--	-----

Development of a Grid Infrastructure for Functional Genomics

<i>Richard Sinnott, Micha Bayer, Derek Houghton, David Berry, Magnus Ferrier</i>	125
--	-----

Building a Biodiversity GRID

<i>Andrew C. Jones, Richard J. White, W. Alex Gray, Frank A. Bisby, Neil Caithness, Nick Pittas, Xuebiao Xu, Tim Sutton, Nick J. Fiddian, Alastair Culham, Malcolm Scoble, Paul Williams, Oliver Bromley, Peter Brewer, Chris Yesson, Shonil Bhagwat</i>	140
--	-----

Mega Process Genetic Algorithm Using Grid MP

<i>Yoshiko Hanada, Tomoyuki Hiroyasu, Mitsunori Miki, Yuko Okamoto</i>	152
--	-----

“Gridifying” an Evolutionary Algorithm for Inference of Genetic Networks Using the Improved GOGA Framework and Its Performance Evaluation on OBI Grid

<i>Hiroaki Imade, Naoaki Mizuguchi, Isao Ono, Norihiko Ono, Masahiro Okamoto</i>	171
--	-----

Author Index	187
---------------------------	-----

Gene Trek in Procaryote Space Powered by a GRID Environment

Hideaki Sugawara

Center for Information Biology and DNA Data Bank of Japan (DDBJ),
National Institute of Genetics (NIG) 1111 Yata,
Mishima, Shizuoka 411-8540, Japan
`hsugawar@genes.nig.ac.jp`

Abstract. More than 100 microbial genomes have been sequenced since 1995 and thousands of microbial genomes will be sequenced in a decade. It implies that millions of open reading frames (ORFs) will be predicted and should be evaluated. Therefore, we need a high throughput system to evaluate the predicted ORFs and understand functions of genes based on comparative genomics. We established and applied a protocol for the prediction and evaluation of ORFs to genome sequences of 124 microbial that were available from the International Nucleotide Sequence Database as of June, 2003. We could carry out the evaluation of about 300,000 predicted ORFs based on clustering and horizontal gene transfer analysis thanks to the GRID environment. This paper introduces mainly the scheme of the GRID environment applied to the comparative genomics.

1 Introduction

Genomes of procaryotes are doubtless small comparing to human genome. However, prokaryote genes are more diverse than human genes. The Genome Information Broker (GIB) [1] is a database of complete microbial genomes in public domain. It contained genome data of 124 microbes as of May, 2003. The number of open reading frames (ORFs) averages 3,000, namely, GIB is also a database of more than 300,000 genes in total. The space composed of procaryotes genes is vast and we need a space ship to trek there. Genome sequences are fundamental parts of the ship, an assembly of data mining tools is the engine and GRID computing is the booster. We successfully applied a GRID environment composed of 5 sites in OBIGrid (<http://www.obigrid.org/>)[2] to the comparative genomics of microbes registered in GIB.

2 Materials and Methods

The machines were connected by VPN in OBIGrid for our study as shown in Fig. 1. They were Linux machines in National Institute of Genetics (NIG) (64CPUs), Japan Advanced Institute of Science and Technology (JAIST) (68-CPUs), RIKEN Genomic Sciences Center (GSC) (10CPUs), Japan Science and

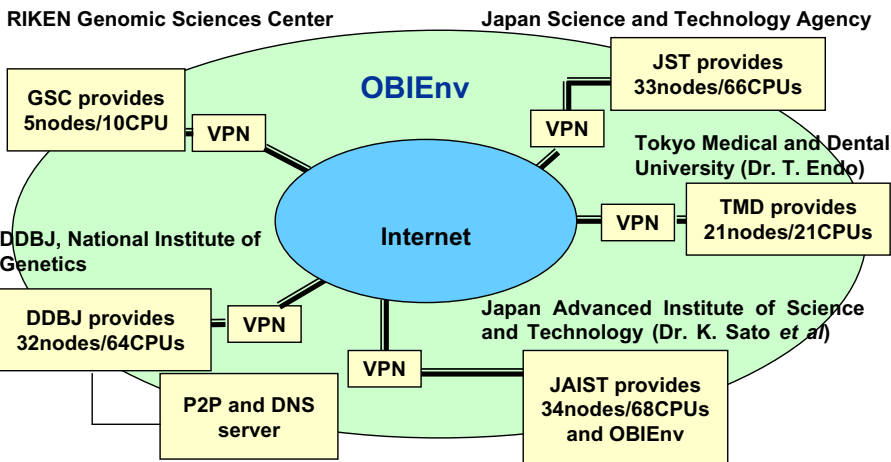


Fig. 1. The GRID environment used for the analysis of microbial genomes. Linux machines in the participating nodes were connected by VPN

Technology Agency (JST) (66CPUs), and Tokyo Medical and Dental University (TMD) (21CPUs). Thus the total number of CPUs is 229. In these machines, the Globus ToolKit version 2.4 [3] and OBIEnv [2] were installed. To monitor the status of the OBIEnv machines, a P2P server is set up in NIG. The snapshot of the monitoring is introduced in Fig. 2 and demonstrates that the work load was actually distributed to CPUs in the OBIEnv.

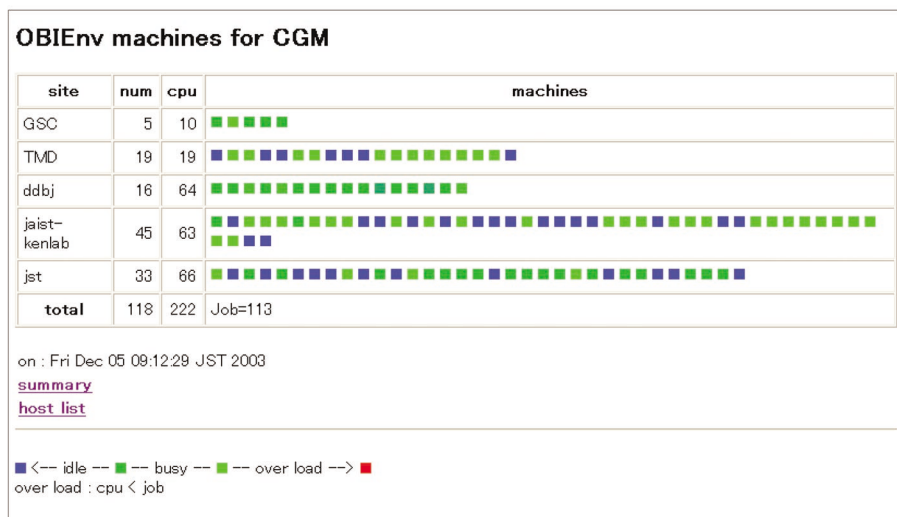


Fig. 2. Monitoring of the GRID environment. Small boxes in the table represent each CPU in nodes that participated in the project. Color codes are explained in the table

In DDBJ [7], we had identified open reading frames (ORFs) of the microbial genomes by our own protocol and stored them in a database. The protocol and details of the identification will be published elsewhere. We analyzed these ORFs by use of a GRID environment in two ways. In the case of microbial strains, genes are transferred among species, namely, horizontal gene transfer (HGT) occurs after species are established during the evolution [4]. Since we have genome sequences, we will be able to carry out a comprehensive analysis of HTG. In addition to the HTG analysis, it is another important issue to infer functions of ORFs from sequence data. The basic method of the inference is clustering ORFs expecting that ORFs in a same cluster will share the same function.

3 Results and Discussions

3.1 Horizontal Gene Transfer (HTG)

We constructed a gene model for each microbial genome to evaluate if the candidate ORFs is intrinsic or introduced by HTG [4]. If a candidate ORF is largely deviate from the model, the ORF may be from other species. Therefore, we had to construct 124 models and then compare a number of ORFs in 124 genomes with all the models. In this way, we are able to identify donor of ORFs of HTG as well.

We estimated that the computation of the HTG analysis took about 60 months with a CPU of 2GHz. In OBIEnv, the task was divided into 17,689 jobs as

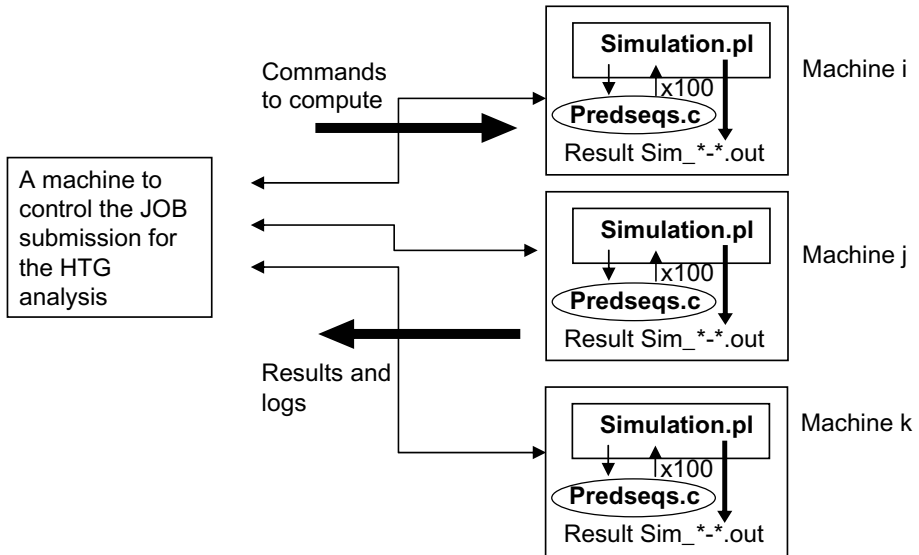


Fig. 3. Distribution of the HTG jobs in OBIEnv. The ORF sequences are distributed among machines and programs of Simulation.pl and Predseqs.c identify candidates of horizontally transferred genes. The total number of the JOBS was 17,689

shown in Fig. 3 to be completed in 18 days, although the network and some CPUs were down from time to time. Therefore, it will be quite feasible to repeat the analysis of HTG, whenever the data is update and a new genome is determined.

Results of the HTG analysis are stored in a database and a sample view of the database is introduced in Fig. 4. The top block in the figure displays

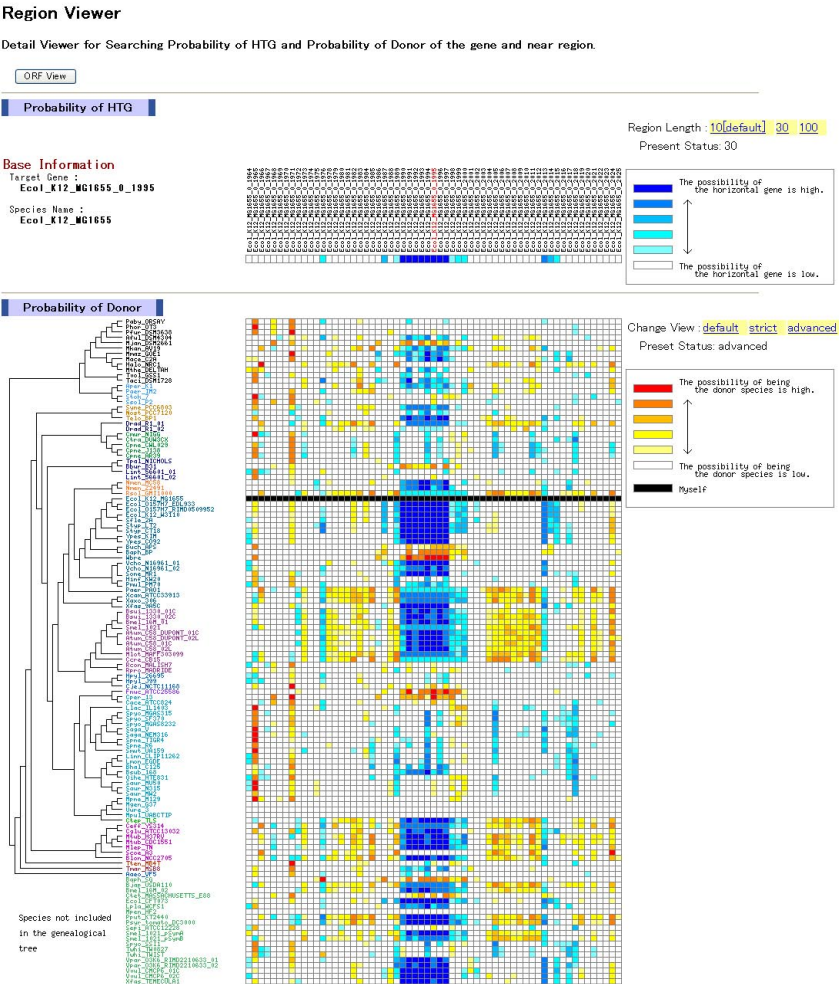


Fig. 4. Some results of horizontal transfer of genes (HTG). Results of the HTG analysis are stored in a database. A view of the database is a matrix of species vs genes in a species. This figure is a matrix of *E. coli* K12 (MG1655) genes and species whose genome sequences are publicly available. Bluish color of the cell in the matrix represents the probability of horizontal transfer of *E. coli* genes. The darker the color is, the higher the probability is. The warm color shows the probability of the donor of the horizontal transfer genes of the species to *E. coli*

ORFs of *E. coli*. Identifiers of ORFs in our system are written vertically there. Colors of cells under each ORF tell the probability of horizontally transferred gene. A blank cell means that the ORF is intrinsic to *E. coli*. On the other hand, a dark blue cell notifies that the corresponding ORF is probably transferred from other species, namely, horizontally transferred from a donor species of the ORF. The left tree in Fig. 4 displays a phylogenetic tree of microbes whose genome sequences are publicly available, e.g., from GIB. The donor species of the ORF notified by the dark blue cell is identifiable in Fig. 4, if you look down the corresponding column until you find a warm color cell. The red cell strongly suggests that the ORF of *E. coli* was transferred from the species in the same line as the red cell is. Thus the database of HTG provides information about not only genes horizontally transferred but also donor species.

3.2 Clustering of ORFs

We clustered 354,606 ORFs in total by SODHO [5] to retrieve information on functions of proteins from amino acid sequences of ORFs. With a CPU of 2 GHz, it would have taken 50 days to complete the clustering. In OBIEnv, we separated the all to all comparison of ORFs into 999 jobs to assign a job to a CPU for a parallel computing. In this way, it took only 17 hours to complete the computation. We evaluated the quality of the cluster by mapping to the motif in InterPro [6]. An example of the mapping is available in Table 1. We may

Table 1. A result of analysis of ORFs by use of InterPro. All the ORFs are compared with the database of InterPro. This table shows InterPro IDs of 6 ORFs that are member of a cluster found by SODOH

Name of ORFs	IPR002528	IPR001064	IPR00678
PFDSM[1850] Pfur_DSM3638:.faa_C10	+	+	
AAVF5[101] Aaeo_VF5:.faa_C10	+		+
PAORS[366] Paby_ORsay:.faa_C10	+	+	
PHOT3[1861] Phor_OT3:.faa_C10	+	+	
TTMB4[1686] Tten_MB4T:.faa_C10	+		
TTMB4[234] Tten_MB4T:.faa_C10	+		

expect that member ORFs of a cluster share the same motif among them, if the clustering is biologically meaningful. The six member ORFs share the same InterPro ID of IPR002528, although some of the ORFs hit to other InterPro IDs as well. Therefore, the correspondence the cluster in Table 1 is comparatively well defined. The precise and global evaluation of the clustering by SODHO is still under way.

3.3 Need of Computer Resources for Comparative Genomics

We analyzed 124 microbial genomes that had been disclosed by May, 2003. You will find genomes of 185 microbial strains as of August, 2004, i.e., 4 genomes a month were recently sequenced on average. We are able to expect that a number of microbial genomes will be continuously sequenced. Some microbes are closely related and even multiple strains belong to the same species. Other microbes are distant in the phylogenetic tree with each other. Therefore, the comparative genomics will be a powerful tool to understand especially the dynamics of genomic evolution and gene functions of microbes. Based on this observation, we are afraid that we need keep expanding the computer resources to compare a number of objects. Otherwise, we will not be able to complete a set of analysis before a new genomic sequence is available.

The computation will be left behind a tidal wave of genomic data, if an expandable and flexible large scale computing facility. The International Sequence Database (INSD) exceeded 30 millions entries and 30 giga nucleotides in the year of 2003 and will keep expanding every year as much as 1.5 times. Protein sequence database such as InterPro will expand too. In this study, we did not use a GRID environment to matching every candidate ORFs to InterPro. We need certainly a GRID environment or a large scale PC cluster to apply the InterPro analysis to 185 microbes as of August 2004 and afterwards.

The INSD already includes hundreds of genomic sequences of such wide variety of species as viruses, microbes, plants, animals and human. The INSD will capture thousands of genomes in the near future. Therefore, it is obvious that GRID environment will be the infrastructure of comparative genomics that traverse all the species to understand the universe of life phenoma.

Acknowledgements

The GRID computer environment in NIG was supported by Life Science System Division of Fujitsu Limited. This work has been partly supported by BIRD of Japan Science and Technology Agency (JST) and also partly by the Grant-in-Aid for Scientific Research on Priority Area "Genome Information Science", Ministry of Education, Sports, and Science (MEXT), Japan

References

1. Fumoto, M., Miyazaki, S., Sugawara, H.: Genome Information Broker (GIB): data retrieval and comparative analysis system for completed microbial genomes and more. *Nucleic Acids Res.*, 30(1) (2002) 66–68
2. Konagaya, A., Konishi, F., Hatakeyama, M., Satou, K.: The Superstructure toward Open Bioinformatics Grid. *New Generation Computing*, 22 (2004) 167–176
3. <http://www.globus.org/>

4. Nakamura, Y., Itoh, T., Matsuda, H., Gojobori, T.: Biased biological functions of horizontally transferred genes in rokaryotic genomes. *Nature Genetics*, 16 (2004) 760–766
5. Naitou, K., Kawai, M., Kishino, A., Moriyama, E., Ikeo, K., Ina, Y., Ikezaka, M., Satou, H., Gojobori, T.: The implementation of parallel processing for molecular evolutionary analysis using the highly parallel processor (in Japanese). *Joint Symposium on Parallel Processing'90*, (1990) 329–226
6. <http://www.ebi.ac.uk/interpro/>
7. DDBJ (<http://www.ddbj.nig.ac.jp/>), EMBL (<http://www.ebi.ac.uk/>) and GenBank (<http://www.ncbi.nlm.nih.gov/>)

An Integrated System for Distributed Bioinformatics Environment on Grids

Kenji Satou^{1,2}, Yasuhiko Nakashima³, Shin'ichi Tsuji³,
Xavier Defago^{4,5}, and Akihiko Konagaya⁶

¹ School of Knowledge Science,
Japan Advanced Institute of Science and Technology
`ken@jaist.ac.jp`

² BIRD, Japan Science and Technology Agency (JST)

³ NEC Software Hokuriku, Ltd.
`y-nakashima@pu.jp.nec.com`,
`s-tsuji@pj.jp.nec.com`

⁴ School of Information Science,
Japan Advanced Institute of Science and Technology
`defago@jaist.ac.jp`

⁵ PRESTO, Japan Science and Technology Agency (JST)

⁶ Riken Genomic Sciences Center Computer Science
`konagaya@gsc.riken.jp`

Abstract. In this paper, an integrated system called OBIEnv, which has been developed on OBIGrid, is described. In addition to automatic database transfer and deployment, it provides various functionalities for transparent and fault-tolerant processing of bioinformatics tasks on Grid. A feasibility study on the analysis of horizontal gene transfer was done using 119 heterogeneous Linux nodes in 5 different sites, and OBIEnv proved its applicability to practical problems in bioinformatics.

1 Introduction

Based on the explosive yield of experimental data on biomolecules, a research area called bioinformatics has grown remarkably in the past decade. This growth has been caused by several factors, such as, success of genome projects on human and model organisms, developments of new protocols and equipments for high-throughput experiments in molecular biology, and rapid growth of the Internet and WWW. Furthermore, traditional and advanced technologies in computer science have been applied to accelerate all the aspect of life sciences. Scientists nowadays are routinely using computers and networks in experiments, search, analysis, and visualization.

While the amount of experimental data is increasing exponentially, focus of genome analysis is shifting from identification of facts (e.g. the sequences and structures of genes and proteins) to conjecture of relationships, like gene network[1] and interaction pathways[2] in a cell. Since the latter requires combinatorial computation, and the search space itself is exponentially extending[3],

researchers in bioinformatics are suffering from chronic shortage of computing power. Therefore, utilization of Grid computing technology is strongly needed also in this research area.

Starting from metacomputing in late 80's, the concept of Grid computing has evolved giving rise to a variety of research projects[4, 5, 6], which can be classified according to different aspects: Compute Grids, Data Grids, Access Grids, Campus Grids, etc.[7] Among them, Open Bioinformatics Grid (OBIGrid) project is one of the comprehensive Grid project in Japan. Aiming at the construction of a practical infrastructure for bioinformatics, OBIGrid[8, 9] has adopted a virtual private network (VPN) in order to guarantee transparency and security issues, and several research and development projects are running based on it.

In this paper, an integrated system called OBIEnv, which has been developed on OBIGrid, is described. Though it adopts the Globus Toolkit as its basic middleware for remote command execution, it provides various and original functionalities for distributed, high-throughput, adaptive, and transparent computing, as well as database management in bioinformatics.

2 Design Issues

In the earliest stages of OBIEnv project, we assumed the following things as a basis for the design of our system.

- Nodes are heterogeneous in its hardware and software. While there might be Linux boxes and clusters, big SMP machines with Unix (e.g. Sun Fire 15k) can coexist on a Grid.
- Unlike cluster computing, in case of distributed computing with nodes scattered over the Internet, instability caused by node and network failures is frequent.
- Bandwidth of wide area network is drastically narrower than that of local area network.
- While there exist many types of computation in bioinformatics, one of the most frequently executed computation is processing a great number of similar tasks independent to each other. For example, all-to-all homology search with a sequence database is a typical one. It requires a job including many BLAST executions with the same search space (database) and different query sequences. This type of computation is classified into embarrassingly parallel processing, and it is well-suitable for Internet-wide distributed computing.
- Though we assume the above type of computation, it is impossible to estimate how long time is needed to process each task. In general, it depends on the application, its input, node characteristics (e.g., CPU type, memory size, I/O speed), and so on.

Next, we specified the requirements to develop an easy-to-use bioinformatics environment on Grid.

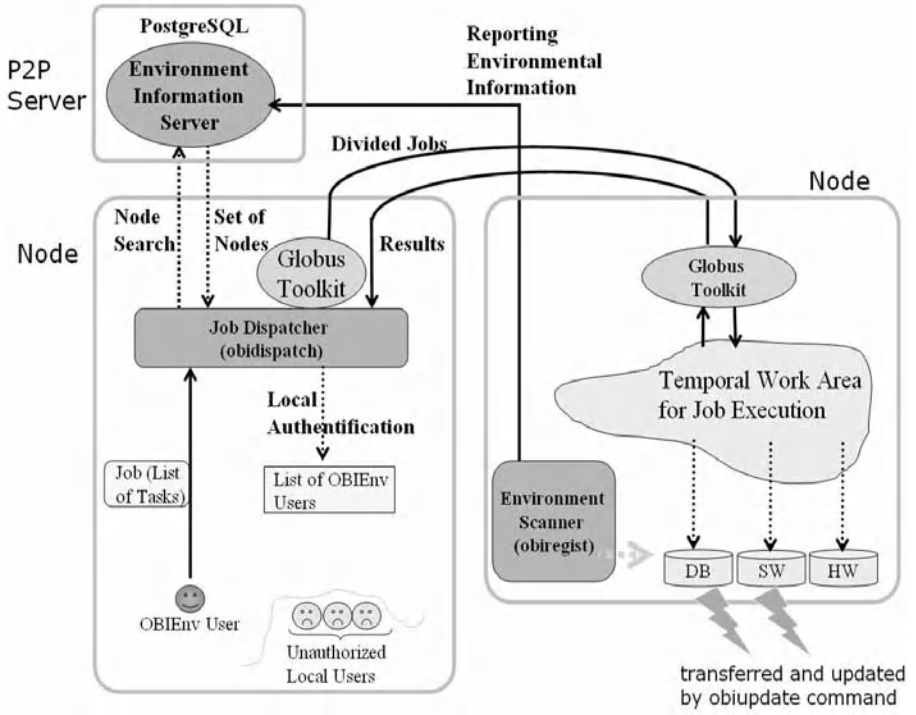


Fig. 1. Design overview of OBIEnv

- Deployment and update of common databases and applications is essential. It should be performed automatically as much as possible. Of course, network bandwidth and computing power should be used effectively for fast transfer.
- Information about the existence of databases and applications in a computation node has to be reported, gathered, and accessible for transparent computing. Furthermore, this information should be managed transparently to the user.
- Database transfer and job processing must be adaptive to the dynamically changing situation of nodes and networks, including slowdown and failure.

Finally, we designed OBIEnv as shown in Fig. 1. It consists of the following three components:

- *P2P server* for gathering node information about what kind of hardware, software, and database are deployed in each node. Update of the information is done in bottom-up manner (reporting from nodes to P2P server).
- *Job dispatcher* for easy and transparent distributed processing. Given a set of similar tasks, it dispatches them to the nodes with required environment. While it is running with the specified constraints, it occasionally searches a set of appropriate nodes by making a query to the P2P server, then adapts to dynamically changing situation of nodes (e.g. addition and retire of nodes).

- *Standard Software Environment (SSE)* to deal with platform heterogeneity. By assuming that all the nodes have essentially the same set of software including GNU tools, programming languages, and bioinformatics applications, a user can safely scatter a set of tasks to heterogeneous Unix/Linux systems. Besides the above free software, a series of programs is included for job dispatching, database update, and node information reporting to P2P server. Wrapper programs for bioinformatics applications are also included for easy execution in the nodes dispersed over OBIGrid.

Note that in this model, a representative user account is adopted. Globus Toolkit requires user accounts and home directories in all remote nodes for job processing. However, we think this requirement is too strict for our purpose. As shown in Fig. 1, local and site-level authentication is sufficient, and given a rich SSE, the need for user’s permanent program files on remote machines could be reduced.

3 Implementation Issues

On the P2P server, a PostgreSQL daemon is running for search and update of information. Fig. 2 illustrates what kind of information is stored in it.

To avoid too frequent access to the P2P server, most of the hardware, benchmark, and OS information are not frequently updated. Database information is typically updated daily. Information about job dispatching and database transfer by rsync is updated on demand. In order to prevent malicious updates from inside the OBIGrid, access to the port of PostgreSQL is controlled by using iptables on the P2P server.

The job dispatcher is implemented in Perl and Java. Given a set of tasks and constraints, the dispatcher checks user authentication. Then, based on the

Table 1. List of specifiable constraint variables for job execution. In addition, values and SQL operators for comparison (=, <, >, ~, LIKE, etc.) are included in a constraint

Constraint variable	Meaning
HOST	host names
SITE	site names
SMP	the number of CPUs
MEM	memory size
HDD	disk size
CPU_MODEL	model name of CPU
CLOCK	CPU clock
INDEX	memory index by nbench-byte
IINDEX	integer index by nbench-byte
FINDEX	float index by nbench-byte
DB	database names
DBV	database names and versions

hostname	crobi01.obigrid.org				
Hardware Information					
cpu_model	Intel(R) Pentium(R) III CPU family 1133MHz				
cpu_speed	1130	cpu_smp	2		
memory	1006.0	hdd	73643.0		
memory_index	4.925	integer_index	4.234		
float_index	10.362	task_max	2		
site	jaist-kenlab	netpath	switch1.vja01		
lastupdate	2003-07-31 11:34:47.035299				
actime	2003-10-31 14:04:10.885393				
Database Information					
name	version	size	lastupdate		
pdb	2003.08.20	12946454895	2003-08-21 15:24:02.682332		
trembl	2003.08.19	3762558765	2003-08-21 15:24:02.643919		
unigene	2003.08.20	5414272581	2003-08-21 15:24:02.496265		
Extra Software Information					
type	name	version	path	note	lastupdate
OS	Linux	2.4.18-10smp		Red Hat Linux 7.3 2.96-110	2003-07-31 11:34:47.635006
Standard Software Information					
version	os	os_version	size	lastupdate	
2003.08.25	Red Hat Linux	7.3 2.96-110	312524800	2003-09-02 10:07:09.505561	
Job Information					
num	dispatch host				
2	crobi01.obigrid.org				
	crobi01.obigrid.org				

Fig. 2. Node information stored in the P2P server

constraints, it searches appropriate nodes for job execution. Table 1 and Fig. 3 show specifiable constraints and two examples of task description, respectively. After the job dispatcher started, it starts to dispatch the tasks to the nodes. The dispatching policy is simple: based on the number of CPUs in a node, the dispatcher attempts to utilize all of them. Since the dispatcher does not care about the load average of a node, if two or more dispatchers are running in a Grid, an overload of nodes can be provoked. However, this is not a critical problem, since relatively few users have massive tasks to be processed. In addition, even with this simple policy, nearly correct load balancing is feasible if the number of tasks is sufficiently large. After the execution of a task on a node, computation results including standard output and standard error are returned to the dispatcher, which stores the results using SQLite for easy management of massive results

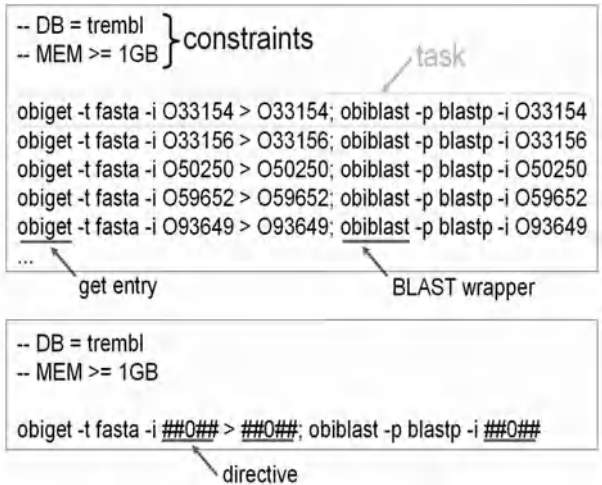


Fig. 3. Two examples of task description. Each line corresponds to one task except the line beginning with “--”. obiget and obiblast are programs included in SSE. To avoid such redundant descriptions illustrated in the upper example, directives are allowed in task description. In case of the lower example, the directive `##0##` is instantiated by STDIN with TSV format given to the job dispatcher (more generally, directives 0, 1, 2, ... are instantiated by each tab-separated values in a line of STDIN to the job dispatcher)

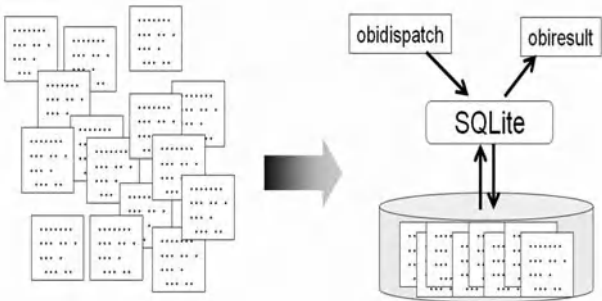


Fig. 4. Easy handling of computation results. To avoid littering massive files containing computation results, SQLITE was adopted for organized storing of massive results (right side)

(SQLite is an endian-free and embeddable DBMS without daemon program). Using a program obireult included in SSE, a user can easily search, select, and extract the specified results of task processing (Fig. 4). Other functionalities of job dispatcher are illustrated in Fig. 5 – 8.

In the framework of OBIEnv, databases are first mirrored and preprocessed for deployment. Typically, flat text files, containing various information about

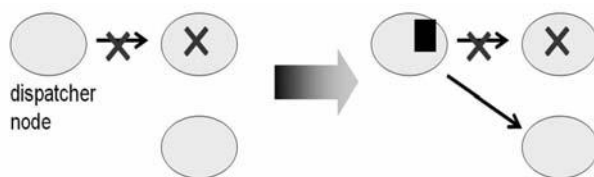


Fig. 5. Tolerance to various failure. In case of abort of task caused by node or network failure, the job dispatcher re-dispatches the task to other working node. While running, a list of defective nodes is maintained for repressing frequent retry to them

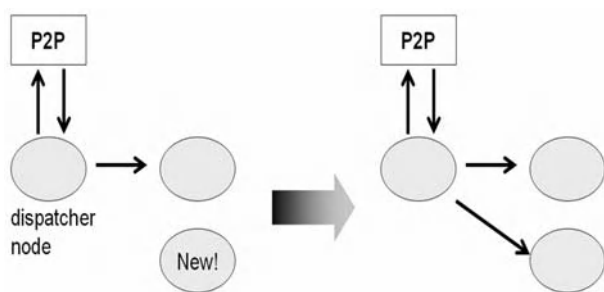


Fig. 6. Dynamic hiring of new nodes. While running, the job dispatcher updates a set of available nodes by periodical polling to P2P server. If it found a new node which satisfies the constraints in task file, it starts dispatching to the node, too

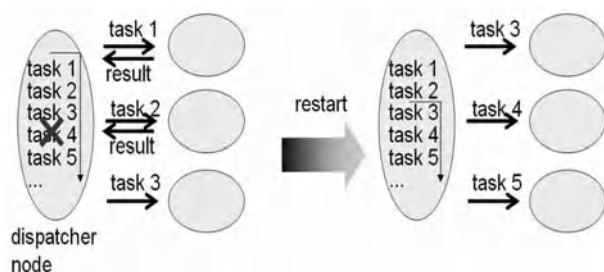


Fig. 7. Tolerance to the dispatcher's sudden death. Even if the job dispatcher aborted, a user only needs to restart it in the same way as its first execution. In this figure, though the same task list is given to the restarted dispatcher, the tasks which have already completed normally (i.e. task 1 and 2) are automatically skipped and others (task 3, 4, and 5) are dispatched. So, a user can freely abort and restart it. This function is practically useful for massive task processing

biomolecules, and FASTA format files, containing only the IDs, names and sequences for homology search, are prepared. Then, these files are indexed and stored by using SQLite again for fast retrieval of an entry and portability across

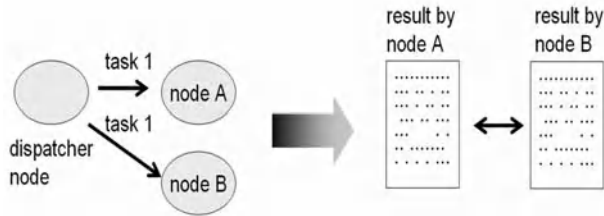


Fig. 8. Redundant execution and result comparison. If a user gave the redundant execution option to the job dispatcher, it tries to dispatch the same task to two different nodes. A program obiniq performs simple comparison on a pair of results for each task, and reports significantly different cases based on statistical evaluation

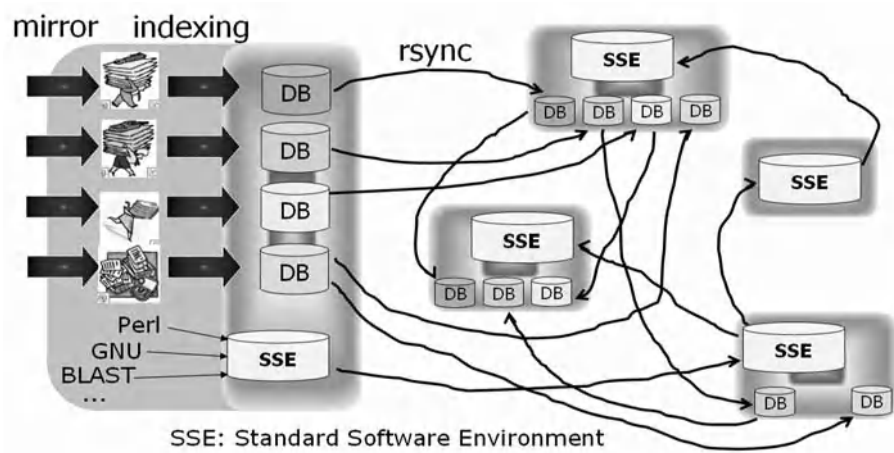


Fig. 9. Database transfer flow

the different platforms. After that, all the files including index files are listed in a file, a set of database files including the list file is deployed in a directory for transfer. The directory name is defined from database name and version number in the form of year/month/date. Finally, the deployment of the new version of a database is notified to P2P server. As in Fig. 9, if there are other nodes configured to transfer the newest version of the same database, they will start transfer by invoking the database update program (typically from cron).

For fast and efficient transfer of databases and software, we adopted rsync since it can provide stable and compressed transfer of files. Based on rsync, we implemented the following functions for rapid and adaptive transfer.

- According to the prepared file list of a database and checksum of it, the updater program transfers them first, then attempts to transfer each files in parallel.

Table 2. List of software in the standard software environment

GNU tools	fileutils-4.1, sed-4.0.7, gawk-3.1.1, grep-2.5, gzip-1.2.4, tar-1.13, diffutils-2.8.1, less-381, textutils-2.1, bash-2.05b, tcsh-6.12.00
Perl	perl-5.8.0 with Pg-2.0.2, bioperl-1.2.1, DBI-1.37, Data-ShowTable-3.3, DBD-SQLite-0.25, DBD-mysql-2.1026
Java	j2sdk1.4.2 with mysql-connector-java-3.0.8-stable, postgresql-7.3.3, javasqlite-20030613
Python	Python-2.2.3 with egenix-mx-base-2.0.4, Numeric-23.0, biopython-1.10
Bioinformatics	blast-2.2.6, clustalw1.83, fasta3
DBMS	sqlite-2.8.4
Benchmark	nbench-byte-2.1, netperf-2.2pl3
Original	obiblast, obichk, obidelete, obidispatch, obiget, obiinfo, obimonitor, obiregist, obiresult, obiupdate, obidispatch.jar (Java library)

- In addition, it tries to transfer a file even if a remote node has it but has not finished transferring the whole database. In other words, the transfer is pipelined.

As to SSE, first we carefully chose a minimum and essential set of software. Current list of software is shown in Table 2, although it is growing due to requests of OBIEnv users.

4 Current Status and Practical Application

Fig. 10 shows a summary of nodes with OBIEnv system. You can see at least two groups of nodes (left and right) are cooperatively working in OBIGrid (OBIEnv allows two or more different P2P servers in a Grid).

Right group in Fig. 10 was organized for general purpose, while left one for a project called “Gene Trek in Prokaryote Space” led by Center for Information Biology and DDBJ, National Institute of Genetics. In the project, OBIEnv proved in practice that it was able to successfully process about 15 thousands of similar tasks for the analysis of horizontal gene transfer within 18 days by using 119 heterogeneous Linux nodes including dual Pentium III, quad Xeon, and single Pentium 4 machines. CPU clocks and memory sizes vary from 900 MHz to 2.4 GHz, and from 1 GB to 8 GB, respectively. These machines were spanning 5 different administrative domains participating in OBIGrid project. Fig. 11 illustrates a job execution in the project.

Furthermore, it was proved that OBIEnv could adapt to serious failure of network. Due to the scheduled power cut in a weekend at the center point of VPN network, DDBJ was isolated from other 4 sites during massive task processing. Then OBIEnv automatically detected the failure and kept processing the tasks by using only 16 nodes in DDBJ. After the reestablishment of VPN connection on Monday, OBIEnv automatically reemployed the machines in other sites (see

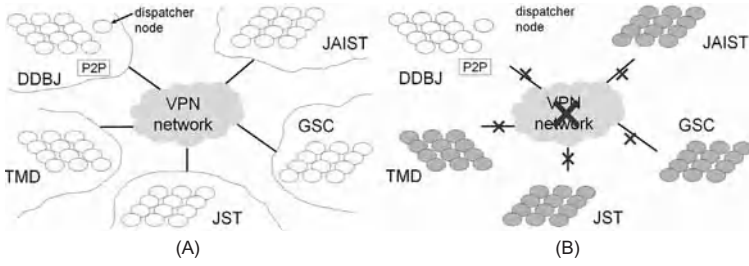


Fig. 12. When started, the job dispatcher was using all the nodes in 5 sites (see (A):white ovals are the nodes which are processing tasks). During the isolation caused by scheduled power cut, it kept running by using only the nodes in DDBJ (see (B): gray ovals are idle nodes). After the powercut, the job dispatcher automatically reemployed the nodes in other 4 sites (see (A))

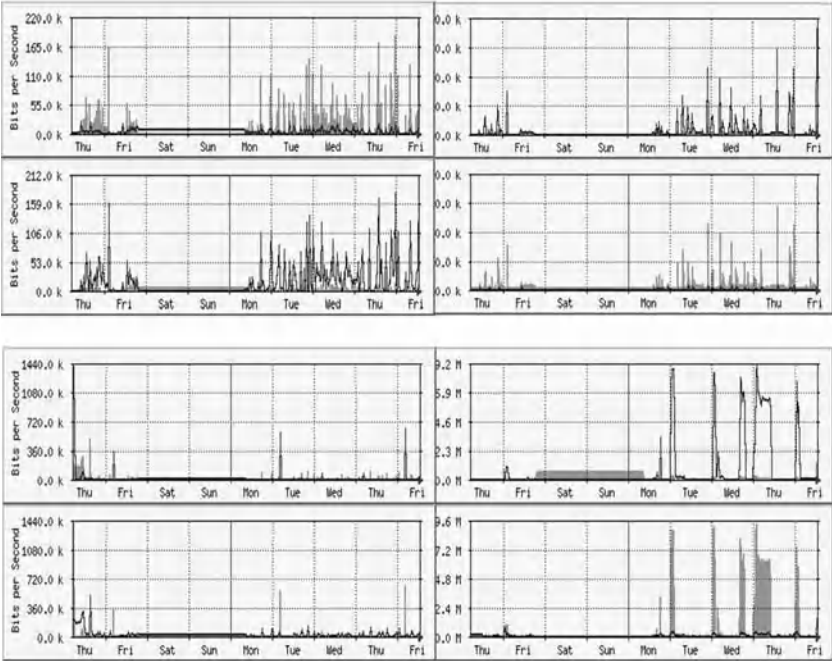


Fig. 13. Traffics from/to DDBJ and 3 sites out of 4 in a weekend

Fig. 12 and 13). During this failure, we did not need to stop/restart the job dispatcher by hand.

Detailed result of horizontal gene transfer analysis project is briefly described in [10], and in a forthcoming paper.

5 Conclusion and Future Works

In this paper, we described OBIEnv, a transparent and easy-to-use integrated system for bioinformatics. Its design and implementation is done with careful discussions and choices based on the mainstream of computations in bioinformatics, and we believe its practical effectiveness for easy maintenance and job processing on heterogeneous computers in different sites in a Grid can greatly help scientific research in bioinformatics. OBIEnv has been continuously improved and the following functionalities are now about to demonstration experiment:

- In addition to Globus Toolkit, rsh and ssh are adopted. So, OBIEnv is now available not only for Grid computing but also for cluster computing.
- For safe and accurate computing, a user of OBIEnv can specify redundant computation. The same task is executed on two different nodes, redundant results are compared, and significant differences in two results for the same task are reported.
- More intelligent and efficient algorithm for database transfer is adopted for optimal utilization of the bandwidths of wide and local area networks.

Acknowledgements

This work was partially supported by a Grant-in-Aid for Scientific Priority Areas (C) “Genome Information Science” from the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

References

1. Imoto, S., Goto, T. and Miyano, S.: Estimation of genetic networks and functional structures between genes by using Bayesian network and nonparametric regression, Pacific Symposium on Biocomputing 7:175–186 (2002).
2. Ogata, H., Goto, S., Fujibuchi, W. and Kanehisa, M.: Computation with the KEGG pathway database. BioSystems 47, 119–128 (1998).
3. Growth of major databases, http://www.genome.ad.jp/dbget/db_growth.gif
4. Foster, I. and Kesselman, C.: The Globus Project: A Status Report, Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop, 4–18 (1998).
5. TeraGrid, <http://www.teragrid.org>
6. Hoppe, H.-C. and Mallmann, D.: EUROGRID — European Testbed for GRID Applications, GRIDSTART Technical Bulletin, October (2002).
7. Foster, I.: What is the Grid? A Three Point Checklist, GRIDToday, July 20 (2002).
8. Konagaya, A.: OBIGrid: Towards a New Distributed Platform for Bioinformatics, 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02), 380–381 (2002).
9. Konagaya, A., Konishi, F., Hatakeyama, M., and Satou, K.: The Superstructure Toward Open Bioinformatics Grid, Journal of New Generation Computing, Vol. 22, No.2, 167–176 (2004).
10. Sugawara, H.: Gene Trek in Prokaryote Space powered by a GRID environment, Proc. of the First International Workshop on Life Science Grid (LSGRID2004), 1–2 (2004).

Distributed Cell Biology Simulations with E-Cell System

Masahiro Sugimoto^{1,2}, Kouichi Takahashi^{1,*}, Tomoya Kitayama³,
Daiki Ito³, and Masaru Tomita¹

¹ Institute for Advanced Biosciences, Keio University,
Tsuruoka, Yamagata 997-0035, Japan
msugi@sfc.keio.ac.jp, shafi@e-cell.org, mt@sfc.keio.ac.jp,
<http://www.e-cell.org>

² Bioinformatics Department, Mitsubishi Space Software Co. Ltd.,
Amagasaki, Hyogo, 661-0001, Japan
sugi@cbo.mss.co.jp

³ Laboratory for Bioinformatics,
Keio University Fujisawa, 252-8520, Japan
{tomoyan, s02082di}@sfc.keio.ac.jp

Abstract. Many useful applications of simulation in computational cell biology, e.g. kinetic parameter estimation, Metabolic Control Analysis (MCA), and bifurcation analysis, require a large number of repetitive runs with different input parameters. The heavy requirements imposed by these analysis methods on computational resources has led to an increased interest in parallel- and distributed computing technologies.

We have developed a scripting environment that can execute, and where possible, automatically parallelize those mathematical analysis sessions transparently on any of (1) single-processor workstations, (2) Shared-memory Multiprocessor (SMP) servers, (3) workstation clusters, and (4) computational grid environments. This computational framework, E-Cell SessionManager (ESM), is built upon E-Cell System Version 3, a generic software environment for the modeling, simulation, and analysis of whole-cell scale biological systems.

Here we introduce the ESM architecture and provide results from benchmark experiments that addressed 2 typical computationally intensive biological problems, (1) a parameter estimation session of a small hypothetical pathway and (2) simulations of a stochastic *E. coli* heat-shock model with different random number seeds to obtain the statistical characteristics of the stochastic fluctuations.

1 Introduction

Computational biology requires high-performance computing facilities. There are many parallel biological applications that can be used in PC cluster environments, e.g. HMMer, FASTA, mpiBLAST, PARACEL BLAST, ClustalW-MPI[1], Wrapping up BLAST[2], and TREE-PUZZLLE[3]. We have developed

* Corresponding author.

an integrated software environment for computational cell biology, the E-Cell System[4, 5, 6, 7]. It is an object-oriented software suite for modeling the simulation and analysis of large-scale complex systems such as biological cells. E-Cell has a hierarchical parallel computing scheme in which (1) simulation sessions can be concurrently executed on remote computation nodes in grid or cluster environments, and (2) each run of the simulator can be parallelized on a shared-memory multi-processor computer employing multiple threads. Here we discuss the scheme for session-level parallelism, or distributed computing. Elsewhere we explained the other scheme, parallelization of Gillespie's Stochastic Simulation Algorithm (SSA) on E-Cell3[8], or simulator-level parallelism. As many simulation applications in computational cell biology require repetitive runs of simulation sessions with different model- and boundary parameters, the distributed computation scheme is highly useful.

Some middleware to assign jobs to distributed environments is already available, e.g. the Portable Batch System (PBS, <http://www.pbs.org/>), Load Sharing Facility (LSF, <http://www.platform.com/>), and Sun Grid Engine (SGE, <http://www.sun.com/software/gridware/>) on the cluster level, and the Globus toolkit[9] on the Grid level. While these low-level infrastructures are extremely powerful, they are not compatible with each other nor are they readily accessible to the average computational biologists. Some higher-level middleware focusing on the use of bioinformatics applications has been developed. For example, the Discovery Net System[10] is an application for Genome Annotation, Talisman[11] is a component of the *my*Grid[12] project that provides a framework to produce web-based applications, OBiGrid[13] accommodates BLAST tasks in a grid environment, OBIYagns is a grid-based simulation environment for parameter estimations[14], and Nimrod/G is a job-scheduler system on dynamic global resources[15]. Many other projects are ongoing under the BioGrid project (<http://www.biogrid.jp/>). A distributed version of E-Cell, E-Cell2D[16], which has a client-server architecture where the client uses a Web browser, has been developed. Although this version of E-Cell is useful for classroom applications, it lacks the scripting feature necessary for the automation of mathematical analysis sessions.

Here we report the development of a distributed computing module of E-Cell System Version 3 (E-Cell3), E-Cell Session Manager (ESM), which can transparently support PC-, SMP-, cluster-, and grid environments. We present performance evaluations of parallel computations using ESM, (1) a parameter estimation of a small hypothetical pathway and (2) simulations of a stochastic *E. coli* heat-shock model with different random number seeds to obtain the statistical characteristics of the model.

2 Architecture

2.1 E-Cell3 and ESM

E-Cell3 is comprised of 3 layers: 1). a class library for cell simulation (libecs) and its C++ API (libemc), 2) a Python language wrapper of libemc, PyEcs

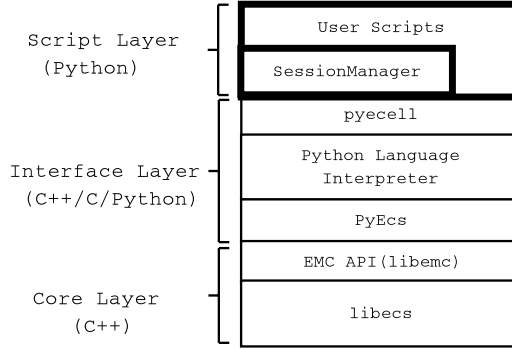


Fig. 1. ESM Architecture. The bottom layer includes a class library for cell simulation (libecs) and its C++ API (libemc). The top layer represents python front-end utilities such as SessionManager, GUI, and analysis tools. The middle layer (PyEcs, python interpreter, and pyecell) is the interface connecting the bottom and top layers

and pyecell, and 3) a library of various front-end and utility modules written in Python (Fig. 1). The pyecell library defines an object class called Session, which represents a single run of the simulator. ESM, constructed on top of the pyecell layer, enables the user to easily script procedures of mathematical analysis methods that instantiate many Session objects.

2.2 ESM Design

The fundamental design of ESM is shown in Fig. 2 as a class diagram. It is comprised of 3 classes, the SessionManager-, SessionProxy-, and SystemProxy class. The SessionManager class provides the user with a basic API to create and run simulation sessions. SessionManager generates and holds a SystemProxy object; it represents and communicates to the computing environment on which ESM is running (such as PC, SMP, cluster, or grid). On request, SystemProxy generates instances of SessionProxy; it corresponds to a process on PC- and SMP environments or a job on cluster- and grid environments, and holds the status of the process or job (waiting, running, recoverable error, unrecoverable error, or finished). The status chart of SessionProxy is shown in Fig. 3.

As depicted in Fig. 2, subclasses of Session and SystemProxy are instantiated and used by the SessionManager class according to the environment. For example, a pair of LocalSystemProxy and LocalSessionProxy is selected when the system is running on a single CPU PC or a SMP machine; the system uses SGESystemProxy and SGESessionProxy when the user request is to parallelize the computation on an Sun Grid Engine (SGE) parallel-batch system. On an SMP or a PC computer, they spawn ordinary processes in the local computer and use system calls to manage the tasks. On cluster and grid environments, these classes make contact with the system that manages the computing environment to submit jobs and obtain the job status.

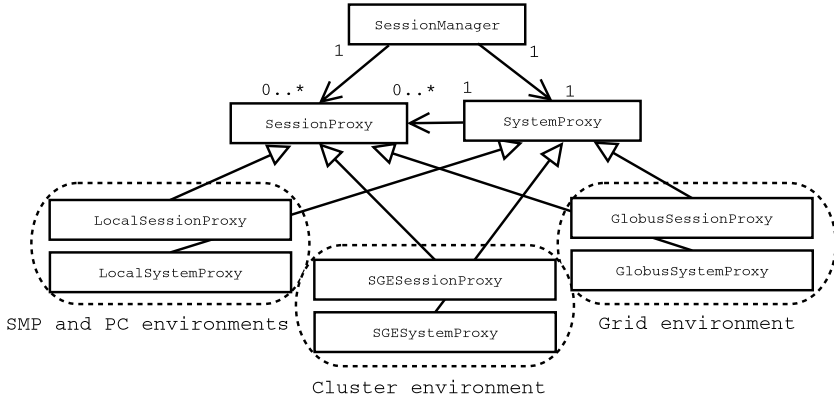


Fig. 2. ESM Design. The SessionManager class provides an API for user scripting. SystemProxy is a proxy of the computing environment such as the cluster- or grid environment. SessionProxy corresponds to a process or a job. LocalSystemProxy and LocalSessionProxy are used both on SMP- and single-CPU PC environments

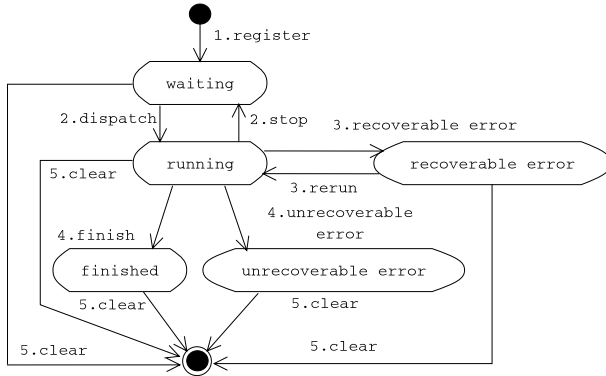


Fig. 3. Status chart representation of SessionProxy. (1) A SessionProxy is created when a job is registered; the initial status is 'waiting'. (2) It changes to 'running' when the registered job is dispatched to a computation node to run, if the run is suspended, it returns to 'waiting'. (3) When a recoverable error occurs, the status changes to 'recoverable error', and the system retries the run until it reaches a user-specifiable limit. If the retry limit is reached, the job goes to 'unrecoverable error'. (4) A job terminates in either the 'finished' or 'unrecoverable error' status, depending on its exit status. (5) Users can terminate the job by invoking the clear() method of the SessionProxy

2.3 ESM Processing Scheme

At least 3 types of files are necessary to run ESM; (1) a model file (EML, or E-Cell model description language file), (2) a session script file (ESS, or E-Cell


```

% ecell3-session-manager --environment=Local ems.py (1)
% ecell3-session-manager --environment=SGE --concurrency=30 ems.py (2)
% ecell3-session-manager --environment=Globus2 --concurrency=100 ems.py (3)

```

Fig. 4. Running ESM from command-line: This figure has three example command-lines that run the `ecell3-session-manager` command. ‘`--environment=`’ and ‘`--concurrency=`’ command-line arguments specify the computing environment and concurrency, respectively. (1) Local environment (single-CPU PC or SMP). The default concurrency in this example is 1. This can be changed by explicitly giving the `--concurrency=` argument. (2) An SGE run with 30 CPUs. (3) A Globus run. The number of simultaneous executions of jobs is limited to 100. All runs execute the EMS file ‘`ems.py`’

session script file), and (3) a Session Manager script file (EMS, or E-Cell session manager script file). Simple examples of command lines, an EMS, and an ESS are presented in Fig. 4, Fig. 5, and Fig. 6, respectively. Below we explain a typical flow of procedures in an EMS. Items (2) to (4) correspond to the bold comment lines in the script of Fig. 5.

(1) Set System Parameters

At least 2 system parameters, the computing environment and the concurrency, should be set when running ESM. The computing parameter specifies what type of facilities the ESM should use to run and control the jobs. The concurrency parameter specifies the maximum number of CPUs that the system can use simultaneously. These parameters are usually given as command-line arguments to the ‘`ecell3-session-manager`’ command, which runs an EMS indicated by the user. (Fig. 4)

(2) Register Jobs

The `registerEcellSession()` method in EMS registers a job. It accepts 3 arguments, (1) the session script (ESS) to be executed, (2) the optional parameters given to the job, (3) the input files to the script (at least the model file) that must be available to the ESS upon execution. In the example in Fig. 5, 100 copies of the session script ‘`runsession.py`’ are registered with a model file ‘`simple.eml`’. An optional parameter to the script, ‘`VALUE_OF_S`’ is also given to each session in the range 1, 2, ..., 100. The parameter is available to the ESS as a global variable. When a job is registered to the system, a `SessionProxy` object for the job is instantiated with a unique ID.

(3) Run

When the `run()` method is called, the registered jobs start executing. In this step, `SystemProxy` transfers the ESS file and all the other files to the execution environment (either a directory in the local machine or a remote computation node). Next, `SessionProxy` starts execution of the job. When `SystemProxy` confirms that all jobs are either finished or have an unrecoverable error, the `run()` method returns. It is also possible to use an asynchronous scheme in which the

```

MODEL_FILE='model.eml'
ESS_FILE='runsession.py'

#(2)Register jobs.
aJobIDList = []
for VALUE_OF_S in range(0,100):
    aParameterDict = {'MODEL_FILE':MODEL_FILE, 'VALUE_OF_S':VALUE_OF_S}
    # registerEcellSession(the script, parameters, files that the ESS uses)
    aJobID = registerEcellSession(ESS_FILE, aParameterDict, [MODEL_FILE,])
    aJobIDList.append(aJobID) # Memorize the job IDs in aJobIDList

#(3)Run the registered jobs.
run()

#(4)Examine the results.
for aJobID in aJobIDList: # Print the output of each job.
    print getStdout(aJobID)

```

Fig. 5. A sample EMS (E-Cell Session Manager Script): This script runs the session script 'runsession.py' 100 times with a changing parameter 'VALUE_OF_S'

```

loadModel( MODEL_FILE ) # Load the model.

S = createEntityStub( 'Variable:/:S' ) # Create a stub object of
# the simulator variable 'Variable:/:S'
S['Value'] = VALUE_OF_S # Set the value VALUE_OF_S given by the
# EMS in Fig.5 to the variable.

run( 200 ) # Run the simulation for 200 seconds.

message( S['Value'] ) # Print the value of 'Variable:/:S'.

```

Fig. 6. A sample E-Cell session script (ESS): This script runs a simulation model for 200 sec and outputs the value of the variable 'Variable:/:S' of the model after the simulation. The initial value of the variable is changed to the value 'VALUE_OF_S' given by the EMS

run() method returns immediately and the user must check the status of the jobs explicitly.

(4) Examine the Results

After running the simulation sessions, simulation results are obtained and examined. The example script in Fig. 5 prints the output of the simulations to the screen; getStdout(aJobID) shows the standard-out of the job specified by a job ID. If more runs of the simulation are necessary, go to (2).

3 Results

This section demonstrates and evaluates the performance of ESM with 2 types of simulation experiments in computational cell biology. The first example is an

automatic estimation of 4 kinetic parameters in a hypothetical small metabolic pathway. The second demonstration runs a stochastic model of the *E. coli* heat-shock response repeatedly with different random number seeds to obtain statistical features of the trajectory yielded by the model. Both types of numerical experiments are representative of the numerical experiments most commonly conducted in computational cell biology research projects.

3.1 Kinetic Parameter Estimation Using Genetic Algorithms (GA)

In computational cell biology, it is often the case that some model parameters, e.g. rate constants, are unknown while other experimental data, e.g. concentrations at steady states and time-courses of concentrations of some molecular species, are available. We developed a parameter-estimation tool that runs on ESM. This program implements a variation of GA that is an evolutionary algorithm to search the global minimum of a given multi-variate fitness function avoiding local minima[17]. A brief overview of the procedures used in this GA program is as follows: (1) Generate individuals and randomly distribute them over the search space. (2) Evaluate each individual with a user-defined fitness function. A square-error function between given and simulation-predicted trajectories is often used as the fitness function. (3) Select a couple of individuals from the group of individuals according to some probability, and (4) crossover the individuals. (5) Mutate each individual. (6) Go to (2), unless the fitness of the best individual meets a user-specified condition. In this study, the master-slave parallel GA was employed. The individuals are evaluated on parallel computational resources in procedure (2); other procedures are conducted on the master node.

We conducted a benchmark experiment using the pathway model shown in Fig. 7. This tiny model of a hypothetical metabolic pathway contains 5 molecular species and 5 reactions, one of which is a positive feedback reaction. Each reaction is represented using an irreversible Michaelis-Menten equation. This requires 2 rate constants, the Michaelis constant KmS , and the catalytic constant Kcf . Four of the 10 parameters are unknown and to be predicted by the parameter estimation tool.

The results are shown in Table 1 and Fig. 8. We measured the time required for the relative square-error between the given and the predicted trajectories to reach the levels of 10%, 7.5%, 5.0%, and 2.5%. We show 10, 20, and 30 CPU cases for each level of the relative error. An SGE system running on a Linux cluster was used for all timings.

3.2 Stochastic Simulation of *E. coli* Heat-Shock Response

Many cellular phenomena require stochastic simulations whose results must be discussed by means of statistical measures. Therefore it is common for a numerical simulator to run a model repeatedly with different random number seeds. We conducted numerical experiments to measure the time taken to run an *E. coli* heat-shock model[18] on ESM.

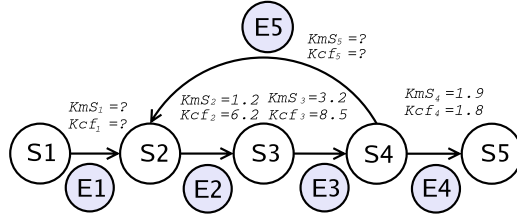


Fig. 7. An example model: S1, S2, S3, S4, and S5 are metabolites; E1, E2, E3, E4, and E5 are enzymes. All reactions are formulated by using Michaelis-Menten equations that have 2 parameters each, KmS and Kcf . The parameters for enzymes E1 and E5 are unknown and to be predicted

Table 1. Timings from the parameter estimation experiment using the GA. Of 10 rate constants in the model in Fig. 7, 4 are unknown. Four sets of training time-courses of concentrations of the 5 molecular species were given. The number of individuals is 100. Each row shows the times in sec required to reach 2.5%, 5.0%, 7.5%, and 10% of the relative error level between the given and the predicted time-courses. Each row has from 10 to 60 CPU cases, therefore, the table shows 24 results. A 40-node, 80-CPU Linux cluster running SGE was used. The CPUs were Pentium 4 Xeon 2.0GHz. Each node was connected by a 1000BaseT local area network

Relative	CPUs(Time s)					
Error %	10	20	30	40	50	60
2.5	11499	6022	4412	3354	2541	2089
5.0	1915	997	732	524	467	419
7.5	1273	604	474	373	304	283
10	627	346	250	183	152	147

The results are shown in Table 2. We measured the time required to run 100, 200, and 300 queries of 100-, 200-, and 300-sec simulations on a single CPU PC, a dual-CPU SMP machine, and a Linux cluster, setting concurrency to 1, 2, 5, 10, 20, and 30. The table, showing a total of 72 cases ($3 \times 3 \times 8$), indicates that timing increases roughly linearly with the number of queries on all the machine configurations. In the case with the shortest simulation time (100 sec), a 5-CPU run of the cluster runs as fast as the single-CPU PC; longer simulation times (500 and 1000 sec) lower the equilibrium point somewhere between 1 and 2-CPU runs of the cluster.

3.3 Discussions

The results of the numerical experiments presented in the previous section clearly show the benefits of using distributed computation in computational cell biology. Only when a small number of CPUs were used did the performance suffer from the overhead of parallel computing (see Table 2). This unacceptable overhead

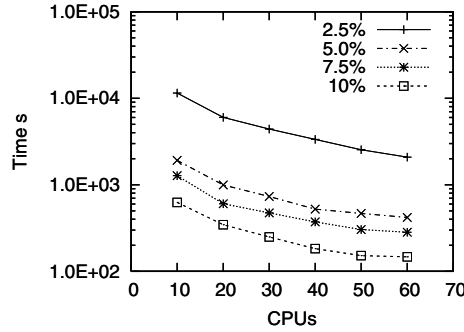


Fig. 8. Performance graph of the GA parameter estimation. Relative error levels of 10%, 7.5%, 5.0%, and 2.5% cases are shown with 10 to 60 CPU runs. The graph demonstrates that computation time increases steeply as the required relative error level decreases. See also Table 1

Table 2. Benchmark results of the stochastic heat-shock response model. We ran 100, 200, and 300 queries of simulations for 100-, 500-, and 1000 sec on a PC, a dual SMP, and 1, 2, 5, 10, 20, and 30 CPUs of a SGE Linux cluster. The CPUs, memory, and other hardware configurations were identical in all cases. The workstation cluster used for this experiments was the same as that used in section 3.1

Simulated Time (s)	#queries	PC	Environments & CPUs (Time s)							
			SMP	SGE						
		1	2	1	2	5	10	20	30	
100	100	302.8	153.4	1492.7	745.8	301.3	148.7	79.4	66.0	
	200	605.6	307.8	3006.7	1298.6	598.0	305.9	150.0	104.0	
	300	910.0	461.9	4501.3	2280.0	902.5	452.6	232.4	152.2	
500	100	1117.8	557.0	1530.4	770.2	303.8	165.5	79.6	65.0	
	200	2223.4	1120.4	3011.7	1520.5	631.3	351.1	174.9	135.3	
	300	3332.4	1677.2	4571.8	2268.1	912.4	475.6	258.2	158.5	
1000	100	2106.0	1088.3	2992.1	1496.5	603.5	298.0	157.4	128.9	
	200	4283.6	2186.5	6030.8	3011.7	1205.2	597.6	289.9	222.3	
	300	6571.8	3292.1	9039.8	4522.9	1802.1	912.5	463.7	312.6	

is attributable to the processing times required for job submission, transfer and data retrieval, however, these procedures are not necessary in PC- and SMP environments.

ESM can distribute simulation sessions of E-Cell transparently on virtually any distributed computation environments. All scripts can be written in Python language utilizing ESM’s user-friendly API methods. In fact, the design of ESM is so generic that it can run any ordinary Python scripts (not E-Cell sessions) if the registerJob()- rather than the registerEcellSession() method is used.

In homogeneous parallel computing environments such as shared-memory machines and PC clusters, it is relatively easy to schedule jobs to minimize the

total amount of processing time. In fact, even the simplest first-come-first-served scheduling scheme, which is the current version of ESS implements, works sufficiently well in many cases. However, heterogeneous environments like Globus Grid require more sophistication in scheduling and synchronization because remote computation nodes generally vary unpredictably with respect to their running and response times. Analysis methods that require all jobs to be finished concurrently, e.g. MCA and bifurcation analysis, may suffer from this fundamental weakness of the Grid. However, some other applications of cell biology simulation of an asynchronous nature may be able to overcome this problem. For example, the statistical computation of the stochastic model demonstrated here does not demand that all dispatched jobs be finished before it wraps up the simulation and proceeds to the next step of the analysis of the results. The same strategy might apply to GA parameter estimations if the algorithm is designed to allow evaluation of fitness functions on an incomplete set of individuals in the population. Various kinds of parallel GAs have already been developed[19, 20]. Island-type GAs[21], in which each 'island' has a subset of the whole population and evolves separately with asynchronous exchanges of individuals, is an example of these types of method.

Various kinds of analysis scripts that run on ESM are now under development. These include a program for the GA-based prediction of power-law based (GMA or S-System) gene-networks[22], a sensitivity analysis toolkit based on MCA[23], a bifurcation analysis toolkit that is used to estimate the stability of non-linear models, and Genetic Programming (GP)[24] for prediction of biochemical reactions mechanisms.

3.4 Conclusions

We developed a distributed computing module for E-Cell System, E-Cell Session Manager, or ESM. This software hides the details of job creation and management, and allows the user to write scripts of numerical experiments in Python language that runs transparently on any local, cluster-, and grid computing environments. Using this software, we demonstrated the benefits of distributed computation in computational cell biology research by presenting 2 demonstrations of numerical experiments, (1) parameter estimation using a GA and (2) stochastic simulation of the *E. coli* heat-shock response model. All software described here is available at <http://www.e-cell.org/>, as part of E-Cell Simulation Environment Version 3, which is OpenSource software under GNU general public license (GPL) version 2.

Acknowledgements

We thank Satya Arjunan and Bin Hu for fruitful discussions. This work was supported by a grant from the Ministry of Education, Culture, Sports, Science and Technology, a grant-in-aid from the 21st Century Center of Excellence (COE) program of Keio University, "Understanding and control of life's function via systems biology". a grant from the New Energy and Industrial Technology De-

velopment and Organization (NEDO) of the Ministry of Economy, Trade and Industry of Japan (Development of a Technological Infrastructure for Industrial Bioprocess Project), a grant from the Leading Project for Biosimulation, Ministry of Education, Culture, Sports, Science and Technology, and a grant from the Japan Science and Technology Agency (JST).

References

1. Li, K.B.: ClustalW-MPI: ClustalW Analysis using Distributed and Parallel Computing. *Bioinformatics* **19** (2003) 1585–1586
2. Hokamp, K., Shields, D.C., Wolfe, K.H., Caffrey, D.R.: Wrapping up BLAST and Other Applications for Use on Unix Clusters. *Bioinformatics* **19** (2003) 441–442
3. Schmidt, H.A., Strimmer, K., Vingron, M., Haeseler, A.: TREE-PUZZLE: Maximum Likelihood Phylogenetic Analysis using Quartets and Parallel Computing. *Bioinformatics* **18** (2002) 502–504
4. Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J.C., Hutchison, C.: E-Cell: Software Environment for Whole Cell Simulation. *Bioinformatics* **15** (1999) 72–84
5. Takahashi, K., Yugi, K., Hashimoto, K., Yamada, Y., Pickett, C.F., Tomita, M.: Computational Challenges in Cell Simulation. *IEEE Intelligent Systems* **17** (2002) 64–71
6. Takahashi, K., Ishikawa, N., Sadamoto, Y., Sasamoto, H., Ohta, S., Shiozawa, A., Miyoshi, F., Naito, Y., Nakayama, Y., Tomita M.: E-Cell 2: Multi-platform E-Cell Simulation System. *Bioinformatics* **19** (2003) 1727–1729
7. Takahashi, K., Sakurada, T., Kaizu, K., Kitayama, T., Arjunan, S., Ishida, T., Berezcki, G., Ito, D., Sugimoto, M., Komori, T., Seiji, O., Tomita, M.: E-CELL System Version 3: A Software Platform for Integrative Computational Biology. *Genome Informatics* **14** (2003) 294–295
8. Arjunan, S., Takahashi, K., Tomita, M.: Shared-Memory Multiprocessing of Gillespie’s Stochastic Simulation Algorithm on E-Cell3. 4th International Conference on Systems Biology in St.Louis, MO (2003) poster 253
9. Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. *Int. J. Supercomput. Appl.* **11** (1997) 115–128
10. Rowe, A., Kalaitzopoulos, D., Osmond, M., Ghanem, M., Guo, Y.: The Discovery Net System for High Throughput Bioinformatics. *Bioinformatics* **19** (2003) 225–231
11. Oinn, T.M.: Talisman - Rapid Application Development for The Grid. *Bioinformatics* **19** (2003) 212–214
12. Stevens, R.D., Robinson, A.J., Goble, C.A.: *my*Grid: Personalized Bioinformatics of The Information Grid. *Bioinformatics* **19** (2003) 302–304
13. Konishi, F., Shiroto, Y., Umetcu, R., Konagaya, A.: Scalable BLAST Service in OBIGrid Environment. *Genome Informatics* **14** (2003) 535–536
14. Kimura, S., Kawasaki, T., Hatakeyama, M., Naka, T., Konishi, F., Konagaya, A.: OBIYagns: A Grid-based Biochemical Simulator with A Parameter Estimator. *Bioinformatics* **20** (2004) 1646–1648
15. Buyya, R., Abramson, D., Giddy, J.: Nimrod/G: An architecture for a Resource Management and Scheduling System in A Global Computational Grid. In *Proceedings of the HPC ASIA’2000, China* (2000)

16. Andrew, S., Ishikawa, N., Yamazaki, T., Fujita, A., Kaneko, I., Fukui, Y., Ebisuzaki, T.: Ecell2d : Distributed E-Cell2. *Genome Informatics* **14** (2003) 288–289
17. Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., Tomita, M.: Dynamic Modeling of Genetic Networks using Genetic Algorithm and S-system. *Bioinformatics* **19** (2003) 643–650
18. Hu, B., Tomita, M.: A Stochastic *E. coli* Heat Shock Model Using E-Cell v3. 4th International Conference on Systems Biology, in St. Louis, MO (2003)
19. Cantu-Paz, E.: A Survey of Parallel Genetic Algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis* **10** (1997) 141–171
20. Tomassini, M.: Parallel and Distributed Evolutionary Algorithms: A Review. John Wiley and Sons (1999) 113–133
21. Horii, H., Kunifuji, S., Matsuzawa, T.: Asynchronous Island Parallel GA Using Multiform Subpopulations. *Lecture Notes in Computer Science* (1999) 122–129
22. Voit, E.O.: Computational Analysis of Biochemical Systems. Cambridge University Press (2000)
23. Athel, C.B.: Fundamentals of Enzyme Kinetics. Portland Press (1995)
24. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press (1992)

The Architectural Design of High-Throughput BLAST Services on OBIGrid

Fumikazu Konishi and Akihiko Konagaya

Bioinformatics Group,RIKEN Genomic Sciences Center, 1-7-22,
Suehiro-cho, Tsurumi, Yokohama, Kanagawa, Japan
(fumikazu, konagaya)@gsc.riken.jp

Abstract. OBIGrid provides high-throughput GRIDBLAST services (OBIGBs) for researchers who need to deal with many BLAST query sequences at one time by exploiting both distributed processing and parallel processing. A new application-oriented grid framework has been introduced to split a BLAST query into independent sub-queries and to execute the sub-queries on remote personal computers and PC clusters connected by a virtual private network (VPN) over the Internet. The framework consists of five functional units: query splitter, job dispatcher, task manager, result collector and result formatter. They enable us to develop a cooperative GRIDBLAST system between a server and heterogeneous remote worker nodes: which consist of various computer architectures, different BLAST implementations and different Job schedulers operated by local resource management policy. The OBIGBs can execute 29,941 PSI-BLAST query sequences in 8.31 hours when using 230 CPUs in total and can return a 1.37 Giga byte result file.

1 Introduction

Any high-throughput "ome" project (genome, proteome, transcriptome, phenotype etc.) requires intensive homology search to known genomic sequences because genomic information is a basis of the central dogma in molecular biology. The increase of genomic information requires masses of computational power for high-throughput homology services to deal with thousands of query sequences at one time.

To realize high-throughput homology services, GRID computing is one of the most attractive and promising approaches currently available because we can create a virtual organization for the tasks which surpass the computational power owned by single organizations. However, the following issues have been raised when constructing network-transparent BLAST[1] services on grid:

1. Load balancing of sub-queries
2. Difference in local resource management policies
3. Variations of computer architectures, BLAST implementations, Job schedulers, etc.

Load balancing is one of the classical issues in parallel processing: It becomes more difficult in heterogeneous wide-area networks where large PC-clusters and conventional personal computers are connected on the Internet. Intelligent and robust task control mechanism is necessary to handle long network communication latency and unexpected CPU work overload as well as network troubles which are rare in local area networks but often happen in wide area networks.

Local resource management policy strongly depends on the design philosophy and objectives of a grid. OBIGrid[2] adopts free local resource management policy in which every site retains their rights to manage their resources by themselves. In this case, a server cannot expect full usage of remote worker nodes for a specific network service.

Variations of computer architectures make resource management more complicated. This is one of the typical differences between grids and PC-clusters. In the case of a PC-cluster, one can assume the same computer architecture, the same CPU performance, the same network performance, the same software version, etc. A grid requires an intelligent resource management system to deal with heterogeneous computing resources.

The OBIGrid BLAST services (OBIGbs) have solved the above issues by providing a grid framework and building a cooperative GRIDBLAST system between a server and heterogeneous remote worker nodes. The grid framework consists of five functional units: query splitter, job dispatcher, task manager, result collector and result formatter.

- The query splitter unit splits an input query into sub-queries in proportion to the power of the remote worker nodes for balancing sub-query processing time.
- The job dispatcher unit dispatches the sub-queries to the BLAST systems on worker nodes through gate keepers and job managers.
- The task manager unit monitors the execution of BLAST systems by sending “alive messages” to the worker nodes.
- The result collector unit gathers BLAST results from the worker nodes.
- The result formatter unit reforms results into a format specified by users when all the sub-queries have been processed.

This paper describes the design of the GRIDBLAST system and show its effectiveness by three performance measurements from the viewpoints of scalability in execution time and throughput performance. First, we describe the details of the GRIDBLAST architecture in section 2 and its implementation in section 3. Then, in section 4, we discuss the results of the GRIDBLAST performance measurements: (1) improvement in execution time when adding the number of CPUs for a fixed query size, (2) throughput performance when increasing the number of query sequences on a local area network, and (3) total throughput performance of a practical application on a wide area network over the Internet. Finally section 5 gives our conclusion of this paper.

2 GRIDBLAST Architecture

Design of BLAST services strongly depend on the purpose of services, target users, available resources, local service policies, etc. OBIGbs provides high-throughput GRIDBLAST services for researchers on the OBIGrid where various kinds of computers resources are connected on the Internet and are controlled under local management policy on each site. There are three major approaches in developing high-throughput GRIDBLAST services over the Internet.

1. Master-slave model (centralized control, centralized data)
2. Peer to Peer model (distributed control, distributed data)
3. Cooperative model (centralized control, distributed data)

A master-slave model consists of a centralized server with a database and stateless remote worker nodes. When a server receives a query, the server splits the query into independent sub-queries and sends a set of a sub-query and a database to a worker node at runtime. Task and database management is simple but handling of database deployment becomes one of the performance bottlenecks in this approach.

A peer to peer model is another way to realize the services. All nodes have a copy or some portion of a database. Any node in this model can split a query into sub-queries and asks other nodes to execute the sub-queries. A database is stored in each node in advance. The database can be deployed on demand but a complicated database management mechanism is required to keep data consistency. Task management becomes simpler if a resource manager is provided as a hybrid peer-to-peer model.

A cooperative model is a mixture of a master-slave model and a peer-to-peer model. It consists of a centralized server and worker nodes which have a copy of the same database. Task control is simple and a database can be deployed in advance or on demand as with a peer to peer model.

OBIGbs adopts a cooperative model for the following reasons. First, we need to avoid runtime overhead of database deployment to make use of the Internet. High-performance network and database caching technology may reduce the overhead. However, runtime deployment of DNA sequence databases would be a good challenge for data grid technologies if considered that the databases become double in every year.

Second, a cooperative model can make use of parallel BLAST systems running on PC-clusters. The parallel BLAST system remarkably reduces BLAST search time by exploiting data parallelism, that is, database search in parallel. The system fits to a cooperative model but needs some elaborated works for handling data parallelism in a master slave model and a peer to peer model.

Third, a cooperative model is flexible enough to make use of various grid middleware such as job schedulers and data management tools. In OBIGbs, we have integrated different computer architectures, different BLAST implementations and different schedulers located at RIKEN GSC, Tokyo Institute of Technology and NEC Corporation to prove the feasibility of our approach.

Open Bioinformatics Environment[3] (OBIEEnv) adopted a hybrid peer-to-peer model as a basis in this issue. This is because OBIEEnv aims to develop a distributed bioinformatics environment among many sites rather than providing a bio application portal like OBIGbs. Although OBIEEnv and OBIGbs have adopted different distributed processing models in terms of BLAST services, each approach has its own strength: node scalability in OBIEEnv and large database handling in OBIGbs. These two approaches will be integrated in the near future by providing web service interface on OBIGbs.

3 Implementation

GRIDBLAST has been developed on a framework which consists of five functional units executed in sequentially (Fig. 1); query splitter (QS), job dispatcher (JD), task manager (TM), result collector (RC) and a result formatter (RF).

3.1 Query Splitter (QS)

The QS unit splits an input query into independent sub-queries in proportion to the ability of the remote computer systems for balancing sub-query processing time as equally as possible. Sub-query sizes are determined by capacity score which represents relative appraisal of remote computer systems in terms of network performance and available computing power. The capacity score is estimated by the throughput performance measured from the execution of benchmark programs on all resources. A calibration table is also provided to adjust the capacity score for input sequences, search programs and search parameters. All parameters in GRIDBLAST are tuned for large-scale homology search against large-scale databases effectively.

3.2 Job Dispatcher (JD)

The JD unit dispatches sub-queries to the remote BLAST systems through JAVA CoG Kit[4], gate keepers and job managers in Globus Tool Kit[5]. Firstly, the JD unit checks the availability of remote sites for an input query to the gatekeepers on the worker nodes. The gatekeepers return the number of processors available by checking a Globus account of an input query for authentication and resource allocation. Site administrators have rights to determine resource allocation rules according to their own resource management policies. The JD unit has a master configuration file for a virtual site which consists of machine configuration, applications, databases, a gatekeeper and a capacity score. The master file plays an essential role in mapping a virtual site to a physical site. This mechanism enables a site administrator to provide different virtual site configurations for GRIDBLAST on the same hardware: four 16-node virtual PC clusters or two 32-node virtual PC clusters on one 64-node physical PC cluster.

3.3 Task Manager (TM)

The TM unit monitors the execution of remote BLAST systems by sending “alive messages” to worker nodes. The TM unit grasps remote job status by Globus Resource Allocation Manager (GRAM) provided by the Globus Toolkit[5]. The TM unit monitors remote BLAST execution status in five stages; INITIALIZE, PENDING, ACTIVE, FAILED and DONE. INITIALIZE indicates that a remote BLAST system is in the preparation stage. PENDING indicates that a remote worker has accepted a query but not yet started a BLAST search. ACTIVE indicates a remote worker is executing a BLAST search. The status becomes FAILED if a search fails due to some reason. It becomes DONE when a search has been completed successfully. GRIDBLAST have fault tolerance facilities to detect the failure of a sub-query execution on a remote worker and retry the sub-query on another worker node.

3.4 Result Collector (RC)

The RC unit gathers BLAST results from remote worker nodes when all sub-queries have been processed. The results are asynchronously transferred from the remote nodes by globus-url-copy.

3.5 Result Formatter (RF)

The RF unit reforms BLAST results into a format specified by a user. It is guaranteed to return a BLAST result in the format of a conventional sequential BLAST returns.

The above five functional units enable us to built GRIDBLAST services (OBIGbs) for large scale BLAST queries on distributed computers connected by a VPN on the Internet. OBIGbs also prepares a web interface as a front end of BLAST search. A user can receive a URL of BLAST search result by e-mail. The system architecture is so flexible that it can make use of various kinds of computer architectures, BLAST implementations and job schedulers.

4 Performance Evaluation

We evaluated the performance of GRIDBLAST from the viewpoints of scalability in execution time and throughput performance. Scalability tests measure increased speed ratio in proportion to the number of worker sites in order to evaluate the effectiveness of a load balancing algorithm and to estimate maximum performance of our system. Early evaluation results were obtained from twelve uniform virtual worker sites constructed on two large PC clusters on a local area network. As for the throughput performance, we measured the number of query sequences processed per hour when changing the number of queries sent to virtual worker sites on a local area network, and the one when sent to real worker sites on a wide area network over the Internet.

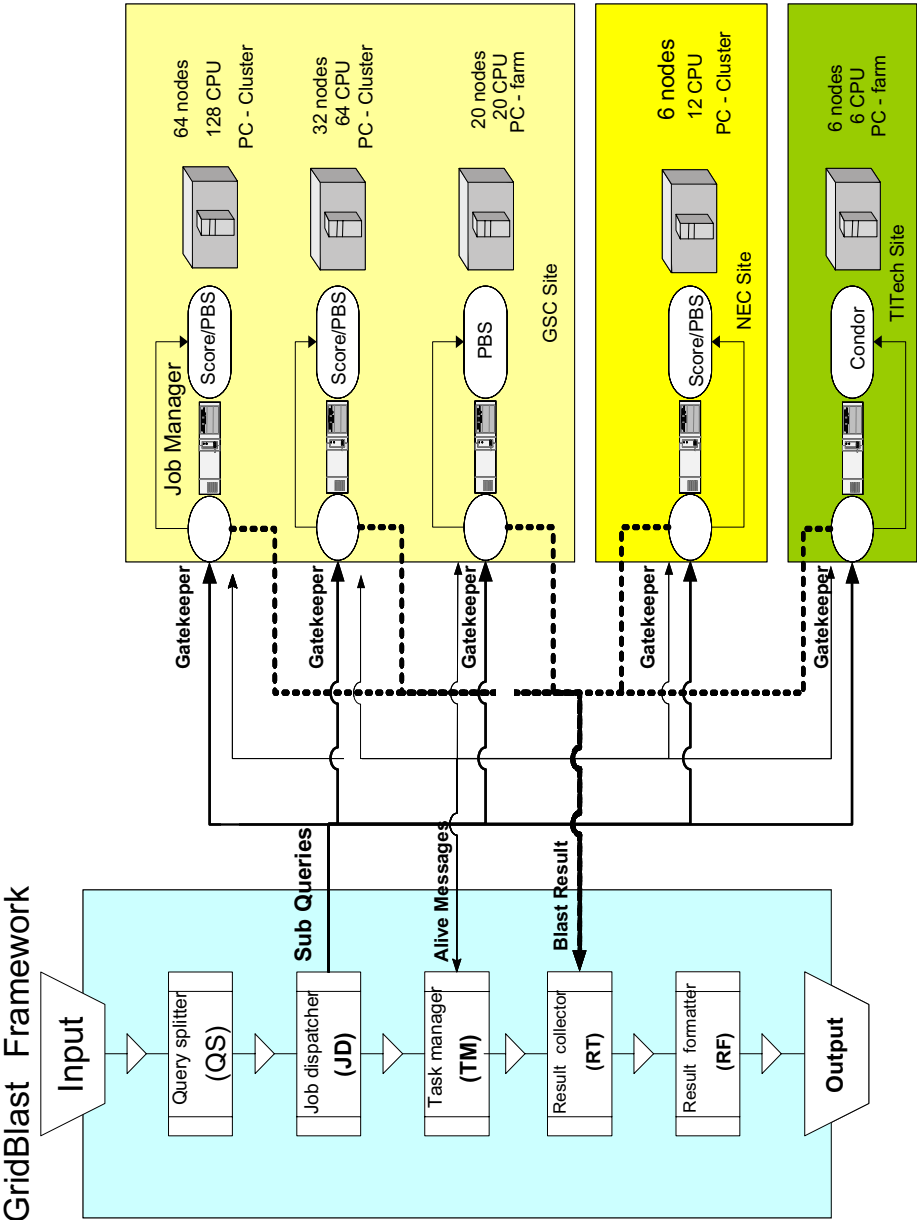


Fig. 1. Diagram of GRIDBLAST Framework

4.1 Overhead Measurement

To measure the overhead of the OBIGbs framework, we developed a real time monitor to check job status accumulation on each site (Fig. 2). A job status ac-

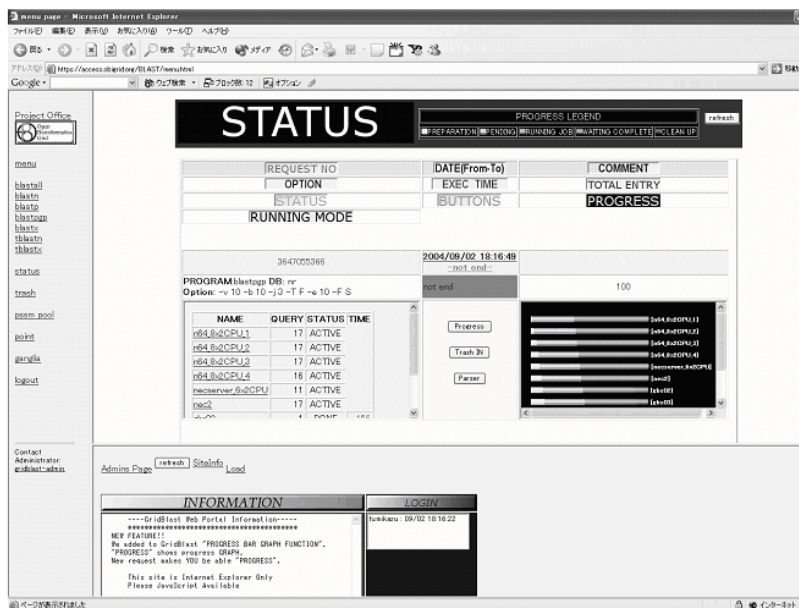


Fig. 2. Screen image of progress monitor for BLAST execution on each site

cumulation is represented by a bar painted in five colors corresponding to time stages obtained from GRAM: preparation, pending, running job, waiting complete and clean up. According to our early measurement, Grid execution overhead decreased when executing a large-scale homology search. However, result data transfer became a bottleneck and severe overhead was found in gsift. Research in multi-site data transfer still has untouched technical issues to explore.

4.2 Scalability in Execution Time

To evaluate the scalability of GRIDBLAST in execution time, we constructed a test bed which consists of twelve virtual worker sites on two PC clusters: 32 nodes and 64 nodes Linux PC-clusters with dual PentiumIII 933MHz CPUs, 1G Bytes main memory and one SCSI 36G disk on each node. The nodes are all connected by Gigabit Ethernet switches. We have developed twelve virtual worker sites, each of which provides a parallel BLAST system with 16 CPUs.

Table 1 shows the BLAST parameters used in this experiment. The target database is all non-redundant protein sequences (nr) constructed from GenBank CDS translation, protein structure database (PDS), protein database (SwissProt), protein identification resources (PIR) and Protein Research Foundation database (PRF). The nr database had 1,794,787 entries and 592,464,797 residues when this measurement was done. Gapped Blast search was applied for an input query sequence with 278 residues, retrieved from *Mycoplasma pneumoniae* peptide sequence (AAB96231).

Table 1. GRIDBLAST Parameters for Scalability Measurement

App	blastpgp
DB	nr (All non-redundant GenBank CDS translations + PDB + SwissProt + PIR + PRF)
Options	-v 500 -b 500 -j 3 -T F -e 10 -F L
Query	AAB96231 Mycoplasma pneumoniae peptide sequence (278 residues)

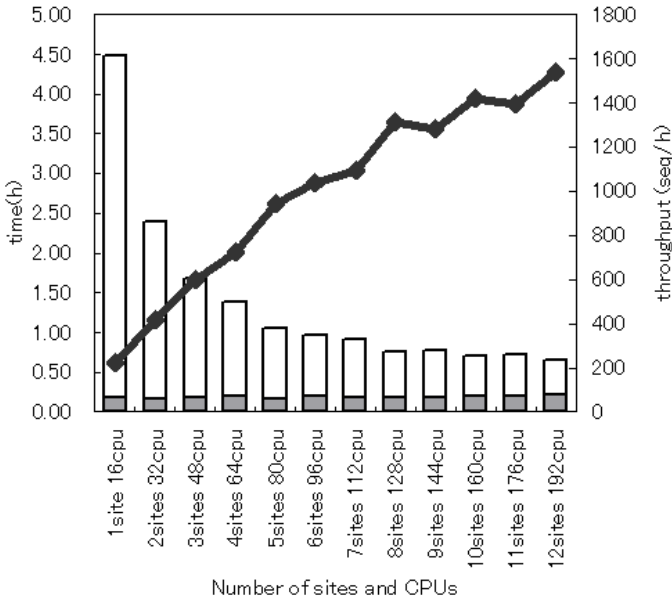
**Fig. 3.** Scalability Measurement on Twelve Virtual Worker Sites

Fig. 3 shows the speedup ratio when increasing the number of virtual sites while using the same database and the same input sequence. It took 4.48 hours for one virtual site (16CPUs) and 0.65 hours for twelve virtual sites (192 CPUs) to search the database. Speed up ratio in execution time is about 6.9 for 12 fold of CPUs. Communication latency was negligible in this measurement but some constant overhead caused by Globus Tool Kit were observed. Other reasons that prevent GRIDBLAST scalability are still under investigation.

4.3 Throughput Performance on Local Area Network

The purpose of OBIGBs is to provide high throughput GRIDBLAST services for multi users. In this case, it is important to find optimal job granularity that can fully make use of a GRIDBLAST system. Otherwise, over problem-decomposition may cause a waste of resources or may reduce the throughput performance. In order to measure the GRIDBLAST throughput performance,

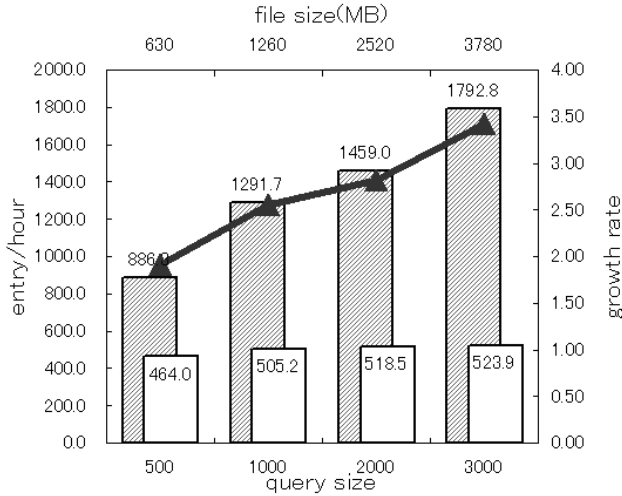


Fig. 4. Throughput Performance on Local Area Network

we have developed two test beds: a small GRIDBLAST system with 34 CPUs (SGS) and a large GRIDBLAST system with 224 CPUs (LGS). SGS consists of 34 Celeron 1.3GHz single CPU with 1Giga Bytes main memory and one 36G Byte IDE disk. LGS consists of SGS and two PC Clusters used for scalability measurement.

Fig. 4 shows the difference of throughput performance between SGS and LGS when changing the number of query sequences. We used the same BLAST parameters in Table 1 and made an input query by duplicating the same sequence to equalize the BLAST search time per sequence. Fig. 4 shows that throughput performance ratio between SGS and LGS is 1.91 for 500 sequences and 3.42 for 3000 sequences. Although it is too early to derive a concrete statement from this measurement, the result indicates that LGS cannot make use their full capacity when the number of input query sequences is less than 3000. The ratio will be worse when PC-clusters are distributed in a wide area network. It should be also noted that a result file becomes 3.78 Giga Bytes for a query with 3000 sequences. Multistage data transfer is mandatory to deal with BLAST results.

4.4 Throughput Performance on Wide Area Network

The GRIDBLAST throughput performance strongly depends on communication overhead in wide area networks. In addition, robustness and flexibility are also required to dealing with various computer resources on the remote sites. The current OBIGRID BLAST services (OBIGbs) consist of three GRIDBLAST sites located in RIKEN Genomic Sciences Center (GSC), Tokyo Institute of Technology (TITech) and NEC Corporation (NEC) connected by a VPN over the Internet. The uniqueness of OBIGbs is in its heterogeneous configuration in computer architectures, BLAST implementations and job schedulers. The GSC

Table 2. GRIDBLAST Parameters for Throughput Measurement on Wide Area Network

App	blastpgp
DB	nr (All non-redundant GenBank CDS translations + PDB + SwissProt + PIR + PRF)
Options	-v 10 -b 10 -j 3 -T F -e 0.0001 -F L
Query	FANTOM 2.1 predicted amino-acid sequences[8] (12 Mega Bytes)

GRIDBLAST site consists of twelve MPI-based parallel BLAST systems running on 64CPUs and 128CPUs PC-clusters scheduled by SCORE/PBS[6] and twenty NCBI BLAST systems running on twenty single-CPU personal computers scheduled by PBS. The TITech GRIDBLAST site consists of six NCBI BLAST systems running on six single-CPU personal computers scheduled by CONDOR[7]. The NEC GRIDBLAST site consists of twelve NCBI BLAST systems running on a 12-cpu PC cluster scheduled by SCORE/PBS.

The OBIGRID BLAST services (OBIGbs) can execute 29,941 BLAST query sequences in 8.31 hours when using 230 CPUs in total and can return 1.37 Giga Bytes BLAST results for the BLAST parameters in table 2.

5 Conclusion

We have designed and developed a prototype of a high throughput GRIDBLAST services (OBIGbs) on Open Bioinformatics Grid (OBIGRID). The GRIDBLAST system gives network transparent BLAST services by encapsulating architectural differences while pursuing maximum performance of local computer resources by tuning BLAST implementations and job schedulers suitable to computer architectures on each site. Performance measurements have been made from the viewpoints of scalability in execution time and throughput performance using virtual remote sites on a local area network and the real remote sites on the Internet. The performance measurements show the effectiveness of the GRIDBLAST architecture in scalability with respect to the number of remote sites and the limitation of query decomposition when a query size is small. More performance measurements and evaluation are necessary to derive concrete results useful for parallel processing and distributed processing. In order to show the feasibility of the GRIDBLAST architecture, we have developed a GRIDBLAST system on OBIGRID. The GRIDBLAST services are now available at our web portal site (<http://www.obigrid.org/>). It can execute 29,941 BLAST query sequences in 8.31 hours and can return 1.37 Giga Bytes BLAST results when using 230 CPUs in total on the Internet.

Acknowledgements

We gratefully acknowledge the members of OBIGrid project for valuable discussion and grid operation, and Shiroto Yukimasa and Yoshitomi Yuuki in NEC In-

formatech Systems, Ltd. for their contribution to the development of the GRID-BLAST systems.

References

1. Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman : Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Res.* vol. 25, pp.3389–3402 (1997).
2. Akihiko Konagaya, Fumikazu Konishi, Mariko Hatakeyama: The Superstructure Toward Open Bioinformatics Grid, *New Generation Computing*, vol.22, no.2, (in printing) (2004).
3. Kenji Satou, Yasuhiko Nakashima, Shin'ichi Tsuji, Xavier Defago, Akihiko Konagaya: An Integrated System for Distributed Bioinformatics Environment on Grids, in *Proc. of Int. workshop on Life Science Grid (LSGRID2004)* (2004).
4. Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane : A Java Commodity Grid Kit, *Concurrency and Computation: Practice and Experience*, Volume 13, Issue 8-9, pp. 643–662 (2001).
5. I. Foster, C. Kesselman, J. Nick, S. Tuecke: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration., *Open Grid Service Infrastructure WG, Global Grid Forum*, June 22 (2002).
6. Yutaka Ishikawa, Hiroshi Tezuka, Atsuhori Hori, Shinji Sumimoto, Toshiyuki Takahashi, Francis O'Carroll, and Hiroshi Harada : RWC PC Cluster II and SCore Cluster System Software — High Performance Linux Cluster. In *Proceedings of the 5th Annual Linux Expo*, pp.55–62 (1999).
7. Jim Basney and Miron Livny: Deploying a High Throughput Computing Cluster, *High Performance Cluster Computing*, Rajkumar Buyya, Editor, Vol. 1, Chapter 5, Prentice Hall PTR, May (1999).
8. The RIKEN Genome Exploration Research Group Phase II Team and the FANTOM Consortium, Functional annotation of a full-length mouse cDNA collection, *Nature*, vol. 409, pp.685–690 (2001).

Heterogeneous Database Federation Using Grid Technology for Drug Discovery Process

Yukako Tohsato¹, Takahiro Kosaka¹, Susumu Date¹,
Shinji Shimojo², and Hideo Matsuda¹

¹ Graduate School of Information Science and Technology, Osaka University,
1-3 Machikaneyama, Toyonaka, Osaka 560-8531 Japan

{yukako, tak-k, sdate, matsuda}@ist.osaka-u.ac.jp

² Cybermedia Center, Osaka University,
5-1 Mihogaoka, Ibaraki, Osaka 567-0047 Japan
shimojo@cmc.osaka-u.ac.jp

Abstract. The rapid progress of biotechnology provides an increasing number of life science databases. These databases have been operated and managed individually on the Internet. Under such a circumstance, it is needed to develop an infrastructure that allows to share information contained in these databases and to conduct research collaboration. Grid technology is an emerging technology for seamless and loose integration of diverse resources distributed on the Internet. In order to achieve federation of the heterogeneous databases, we have developed a system for supporting a drug discovery process using Globus Toolkit3/OGSA-DAI. As an essential part of the system, we introduce a protein-compound interaction search based on a meta-data bridging protein and compound information with their interaction types; such as, inhibitor, agonist, antagonist, etc. The effectiveness of our system is demonstrated by searching for the candidate compounds interacting with the glucocorticoid receptor protein.

1 Introduction

Recently, life science research has progressed rapidly. Acquired knowledge is accumulated in hundreds of bio-related databases. On the other hand, a large number of sophisticated analytic software tools have been developed. Scientists expect an integration-understanding of life activity using these life scientific resources. Bio-related databases have an enormous amount of data. Types of the data have been diverged rapidly, and the databases are distributed on web. User needs a method of effective utilization for the databases. Thus, database integration technology has been studied. Physical integration of the databases is not feasible since the cost to unify data models of the databases is very expensive due to frequent update of the databases. For this reason, we took an approach of distributed query based on metadata describing relationships of databases[1].

We focus on grid technology as one of the most promising technologies. Open Grid Services Architecture (OGSA[2]) enhances web service technology with advanced functions such as state management. The utilization of these functions standardized by Global Grid Forum (GGF)[3] makes it possible to realize efficiently our approach for integrating heterogeneous databases.

In this paper, we introduce a protein-compound interaction search in the heterogeneous database federation using OGSA in our prototype system. The effectiveness of the search is demonstrated by applying the method to searching for drug candidates which relate to the glucocorticoid receptor protein.

2 Approach for Heterogeneous Database Federation

2.1 Metadata-Based Database Federation

Life science researches often require integration of interdisciplinary information extracted from domain-specific databases. For example, drug discovery process is composed of many stages: target identification/validation, lead identification/optimization and clinical trials. Each stage needs different knowledge and information from a wide variety of data resources, such as medical, gene, protein and compound databases. It is, however, very difficult to interoperate these databases due to large semantic gaps among their background knowledge (medical science, molecular biology and pharmaceuticals). To cope with this issue, we have introduced two types of metadata, application metadata (AP-Metadata) and data service metadata (DS-Metadata), for bridging the information among databases. AP-Metadata plays a role to fill the gaps between applications (e.g., drug discovery) and databases, whereas DS-Metadata provides a unified view of the databases that are expressed in different formats but are based on the same background.

Fig. 1 shows the relationship between AP-Metadata and DS-Metadata for an example of protein-compound interaction search services. AP-Metadata describes relationships between proteins and compounds. When databases include the same type of data such as protein-related databases, DS-Metadata links these databases such as SWISS-PROT[4], PIR[5] and PDB[6]. Here, we design a standard data format to resolve their heterogeneity. We assign a unique identifier (id) to each entry in these databases using the standard data format. AP-Metadata and DS-Metadata keep reference pointers to the original databases. The reference pointers include their database names and their database IDs for database entries.

2.2 Open Grid Services Architecture

We use the grid technology as one of the most promising technologies that enable us to efficiently integrate heterogeneous resources for life science. Recently GGF has proposed OGSA. OGSA has prescribed uniform grid service interfaces as an extension of traditional web services with new functions such as state management and life cycle management. Every service is expected to have an

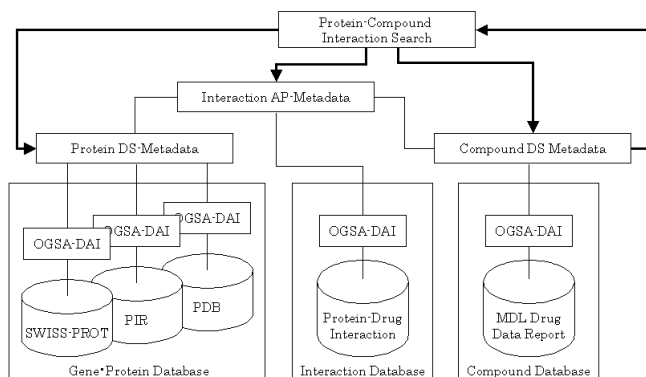


Fig. 1. Outline of our prototype system

interface that is described with XML, exchange messages in an XML format via Internet-based protocol (e.g., SOAP).

As a tool which is constructed by OGSA, OGSA-DAI (Open Grid Service Architecture Data Access and Integration)[7] has been developed in the e-Science project[8]. OGSA-DAI is a set of grid services that enables us to make various data resources accessible as grid services. It will support DB2, Oracle, MySQL and XIndice. By using OGSA-DAI, the database will be integrated virtually using web mechanisms such as SOAP to enable database services to operate within the XML scheme. The architecture was proposed in detail in our previous papers[1][9]. In our system, the grid services are integrated by using Globus Toolkit 3 with OGSA-DAI (see Fig. 1).

3 An Application Example for Drug Discovery

3.1 Protein-Compound Interaction Search

Our prototype system has been built with a focus on an actual use case in drug discovery. Drug discovery is a process for finding chemical compounds (drugs) that have effects on their target proteins. The candidate compounds must be refined and tested carefully for their safety and efficacy, usually resulting in its discard (only 1 in 5,000 compounds tested reaches the market). In this step, first, users must select a known protein involved in a disease. Next, users have to search for candidate compounds that can alter the action of the target protein. If we can use the known data for systematically searching for the compound, we minimize the cost of drug development.

Schuffenhauer and coworkers introduced a compound-protein interaction database[10]. The database covers compounds with hierarchical levels of protein classification which is based on pharmacological activity. The classification used for a protein-compound interaction search (see Fig. 2)[11]. An advantage of the

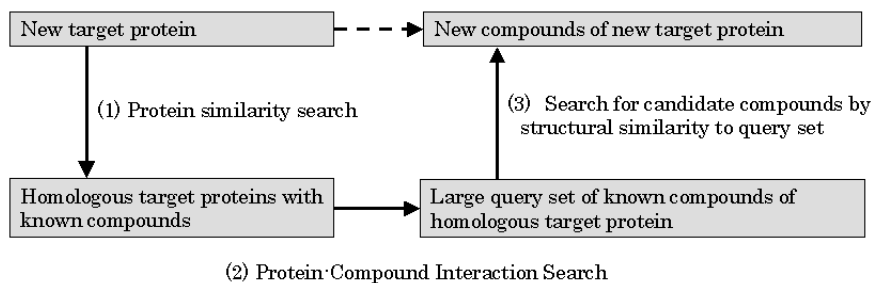


Fig. 2. Work flow of protein-compound interaction search

method is that it is able to find the candidate compounds for target protein that has unknown biological function. The searching process is as follows.

1. Search for homologous proteins of a target protein from protein databases using a specialized tool such as NCBI-BLAST[12].
2. Retrieve compounds bound to the homologous proteins from compound databases and interaction databases.
3. Search for drug candidates from the retrieved compounds based on structural similarity.

Fig. 3 shows an example of the correlation between the sequence similarity of proteins and the structural similarity of compounds bound to the proteins. Rosiglitazone is a compound bound to the peroxisome proliferator-activated re-

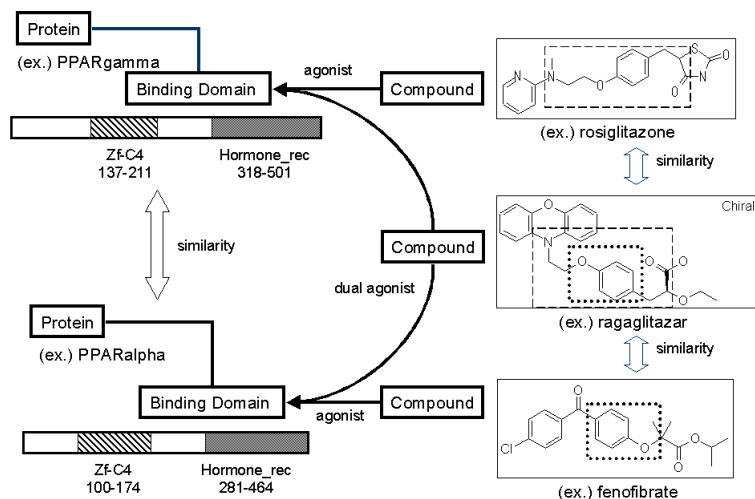


Fig. 3. An example of the protein-compound relationship with protein-protein and compound-compound similarity

ceptor gamma (PPAR-gamma) protein. Fenofibrate is a compound bound to the peroxisome proliferator-activated receptor alpha (PPAR-alpha) protein. The PPAR-gamma protein and the PPAR-alpha protein belong to the same family, and each sequence has high sequence similarity. The structures of their compounds are similar to each other. We introduce the protein-compound interaction search to heterogeneous database federation.

3.2 Compound Representation and Similarity Measure

Since structure searches are typical NP problems, they are computationally costly[13]. Thus in order to enhance the computation performance, pre-defined substructures are used to search for a compound[14]. Such substructures are called descriptors. Using pre-defined N descriptors, a structure of a compound is represented by a bit-string (a sequence of “0” and “1” digits). When a bit is set to 1 in a bit-string, it means the corresponding substructure is present, and 0 means the substructure is absent. For example, MDL[15] represents the 166 descriptors in the ISIS structure database systems. We used the ISIS public key (166 descriptors) to express the compound structure.

Many different similarity calculations between two bit-strings have been reported [10] [16]. For example, the Tanimoto coefficient $t(x, y)$, takes the form

$$t(x, y) = \frac{\gamma}{\alpha + \beta - \gamma} \quad (1)$$

where α and β are the number of 1 bits (i.e., the number of substructures) in bit-strings x and y , respectively, and γ is the number of common substructures in both bit-strings x and y . The Tanimoto coefficient has value 1 representing that the two compounds have the identical set of substructures, and has value 0 representing that they do not share any substructures. Xue and coworkers indicated that, a high Tanimoto score of 0.7 or above is representative of two compounds having high structural similarity which is a good indication of similar biological activity[17]. Willet and Winterman showed the Tanimoto coefficient is the best performance compared to the Euclidean distance, cosine and Dice coefficients for similarity searching in structural databases and is implemented in most of the publicly available structure similarity searching tools. As similarity function for a pair of compound structures we used the Tanimoto coefficient in this study.

For the protein-compound interaction search in Sect. 3.1, the structural similarity search in step 3 needs to find compounds similar to a set of query compounds (the retrieved compounds in step 2) instead of a single query compound. To use a set of compounds as a query for similarity searching, a method to calculate the similarity of a candidate compound to a query set of N compounds needs to be developed. We use the *nearest neighbor* method for the search with a large set of compounds as a query. Hereafter we refer to a reference set as a set of query compounds, and refer to a candidate set as a set of compounds to be searched by the method.

Let r_i be the representation vector of compound i in the reference set R . Let x be the representation vector of the candidate compound. The similarity between two compounds x and y is $t(x, y)$. The nearest neighbor method takes a similarity score $Min(x, R)$, given by Equation(2). Here the similarity between compound set R and compound x is defined as similarity between x and its nearest neighbor in R .

$$Min(x, R) = \min \{t(x, r_i) | r_i \in R\}. \quad (2)$$

In our prototype system, both reference and candidate sets are included in the same database. Currently the reference set is selected manually based on user interest. The candidate set is obtained by removing the reference set from the database. All compounds in the candidate set were ranked by their similarity.

4 Implementation

We apply the protein-compound interaction search to our prototype system with the following configuration:

- OS: Redhat Linux 9
- CPU: Pentium4 2.4 GHz
- Memory Size: 4GB
- Java: Java SDK 1.4.1_03_b02
- Globus Toolkit: Globus Toolkit 3.0.2
- OGSA-DAI: Release 2.5
- Container: Jakarta Tomcat 4.1.24
- DBMS: MySQL 3.23.54

We aim for efficient integration by categorization and aggregation of databases based on their types. In our prototype system, 11 bio-related databases are used, which are categorized by disease, genome, protein, compound and interaction as shown in Table 1. The program and data for protein-compound interaction searching are assigned on one computer. Querying databases for each category are provided as our grid service.

Table 1. Databases we used in our prototype system

Type	Database	Size
Disease	Medeical Encyclopedia	3079 entries
Genome	DDBJ	Human 7037852 entries, 10176023644 bases
		Mouse 5063486 entries, 6071844270 bases
Protein	Swiss-Prot	137885 entries, 50735179 amino acids
	PIR	283227 entries, 96134583 amino acids
	PDB	23073 entries
Compound	MDL Drug Data Report (MDDR-3D)	142553 entries
Interaction	ENZYME, GPCR-DB	
	NucleaRDB, LGIC-DB	

4.1 Databases

For the protein-compound interaction search, we use Swiss-Prot Release 39.17 of 27-Apr-2001 for a protein database and MDL Drug Data Report (MDDR) Release 2003.2 for a compound database. Swiss-Prot includes 137885 proteins. MDDR includes 142553 compounds. In this paper, proteins are denoted by their Swiss-Prot accession numbers (e.g., P014050) and compounds are denoted by their MDDR registry id (e.g., 209035). We bridge proteins and compounds to use the NucleaRDB relational database for a protein-compound interaction database which is annotated by protein classification[10]. For a protein similarity searching, we use NCBI-BLAST Version 2.2.6.

Each of these databases are accessible through OGSA-DAI. OGSA-DAI receives a SQL statement in an XML document from the metadata service and queries to the underlying database using JDBC. Results of the query are translated into an XML document and transferred back to the metadata service. The databases have been stored into a MySQL server.

Many redundant compounds exist in the MDDR database. For example, it is shown that the compound of MDDR registry ids 209035 and 209040 are the derivatives of 207704. Thus, we use PREF.NUMBER for identifying derivatives of compounds. PREF.NUMBER contains the Prous Entry Number of a compound, in a series of derivative compounds, which reportedly exhibits the greatest biological activity. We select the compound which has the same value in both the MDDR registry id and PREF.NUMBER fields. This indicated that it has the greatest biological activity or is the representative compound in the derivative compounds.

4.2 Search Results

We apply a protein-compound interaction search for the compounds having activity to the glucocorticoid receptor protein (Swiss-Prot accession number is P01450). When we search the homologous proteins for P01450 protein in the Swiss-Prot database, we use NCBI-BLAST. We select P08235, P06401 and P10275 in order of BLAST E-value which have similarity to the receptor protein. 163 compounds are retrieved as the compounds having activity to these proteins from the interaction database. We select 5 compounds (115029, 170262, 315962, 322129 and 329279) from the compounds that interaction types are agonist. When we search for similar structures from the MDDR database on the condition that the Tanimoto coefficient threshold is 0.9, we get 26 compounds. The processing time of the query is 7 seconds.

To evaluate processing times by using OGSA-DAI in the prototype system, we select the top similar proteins for the P01450 protein in order of BLAST E-value using NCBI-BLAST. Here, BLAST E-value threshold is 6E-39. We search for the compounds interacting with different numbers of proteins. The result is shown in Fig. 4(a)(b). Fig. 4(a) shows the average processing time of the three times of protein-compound interaction search compared to the number

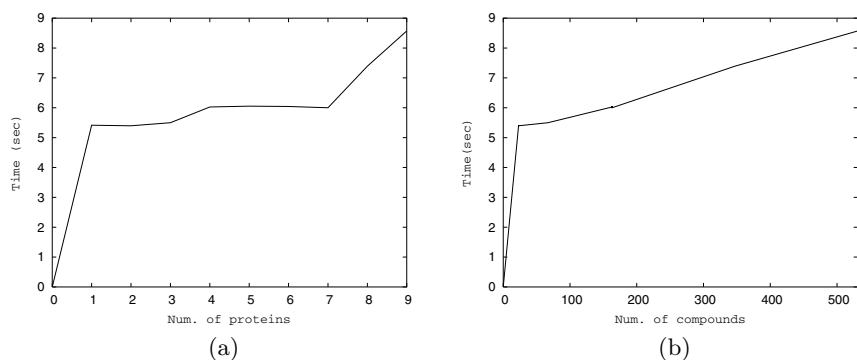


Fig. 4. Processing time of the protein-compound interaction search using OGSA-DAI

of proteins. Fig. 4(b) shows the average processing time of the three times of protein-compound interaction search compared to the number of the compounds. For example, the average processing time of the search for 531 compounds bound to 9 proteins is 8.5 seconds. In Fig. 4, the processing time increases linearly relative to the number of compounds.

5 Conclusions and Future Works

We have introduced a protein-compound interaction search in heterogeneous database federation with Globus Toolkit 3 and OGSA-DAI based on the consideration of the identification process of candidate compounds for a target protein. The effectiveness of our implementation is demonstrated by applying the method to the glucocorticoid receptor protein.

In the current implementation, we store the data into the relational database. We need to introduce an XML native database for expressing hierarchical structure of the bio-related data. In the drug discovery, users need security for the system. Thus, for further improvement, we plan to extend our system.

From December 2003, the UniProt database, which is a single, centralized, authoritative resource for protein sequences and functional information, has started [18]. The Swiss-Prot, TrEMBL and PIR protein database activities have united to form the Universal Protein Knowledgebase (UniProt) consortium. Its approach is close to ours since it assigns a unique id to each entry in protein databases and provides various services for biological research.

In this system, we need to set a user-defined threshold for limiting the number of results. In the future, this operation needs to be automated. In addition, although we used the ISIS public key as a compound representation, we need compound descriptors for the efficient substructure search in order to recognize of the characteristic substructure patterns.

Acknowledgments

This study was performed through IT-program of Ministry of Education, Culture, Sports, Science and Technology. The authors thank to the Biogrid project members.

References

1. Nakamura, H., Date, S., Matsuda, H., Shimojo, S.: A Challenge towards Next-Generation Research Infrastructure for Advanced Life Science. *New Generation Computing* **22** (2004) 157–166
2. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum (2002) <http://www.ggf.org/>
3. Global Grid Forum, <http://www.gridforum.org/>
4. Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.C., Estreicher, A., Gasteiger, E., Martin, M.J., Michoud, K., O'Donovan, C., Phan, I., Pilboud, S., Schneider M.: The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research* **31** (2003) 365–370
5. Wu, C.H., Yeh, L.S., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Kourtesis, P., Ledley, R.S., Suzek, B.E., Vinayaka, C.R., Zhang, J., Barker, W.C.: The Protein Information Resource. *Nucleic Acids Research* **31** (2003) 345–347
6. Bourne, P.E., Address, K.J., Bluhm, W.F., Chen, L., Deshpande, N., Feng, Z., Fleri, W., Green, R., Merino-Ott, J.C., Townsend-Merino, W., Weissig, H., Westbrook, J., Berman, H.M.: The distribution and query systems of the RCSB Protein Data Bank. *Nucleic Acids Research* **32** (Database issue) (2004) D223–D225
7. OGSA-DAI Project, <http://www.ogsadai.org>
8. UK e-Science (GRID) core programme, <http://www.escience-grid.org.uk/>
9. Kosaka, T., Tohsato, Y., Date, S., Hatsuda, H., Shimojo, S.: An OGSA-Based Integration of Life Scientific Resources toward Drug Discovery Proc. of HealthGRID 2004, Clermont-Ferrand (2004 in press)
10. Schuffenhauer, A., Zimmermann, J., Stoop, R., van der Vyver, J.J., Lecchini, S., Jacoby, E.: An ontology for pharmaceutical ligands and its application for in silico screening and library design. *Journal of Information and Computer Sciences* **42** (2002) 947–55
11. Schuffenhauer, A., Floersheim, P., Acklin, P., Jacoby, E.: Similarity metrics for ligands reflecting the similarity of the target proteins. *Journal of Information and Computer Sciences* **43** (2003) 391–405
12. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25** (1997) 3389–3402
13. Xu, J.: GMA: A Generic Match Algorithm for structural Homomorphism, Isomorphism, Maximal Common Substructure Match and Its Applications. *Journal of Chemical Information and Computer Sciences* **36** (1996) 25–34.
14. Flower, D.R.: On the Properties of Bit String-Based Measures of Chemical Similarity. *Journal of Chemical Information and Computer Sciences* **38** (1998) 379–386
15. MDL Drug Data Report Version 2003.2, MDL ISIS/HOST software, MDL Information Systems, Inc. San Leandro, CA, <http://www.mdl.com>

16. Brown, R.D., Martin, Y.C.: Use of Structure-Activity Data To Compare Structure-Based Clustering Methods and Descriptors for Use in Compound Selection. *Journal of Information and Computer Sciences* **36** (1996) 572–584
17. Xue, L., Godden, J.W., Bajorath, J.: Database Searching for Compounds with Similar Biological Activity Using Short Binary Bit String Representations of Molecules. *Journal of Chemical Information and Computer Sciences* **39** (1996) 881–886
18. Apweiler, R., Bairoch, A., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O'Donovan, C., Redaschi, N., Yeh, L.S.: UniProt: the Universal Protein Knowledgebase. *Nucleic Acids Research* **32** (Database issue) (2004) D115–D119

Grid Portal Interface for Interactive Use and Monitoring of High-Throughput Proteome Annotation

Atif Shahab¹, Danny Chuon¹, Toyotaro Suzumua², Wilfred W. Li⁵,
Robert W. Byrnes⁵, Kouji Tanaka^{3,4}, Larry Ang¹, Satoshi Matsuoka^{2,3},
Philip E. Bourne^{5,6}, Mark A. Miller⁵, and Peter W. Arzberger^{7,*}

¹ Bioinformatics Institute, 30 Biopolis Street,
#07-01, Matrix, Singapore 138671

{atif, dannyc, larry}@bii.a-star.edu.sg

² Dept. of Mathematical and Computing Sciences

³ Global Scientific Information and Computing Center

⁴ Tokyo Institute of Technology,

2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
Research and Development for Applied Advanced

⁵ Computational Science and Technology,

Japan Science and Technology Agency,

4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan

{tanaka, suzumura, matsu}@is.titech.ac.jp

Integrative Biosciences Program

San Diego Supercomputer Center

⁶ Department of Pharmacology

⁷ Life Sciences Initiatives,

University of California, San Diego

9500 Gilman Drive, La Jolla, CA 92093, USA

{wilfred, rbyrnes, bourne, mmiller}@sdsc.edu,
parzberger@ucsd.edu

Abstract. High-throughput proteome annotation refers to the activity of extracting information from all proteins in a particular organism using bioinformatics software on a high performance computing platform such as the grid. The Encyclopedia of Life (EOL) project [1] aims to catalog all proteins in all species for public benefits using an Integrative Genome Annotation Pipeline [2] (iGAP). The intrinsic complexity of the pipeline makes iGAP an ideal life sciences application to drive grid software development. It is a flagship application for the TeraGrid project [3]. The deployment of iGAP on the grid using grid middleware and mediator software has been described previously [4]. The heterogeneous and distributed computing environment on the grid requires an interactive user interface where jobs may be submitted and monitored. Here we describe our international collaborative effort in creating a grid

* To whom correspondence should be addressed. Phone 858-822-1079, Fax 858-822-4767

portal solution for grid computing in life sciences under the auspices of PRAGMA [5]. Specifically, the development of GridMonitor for interactive monitor of iGAP workflow, and the use of a GridSpeed [6] generated iGAP application portal are described. The portal solution was part of the EOL demonstration at Supercomputing 2003 (SC'03) [7], where resources from 6 institutions on 5 continents are utilized to annotate more than 36,000 proteins from various species. It is a testimony to the necessity and expediency for international collaboration in an increasingly global grid computational environment to advance life sciences research.

1 Introduction

The international genome sequencing effort has produced a deluge of genomic information. To date, more than 132 complete, and over 800 partial proteomes are publicly available. However, genome sequencing per se does not provide cures for diseases and cancers. There is now a global push to actively translate the genomic/proteomic information into tangible benefits for public health and education. The Encyclopedia of Life (EOL) project [1] aims to provide a comprehensive electronic reference system for biological research and education. Currently our annotation effort focuses on predicting protein structural information using an Integrative Genome Annotation Pipeline, iGAP [2]. The computational requirement of iGAP and our initial experience in using AppLeS Parameter Sweep Template (APST) [8] to deploy it on the grid has been previously described [4]. As discussed in the paper [4], a prototype Bioinformatics Workflow Management System (BWMS) is essential to facilitate the multi-genome annotation effort.

EOL has become a global collaborative effort over the past two years. During SC'03, 6 institutions from North America, South America, Asia, Australia, and Europe participated in the demonstration of grid deployment of the EOL pipeline using grid workflow software. While the experience in using the grid workflow software during SC'03 is described in details in a separate manuscript [5], the international collaborative activities to develop an integrated grid portal solution for iGAP under the auspices of PRAGMA [9] in the EOL project is described in this paper. We first provide a brief survey of common portal solutions. Then we describe our on-going collaborative efforts to provide an advanced grid portal solution for life sciences using GridSpeed and GridMonitor.

1.1 Grid Portals

Grid portals are defined as a “class of www application servers the provide a secure online environment for gathering information about grid services and resources as well as provide tools for utilizing these grid services and resources to perform useful tasks” [10]. The Grid Computing Environment Research Group [11] divides grid portals into user portals and application portals. While the former provides grid services such as single sign-on, job submission and status tracking, etc. for a Virtual Organization (VO), the latter is the interface for

most scientific users who wish to harness the computational power on the grid to execute complex application tasks in a problem solving environment (PSE). The Cactus portal [10], Astrophysics Simulation Collaboratory (ASC) [12] are early examples of the latter, whereas PACI HotPage [13] is an early example of the former.

Development efforts like GridPort toolkit [14], used for the development of PACI HotPage, or the Grid Portal Development Toolkit (GPDK) [15], aim to ease the development of grid portals. These toolkits provide an abstract API to the fundamental Globus layer. This allows a portal developer to focus on the development of the portal rather than keeping up with new Globus releases and changes. We briefly describe these toolkits below; then we identify some challenges that we tried to address during our portal development.

1.2 Toolkits for Grid Application Portals

Commodity of Grid (COG) [16] or Commodity Grid Kits is an effort to provide native and wrapper implementation of Globus [17] in programming languages such as Perl, Python and Java. The general development architecture supported by these implementations requires a user to interact directly with the COGs to access grid services.

GridPort toolkit is a Perl API built on top of the Perl CoG [18]. It employs the Globus Toolkit to connect to remote resources for job execution and information service. Therefore, it requires a local copy of the Globus client to be installed. Once an application portal is developed, the toolkit requires the user to have a valid Globus certificate in order to interact with a grid execution environment. Grid Portal Development Toolkit (GPDK) is a Java based implementation built on top of the Java CoG [19]. It contains a native implementation of the Globus client and does not require a Globus client installation on the client. A portal built with this toolkit also requires a user to have a valid Globus certificate.

GridSphere [20], part of the GridLab [21] project, provides a portlet model for developers to develop third-party portlet web applications. Once the portlet applications are deployed, it is easy for users to select portlets of interest and create a personalized grid portal. The Open Grid Computing Environment (OGCE) [22] is a recent project funded by the US National Science Foundation to provide sharable grid portal solutions, which leverages a number of ongoing portal development efforts. The myGrid project [23], part of the UK e-Science project, aims to provide web services based portal solutions.

The toolkits have ranged from simple API's to the portlet model of GridSphere with support for web services. The portlet API is standardized by the Java Community Process (JCP). Efforts are also underway to identify best practices in portal design and implementation [11]. One common theme in all these toolkits is the requirement for the application portal/portlet developer to interact directly with the underlying grid service. The people in charge of building application portals are often grid experts and/or savvy web programmers. Therefore, providing such developers with potentially sophisticated APIs is adequate so far.

Current trends show that the number and variety of potential grid applications will soon increase sharply. Application portals are to be built by developers with much varied background, especially in various scientific fields. The specialized developers are often computer literate but not necessarily computer scientists skilled in grid programming. The rapid changes in Globus toolkits, and the cascading effect on the grid portal development toolkits means that these portal developers are unable and unwilling to keep up with these changes. Consequently, we argue that there is a strong need for an integrated environment that allows application developers to generate their application portals in a straightforward fashion, without having sophisticated knowledge of either web programming or grid programming.

1.3 GridSpeed

The main goal of GridSpeed [6], developed at Tokyo Institute of Technology (TiTech), is to make it possible to meet this requirement, i.e., to allow users to dynamically generate application portals or instantiate/publish them without any programming. The overall architecture and deployment of a GridSpeed application portal [24] is depicted in Fig. 1.

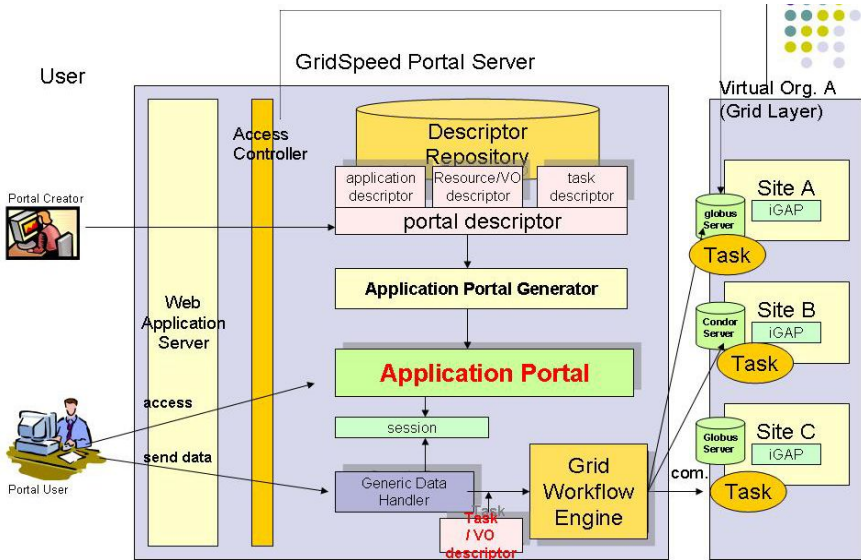


Fig. 1. GridSpeed Architecture

It provides a wizard-like web interface, the Grid Portal Generation Wizard, to dynamically generate application portals and/or publish them. The wizard guides a user to describe grid resources and target applications. Then it generates a portal that can be published to the public or restricted to certain groups.

The GridSpeed Portal Repository (Descriptor Repository) allows portal creators to publish their generated application portals. Generated application portals on the GridSpeed server can be published and shared among other user. It also provides role-based access control mechanism to limit the usage of registered application portals in the system. Users may search the Repository for their target applications or describe the name, category, manufacturer, physical resources to be used.

GridSpeed separates applications from resources clearly. A portal corresponds to the binding from an application to multiple resources. This makes it possible for a portal administrator to reuse an application interface published by someone else, and bind it to any set of resources.

Application Structures and Grid Middleware Supported by GridSpeed. For GridSpeed to be useful to a large community of users it must support a wide spectrum of application structures that are relevant to existing scientific applications. Currently, GridSpeed supports the following: 1) A single task to be invoked once, 2) A single task to be invoked multiple times with different input, or 3) combination of the above, with possible dependencies among tasks. The first type is representative of many applications, for instance, ones in which users periodically compare their experimental data with a model. The second type is often labeled as a parameter sweep application and has been shown to be well-suited to grid executions. Other scientific applications that have more complex structures are of the third type. GridSpeed offers a simple workflow schema to describe and support these application structures.

In addition, different applications may use different computing services such as Globus and SSH, different schedulers such as Condor [25], LoadLeveler, LSF, PBS, etc., and different data services such as GridFTP [26] and SRB [27]. GridSpeed meet these requirements by providing a higher-level resource description bound to tasks. GridSpeed currently makes use of APST as the grid workflow engine and meta-scheduler. DAGMan [28] and UNICORE [29] are examples of other workflow engines to be supported.

2 Generating iGAP Application Portal with GridSpeed


The Grid Portal Generation Wizard is organized as a set of web pages in which the user is asked to answer questions concerning grid resources and the target application. Specifically, an application portal is generated by defining two objects: a computing environment object and an application object, and then these objects are bound together. Definitions are independent from each other, thus, objects need only to be defined once and can be reused by the same or other portal creators. We illustrate the process of instantiating the iGAP application portal as follows:

2.1 Computing Environment

The first step of defining the computing environment is performed by resource providers, allowing them to register their computing environment that can be used by portals generated by GridSpeed. A computing environment is comprised of storage resources (disks) and of compute resources (hosts). Each disk requires a host server, a default directory for application input/output, an access protocol such as GridFTP, SRB or SFTP (Fig. 2). Each host requires an access protocol such as Globus Gram, SSH, and scheduler methods such as Condor, PBS or LoadLeveler.

Storage > Files > Compute > Registration

Computing Environment : Data Storage



The storage element specifies disks where application input is found and output will be generated. Information for the local disk (attached to the host where apsd is running) is automatically generated if not included in the XML. Each disk may specify an access method, the default is local.

[More Disk](#) | [Delete All Disks](#)

Disk ID	Protocol	Hostname	Data Directory	
Teragrid_SDSC	scp	wilfred@logon1.sdsc.edu	/users/wilfred/eol	Delete
NBCR	scp	wilfred@nbc6.sdsc.edu	/projects/eol/wilfred/eol	Delete
NPAClogon1_M	scp	ux429426@morpheus.engr.uiowa.edu	/home/ux429426/eol	Delete
NPAClogon1_H	scp	ux429426@hypoos.engr.uiowa.edu	/home/ux429426/eol	Delete
Condor_Titech	scp	wilfred@port.is.titech.ac.jp	/home2/cc/qsar32/eol	Delete
RL_Viper	scp	viper.bnl-eg.org	/home/wilfred/eol	Delete


[Back](#) [Next](#)

[Define Files \(just only for specialists\)](#)

Dump Storage

Storage > Files > Compute > Registration

Computing Environment: Hosts



Each host may specify an access method and one or more scheduler methods, the defaults are local (process spawning via fork) and shell (direct execution rather than through a batch scheduler). Although the DTD allows multiple access methods to be specified, apsd presently ignores all but the last.

[More Host](#) | [Delete All Hosts](#)

Host ID	Protocol	Hostname	Disk	Directory	CPUs	Batch Type	OS	
Teragrid	ssh	wilfred@logon1.sdsc.edu	Teragrid_SDSC		1	none	none	Delete

Host ID	Protocol	Hostname	Disk	Directory	CPUs	Batch Type	OS	
NBCR/H	ssh	wilfred@nbc6.sdsc.edu	Teragrid_SDSC		1	none	none	Delete

Host ID	Protocol	Hostname	Disk	Directory	CPUs	Batch Type	OS	
NPAClogon1	ssh	ux429426@morpheus.engr.uiowa.edu	Teragrid_SDSC		1	none	none	Delete

Host ID	Protocol	Hostname	Disk	Directory	CPUs	Batch Type	OS	
NPAClogon1_H	ssh	ux429426@hypoos.engr.uiowa.edu	Teragrid_SDSC		1	none	none	Delete

Host ID	Protocol	Hostname	Disk	Directory	CPUs	Batch Type	OS	
Condor	ssh	wilfred@port.is.titech.ac.jp	Teragrid_SDSC		1	none	none	Delete

Host ID	Protocol	Hostname	Disk	Directory	CPUs	Batch Type	OS	
RL_Viper	ssh	viper.bnl-eg.org	Teragrid_SDSC		1	none	none	Delete

[Back](#) [Reset](#) [Next](#)

Fig. 2. Data Storage and hosts

2.2 Application Information and Parameters

Next, the user fills in various information about the target application, e.g., name, subtitle, manufacturer, category, textual description, in a form. Such information is used to categorize applications, allowing one to search for target applications with the keywords. This information may be optionally displayed (Fig. 3).

The user also decides which application parameter is exposed in the generated portal by specifying its name, widget type, title, data type, method type,

Information → **Parameter** → **Template** → **Pipeline** → **Registration**

Application Information

In this page, you fill in a form with various information about the target application (name, subtitle, manufacturer, category, textual description, etc.) Such information is used to categorize applications, allowing one to search for target applications with the keywords. This information is also displayed in the generated application portal.

Application Information	
Name	<input type="text" value="Encyclopedia of Life"/>
Subtitle	<input type="text" value="Q&P Portal that shepherds genomic data through sevens"/>
Category	<input type="text" value="Biology"/> <input checked="" type="checkbox"/> biology
Manufacturer	<input type="text" value="San Diego SuperComputer Center, Bioinformatics Instt"/>
URL	<input type="text" value="http://eolodc.edu/"/>
Image	<input type="text" value="http://eolodc.edu/images/index_0_02.gif"/>
Template Head	<input type="text" value="/jsp/Templates/eol/header.jsp"/>
Template Tail	<input type="text" value="/jsp/Templates/eol/footer.jsp"/>
Description	<div style="border: 1px solid black; padding: 5px;"> <p>The Encyclopedia of Life (EOL) is an ambitious project to catalog the complete proteome of every living species in a flexible, powerful reference system. An open collaboration led by the San Diego Supercomputer Center, the EOL will generate biological insight using the world's foremost academic computational resources. This includes calculating three-dimensional models and assigning biological function for all recognizable proteins in all currently known genomes. Scientists will be able to uncover the prevalence of a given protein across all kingdoms of life, molecular interactions with that protein, and whether the function of the protein varies across species. The EOL caters to a diversity of users, from researchers interested in proteomic associations, to undergraduates wishing to know the name and function of proteins associated with a</p> </div>

Information → **Parameter** → **Template** → **Pipeline** → **Registration**

Defining Parameters

This page allows the definition of a parameter to be exposed on the generated portal by specifying its name, widget type, title, data type, method type, default value, and description. The name is used to be referenced from the Template Page and the Task Page.

Name	Title	Widget	Data	Method	Value	Desc	Adv Action
genome	Genome	upload	<input type="text" value="string"/>	<input type="text" value="string"/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="button" value="Update"/>
<input type="button" value="New"/>							

Parameter List

Name	Title	Widget	Data	Method	Value	Description	Action	Action
genome	Genome	upload	string	string			Edit	Delete
species	Species	text	string	string	AGRRD		Edit	Delete
category	Category	text	string	string	11042003		Edit	Delete
source	Source	text	string	string	nctd		Edit	Delete
sequence	Sequence	text	variant	string	1.1.1		Edit	Delete
hvalue	H-Value	text	string	string	1e-6		Edit	Delete
round	Round	text	string	string	3		Edit	Delete
numseq	Numseq	text	string	string	1		Edit	Delete
EOL_HOMEDIR	EOL Homedir	text	string	string	/home/watfred/eol		Edit	Delete
EOL_DATADIR	EOL Datadir	text	string	string	/home/suzumura/eol_data		Edit	Delete
debug	Debug Mode	select	string	string	true		Edit	Delete
id-prefix	Task ID Prefix	text	string	string	pstblast		Edit	Delete

[Add Parameter](#) | [Delete All Parameters](#)

[Back to Information](#) | [Proceed to Tasks](#) | [Proceed to Templates](#)

Fig. 3. Application Information and Parameters

default value, and description. The widget type is used to represent the actual widget component for the parameter and can be chosen from “text”, “textarea”, “select”, “upload file”, and “password”. For a parameter whose widget type is “select”, one must input a list of items that is name-value pairs, so that portal users can select one or more of these items. Moreover, a parameter whose widget type is “upload file” is a file uploaded by the portal user. Some parameters must be passed to applications as a file even though they are inputted by users as string. For this case, one can specify “file” as the method type and then a temporary file containing the value of the parameter is created at runtime and passed to the application.

2.3 Advanced Application Archietcture

As iGAP is a set of bioinformatics applications pipelined with task dependencies, the portal creator can define an advanced structure for the target application, denoted by “application pipeline” in the wizard (Fig. 4). The pipeline is structured as one or more tasks. A task corresponds to the action of launching a

Application Pipeline

Your application pipeline is structured as one or more tasks. A task corresponds to the action of launching a certain executable regardless of whether the executable is grid-enabled or not. Each task can be defined by filling in an executable path, input, output, stdout, stin, stdout, estimated runtime, priority, and a description. Dependency between multiple tasks can be controlled by adjusting the task priorities. A task with higher priority can be executed at earlier stage. In each form, it is possible to refer to the actual user input data for each parameter defined in the Parameter Page and the actual instantiated file from a template predefined in the Template Page in the same manner as the Template Page.

[More Task](#) | [Delete All Tasks](#)

Task

☐ Delete | ☒ Detailed Options

Name

Host

☐ run on [Base Host](#) ?

Path

Argument

-F -debug {debug} -D

{EOL_DATADIR} -R {EOL_HOMEDIR} -genome

{EOL_DATADIR}/{genome} -species {species} -

source {source} -category {category} -numseq

{numseq} -psiblast -matrix_single -psiblast -

Input

Output

db/{species}/{source}/

{category}/data/psm/psiblast/{hvalue}

{round}/seq{sequence}.matrix tmp/{species}

Variant

string task word value

☐ sequence ☐ ☐ ☐ ☐

Directory

Stdout

Stderr

Stdin

Priority

Cost

Memory

Description

Tasks [Add](#)

☐ {id-prefix}/

[{sequence}](#)

Parameters [Add](#)

☐ genome

☐ species

☐ category

☐ source

☐ sequence

☐ hvalue

☐ round

☐ numseq

☐ EOL_HOMEDIR

☐ EOL_DATADIR

☐ debug

☐ id-prefix

Templates [Add](#)

Fig. 4. iGAP Pipeline Description

certain executable. Each task can be defined by filling in an executable path, input, output, stdout, stderr, estimated runtime, priority, and a description. Dependency between multiple tasks can be controlled by adjusting the task priorities. A task with higher priority can be executed at an earlier stage. In each form, it is possible to refer to the actual user input data for each parameter defined in the Parameter Page (as shown in Fig. 3).

2.4 iGAP GridSpeed Portal Interface and Monitoring

2.5 Portal Publication

In the GridSpeed context, an application portal is considered to be an interface to the application along with specific set of resources. Therefore, as the final step, generation of a portal is accomplished by binding an application with multiple computing environments through this step. This binding operation realizes on-demand creation of application portals.

The iGAP Portal (Fig. 5) is generated by selecting one application interface and one computing environment, both of which are already defined and published

The screenshot displays the iGAP GridSpeed Portal interface, which is divided into several functional areas:

- Header:** Features the 'ENCYCLOPEDIA OF LIFE' logo and the title 'iGAP on the Grid'.
- Left Sidebar:** Contains navigation links such as 'iGAP HOME', 'ABOUT EOL', 'GENOME STATUS', 'GENOME SUMMARY', 'CURRENT TASK', 'RESOURCES', 'EOL PARTNERS', 'EOL HOME PAGE', and 'EOL HELP'.
- Parameter Setting:** A section for configuring application parameters. It includes fields for 'Genome', 'Species', 'Category', 'Source', 'Sequence', 'EOL Homedir', and 'EOL Datadir'. Below these are 'Advanced Setting' fields for 'HValue', 'Round', and 'Numexp', with 'Submit' and 'Reset' buttons.
- Data Repository:** A section for managing data. It includes an 'Upload File' button, a 'Create a Directory' button, and a table listing files in the 'Alemoldata' directory. The table has columns for 'Filename', 'Size', and 'Last modified'.

Filename	Size	Last modified
Top		
U		
echo-output-message	12	Oct 31, 2003
echo-output-message.stderr	0	Oct 31, 2003
repository.PG	216398	Oct 29, 2003
output.jpg	190438	Oct 30, 2003
convert.stderr	0	Oct 30, 2003
convert.stdout	0	Oct 30, 2003
CSCH0077.PG	1413408	Nov 1, 2003
- Monitoring:** A section for monitoring task execution. It includes a table showing task status (Time running, Performed, Total, New, Running, Finished, Failed) and a 'Submitted Tasks' table with columns for 'id', 'status', 'host', 'started', 'finished', 'output', 'stdout', 'stderr', and 'Action'. There is also a 'Refresh Interval' button and a 'Registered Disks' section.
- Footer:** Indicates the page is generated by GridSpeed and designed by Edna Nepora. It also includes a copyright notice: '©2003 Encyclopedia of Life. All rights reserved.'

Fig. 5. iGAP GridSpeed Portal interface and monitoring

in the previous steps. From this portal page, a user can upload a set of proteins or a complete proteome, which is the input for iGAP. All subsequent steps are automatically executed on the user's behalf using APST. A simple web client queries the APST daemon for task status to be displayed. The data and output for particular tasks may be accessed from the web if desired.

3 GridMonitor Portal Design and Development

While the application portal generated with GridSpeed provides the interactive interface for a web user to submit proteins or proteomes of interest for analysis using iGAP, the large scale automated analysis of all proteomes in the EOL project requires a robust, scalable workflow management system which tracks overall progress and provide recovery information in case of resource failure [4]. Such a system also requires an interactive user interface for monitoring of genome annotation status and task progression. We now describe the GridMonitor Portal developed by the Bioinformatics Institute in Singapore.

3.1 System Overview

Integrative Genome Annotation Pipeline (iGAP) incorporates well-established bioinformatics tools to predict the structure of a protein from its sequence. When annotating a large number of protein sequences, the pipeline is required to run for weeks utilizing geographically and internationally distributed compute and storage resources. One unique feature of this PSE is that various users of the pipeline do not necessarily want or need to execute their own pipeline instance. They only need to queue their jobs with the appropriate priority and leave it to the workflow management system to handle the job submission and execution.

Currently, the iGAP portal consists of two main front-end components: a publicly available information portal, and a restricted access admin portal. The portal architecture (see Fig. 6) consists of information and admin portals sharing a common backend relational database. The database allows retrieval access for the information portal, and updates/modifications for the admin portal. This design allows abstraction of grid middleware, workflow and job management related issues from the actual portal developers. Using this design, the information and admin portals can be developed using appropriate tools and application frameworks. The prototype implementation of the portal currently consists of the Information Portal only.

The implementation of information portal is completed in Java Server Pages (JSPs), using the open source Tomcat servlet container bundled with another open source Java 2 Enterprise Edition (J2EE) compliant application server, JBoss [30]. To implement the portal we used a 3-tier model: client, web and database tier. As in the situation with the development of BWMS [5], the number of emerging open source solutions are either premature or not easily adaptable to the production requirement of iGAP. Our approach is to support routine production use of the grid and comply with emerging grid service standards as

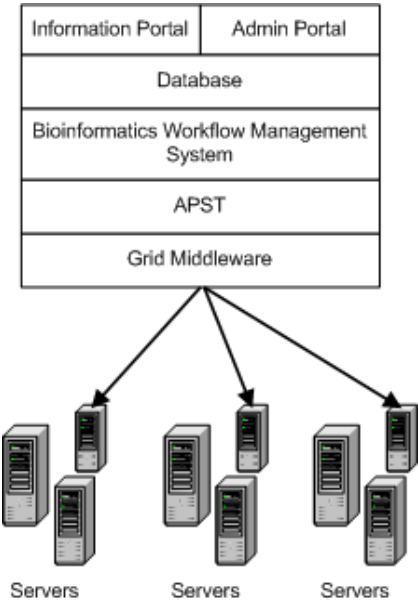


Fig. 6. iGAP Grid Monitor Portal Architecture

they stabilize. The current 3-tier model provides the functionalities required and offers excellent performance.

3.2 GridMonitor for iGAP

The iGAP portal is structured to present users with the ability to monitor the progress of the pipeline at either the proteome level, or the task level. The underlying information is retrieved from the BWMS relational database engine.

The Task Status page (Fig. 7) offers detailed look for all the tasks associated with a protein. A task is a single application executed with a single protein sequence on one of the requested cpus. The task execution host, run time, and status may all be viewed from the web interface. As tasks for each proteome are distributed to different resources, the status of a particular task may be queried to see which resource is used, and this information may be used by meta-schedulers to determine throughput from a particular resource.

Genome summary lists the genomes and at what state they are in: Not Submitted, Submitted, Not Processed or Processed. All genome names are further linked to provide the user with a small summary about the genome, structure id and structure information, if applicable.

The Genome Status page allows the user to look at the overall progress and see how many genomes have been submitted to the pipeline and how many have finished. Those interested in the status of a particular genome can select the genome of their choice and find specific information pertaining to it. A hyperlink to the EOL Book is also present for more information.



Fig. 7. The IGAP GridMonitor Task Status Page

4 Discussion

The large scale proteome annotation undertaken by the EOL project poses challenges to grid software development in areas such as grid workflow, scheduling and portal development. The rapid change in Globus Toolkits (GT) in the past few years is necessary in the long term, but poses a challenge to running life sciences applications on the grid. In particular, the current scientific requirement of proteome analysis cannot wait until grid computing standards are stable. Fortunately, with the hard work of numerous developers worldwide, the situation is improving. PRAGMA has provided a mutually beneficial environment where developers from the Pacific Rim can establish collaborations, exchange ideas and promote grid middleware and application development.

The collaborative effort on the EOL project by developers from BII, TiTech and SDSC is crucial in making the demonstration at SC'03 [7] a success. Grid-Speed from TiTech provided a much needed application portal where users can submit jobs from a web interface. The job submitted now includes whole proteome analysis made possible by the development of grid computation resources.

The GridMonitor from BII provides a prototype user portal where large scale automated analysis may be monitored. Under the hood, both GridSpeed and GridMonitor utilize APST, part of the NMI [31] and NPACKage [32], for grid scheduling and task execution.

While both GridSpeed and GridMonitor uses APST as the underlying grid execution engine, GridSpeed generates the task XML input for APST, whereas GridMonitor relies on BWMS to provide the XML input. Since APST relies on a XML file for task description, resource description, these components may be loosely coupled together. The XML input from GridSpeed may be consumed by BWMS so that tasks submitted by GridSpeed may also be monitored from GridMonitor. The XML input required by APST may eventually adopt the developing standards of workflow specification language [33].

The generation of the application portal by GridSpeed is a simple process. However, APST was originally designed for a single user because it aims at user-level scheduling. Clearly using APST as GridSpeed's workflow engine is not scalable in terms of number of simultaneous users. Potentially this problem may be resolved using BWMS, which may be enhanced to record multiple user requests and dispatches them to APST. In fact, such a model will also handle situations where multiple users submit the same proteome to be processed, or where a proteome has already been scheduled by the automated proteome annotation update process. The disadvantage of such a system is that BWMS must act as a central server for user requests. Such scenarios are also common in data services using SRB or GridFTP and solutions are not easily available at this time.

Grid security is another topic receiving much attention recently. The development of a user portal which supports GSI (Global Security Infrastructure) with single sign on and data privacy is important for production use of life sciences applications. The economic model of the grid also dictates that some users will be able to afford access to more resources. Therefore, our immediate goal is to investigate new developments in the grid portal toolkits, and improve the reusability of our software.

Our future plan is to generate a loosely coupled portal solution using GridSpeed, and GridMonitor. Currently the portlet technology appears most suitable. GridSpeed will support "GridSpeed Portlets", which can be integrated and runnable on the portlet containers such as GridSphere and JetSpeed. GridSphere is nearly compatible with IBM WebSphere 4.2. While initially WebSphere portlet API was based on JetSpeed; WebSphere 4+ API is proprietary. The future portal architecture will be built to leverage open source technologies like Struts, JetSpeed, Turbine, Velocity, etc. [34], while abstracting the underlying grid services architecture to BWMS and APST. Currently we are also evaluating GridSphere, OGCE portal and myGrid development efforts.

A more distant goal is to realize distributed data service on the grid from the grid portal. This will allow user to view analysis results on the fly while proteins are being annotated on the grid. We are currently exploring the use of OGSA-DAI [35] to allow the users to access the EOL data warehouse, which contains

value added information besides the pipeline output. The information will be integrated in a more general grid portal, which will encompass GridMonitor, GridSpeed, BWMS and the EOL data warehouse. The end result would be state of the art high throughput proteome analysis in the grid environment.

5 Conclusion

In this paper, we have shown that by leveraging on the collective expertise of the 3 PRAGMA sites, SDSC, TiTech and BII, we were able to speed up the whole process of development and deployment of iGAP grid portal. The scientific achievement from the EOL demo during SC'03 shows that grid technology is finally beginning to enable biological research. The reciprocity between grid software development and high throughput proteome analysis will make the routine use of the grid a reality in the near future.

Acknowledgements

The authors would like to thank Dr Gunaratnum Rajagopal, Elise Tan, and Stephen Wong from the Bioinformatics Institute for valuable input; Jim Hayes, Adam Birnbaum, Henri Casanova at SDSC for their work in the workflow and APST; Hidemoto Nakada at TiTech for help during GridSpeed development. P.W. Arzberger wishes to acknowledge the support of the National Science Foundation for PRAGMA (Grant No. INT-0314015), as well as that of the National Institutes of Health for the National Biomedical Computation Resource (P 41 RR08605). PRAGMA is supported by its twenty member institutions and their respective funding organizations. W.W. Li is also supported by the National Partnership for Advanced Computational Infrastructure (NPACI), funded by the National Science Foundation (NSF) Grant No. ASC 9619020, and Systematic Protein Annotation and Modeling, funded by the National Institutes of Health (NIH) Grant No. GM63208-01A1S1.

References

1. The Encyclopedia of Life Project, <http://eol.sdsc.edu>.
2. Li, W. W., Quinn, G. B., Alexandrov, N. N., Bourne, P. E. & Shindyalov, I. N., "A comparative proteomics resource: proteins of *Arabidopsis thaliana*", *Genome Biol*, 4, pp.R51, 2003.
3. TeraGrid, <http://www.teragrid.org>.
4. Li, W. W. et al., "The Encyclopedia of Life Project: Grid Software and Deployment", *New Generation Computing, In Press*, 2004.
5. Birnbaum, A. et al., "Grid Workflow Software for High-Throughput Proteome Annotation Pipeline", *Life Sciences Grid Workshop Proceedings, Submitted*, 2004.
6. GridSpeed, <http://www.gridspeed.org>.
7. Supercomputing 2003, <http://www.sc-conference.org/sc2003/>.

8. Casanova, H. & Berman, F. in *"Grid Computing: Making the Global Infrastructure a Reality"*. (eds. Berman, F., Fox, G. C. & Hey, A. J. G.). Wiley Publishers, Inc., West Sussex, 2003.
9. Pacific Rim Applications and Grid Middleware Assembly, <http://www.pragma-grid.net/>.
10. Cactus Project — An open source problem solving environment, http://www.cactuscode.org/Presentations/GridPortals_2000.ppt.
11. Grid Compute Environment (GCE) Research Group-Global Grid Forum, <http://www.computingportals.org>.
12. The Astrophysics Simulation Collaboratory (ASC), <https://www.ascportal.org>.
13. NPACI HotPage Grid Computing Portal, <http://hotpage.paci.org>.
14. Thomas, M. et al., "The GridPort Toolkit Architecture for Building Grid Portals", in *the 10th IEEE Intl. Symp. on High Perf. Dist. Comp.*
15. Novotny, J. in *"Concurrency and Computation: Practice and Experience"*. pp.1129–1144, 2002.
16. Commodity Grid Kits, <http://www-unix.globus.org/cog/>.
17. The Globus Alliance, <http://www.globus.org>.
18. Perl CoG, <https://gridport.npaci.edu/cog/>.
19. von Laszewski, G., Foster, I., Gawor, J. & Lane, P., "A Java Commodity Grid Kit", *Concurrency and Computation: Practice and Experience*, 13, pp.643–662, 2001.
20. GridSphere, <http://www.gridsphere.org>.
21. GridLab, <http://www.gridlab.org>.
22. OGCE: Open Grid Computing Environment, <http://www.ogce.org/index.php>.
23. The MyGrid Project, <http://www.ebi.ac.uk/mygrid/>.
24. Suzumura, T., Nakada, H. & Matsuoka, S., "GridSpeed: A Web-based Grid Portal Generation Server", in *CCGrid*, Poster.
25. Litzkow, M., Livny, M. & Mutka, M., "Condor - a hunter of idle workstations." in *Proceedings of the 8th International Conference of Distributed Computing Systems*, 104–111, June 1988.
26. Allcock, W. et al., "GridFTP: Protocol Extension to FTP for the Grid, Grid Forum Internet-Draft", in March, 2001.
27. The Storage Resource Broker, <http://www.npaci.edu/dice/srb>.
28. DAGMan, <http://www.cs.wisc.edu/condor/dagman/>.
29. UNiform Interface to COmputing REsources (UNICORE), <http://www.unicore.de/>.
30. JBOSS, <http://www.jboss.org>.
31. NSF Middleware Initiative, <http://www.nsf-middleware.org/>.
32. NPACIgrid, <http://npacigrid.npaci.edu/>.
33. Workflow Management Research Group, <http://www.isi.edu/~deelman/wfm-rg/>.
34. The Apache Jakarta Project, <http://jakarta.apache.org>.
35. OGSA-Data Access and Integration, <http://www.ogsadai.org.uk/>.

Grid Workflow Software for a High-Throughput Proteome Annotation Pipeline*

Adam Birnbaum¹, James Hayes¹, Wilfred W. Li¹, Mark A. Miller¹,
Peter W. Arzberger², Philip E. Bourne^{1,2}, and Henri Casanova^{1,2}

¹ San Diego Supercomputer Center,
La Jolla, CA 92093, USA

² University of California, San Diego,
La Jolla, CA 92093, USA

{birnbaum, jhayes, wilfred, miller, parzberg, bourne, casanova}@sdsc.edu

Abstract. The goal of the Encyclopedia of Life (EOL) Project is to predict structural information for all proteins, in all organisms. This calculation presents challenges both in terms of the scale of the computational resources required (approximately 1.8 million CPU hours), as well as in data and workflow management. While tools are available that solve some subsets of these problems, it was necessary for us to build software to integrate and manage the overall Grid application execution. In this paper, we present this workflow system, detail its components, and report on the performance of our initial prototype implementation for runs over a large-scale Grid platform during the SC'03 conference.

1 Introduction

In 1995, the influenza virus became the first genome to be completely sequenced. Since that time, more than 1000 other organisms have been at least partially sequenced and are freely available at the National Center for Biotechnology Information (NCBI) [30]. As of this writing, this includes the sequence information of more than 1.6 million different proteins in the non-redundant (NR) database. However, protein sequence information is only part of the story. While a protein sequence determines its shape, and biological function, the ability to predict these features from the sequence remains one of the elusive goals of modern biology. At present, structure and function determinations rely upon laborious experimental methods. For example, the structures of approximately 24,000 unique proteins have been measured experimentally and deposited in the

* This research was supported in part by the National Science Foundation under the NPACI Cooperative Agreement No. ACI-9619020 and under award No. ACI-0086092. W.W. Li, is also supported in part by PRAGMA, funded by NSF Grant No. INT-0314015, and Systematic Protein Annotation and Modeling, funded by the National Institutes of Health (NIH) Grant No. GM63208-01A1S1.

Protein Data Bank [9]. Despite the advent of high throughput crystallization pipelines [25], there is still a large disparity between the rates of protein structure and sequence determination. Strategies must be adopted that allow one to infer protein structure and functions based on sequence alone in order to gain immediate benefits from genome sequencing efforts.

As part of a large body of works devoted towards this goal, we had previously described the construction of a *fold library* (i.e., a library of known structural templates based on known structures), and the development of an Integrative Genome Annotation Pipeline (iGAP) [28]. iGAP, which incorporates a number of well established bioinformatics applications, predicts protein structure based on sequence and structural similarity to the fold library templates. It has been successfully used to annotate more than 130 genomes in an international effort known as the Encyclopedia of Life project, designed to catalog the complete proteome of every living species in a flexible reference system [20]. Our goal in this work is to scale this system to process all existing protein sequences, without human intervention, and to deposit the output in a public database for access by the entire scientific community.

The magnitude of the computation to be accomplished has been quantified and discussed previously [27]. Briefly, our estimate is that the overall calculation will require approximately 1.8 million CPU hours on a 1.8 GHz CPU, or more than 200 years on a single CPU. This calculation needs to be repeated and updated as more sequences and structures become available. Because of the overall scale of these computational requirements, it was necessary to deploy iGAP on a multi-institutional international Grid platform. The complexity of the calculation made it necessary to develop a domain specific bioinformatics workflow management system, to automatically manage the transfer of data files and the correct and efficient distribution of calculations to distributed resources, while maintaining correct records on the ongoing state of the overall computation.

In this paper, we report the progress we have made in developing an integrated grid workflow solution for EOL, a prototype of which was demonstrated at SC'03 using storage and compute resources distributed over various international partners, including PRAGMA [33] members such as the Tokyo Institute of Technology (TiTech), SDSC, and Singapore's Bioinformatics Institute (BII). The collaborative effort between the international partners will be described in a separate submission. In this particular paper, we focus on the grid workflow software, specifically:

1. We provide technical details about the implementation, which were driven by and critical for EOL but are relevant to Grid applications at large;
2. We report on the status of our implementation and present experimental results obtained on a large-scale Grid during the SC'03 conference; and
3. Based on this hands-on experience with our prototype software, we have identified two scalability and performance bottlenecks that we have removed in the current version of the software.

2 The APST Grid Execution Environment

2.1 Challenges for Deploying Grid Applications

The EOL project is an example of a Grid application that is attempting to achieve high levels of performance and scale via the use of a Grid platform where aggregated resources (in our case compute, storage, and data collections) are distributed among different institutions nation- or world-wide [22, 7]. Three main challenges typically arise for such application deployment: (i) application scheduling; (ii) resource heterogeneity; and (iii) deployment logistics. We briefly discuss each challenge below; we then introduce the AppLeS Parameter Sweep Template (APST) and explain how it provides software technology that addresses the above challenges at least partially.

The *scheduling* problem, i.e. assigning application components to resources with a view to optimize some metric of performance, has been studied for decades and there is a large literature for different classes of applications (e.g. see [26, 11] for surveys). Unfortunately, most algorithms are based on assumptions that do not hold on Grid platforms. For instance, it is often assumed that it is possible to obtain 100% accurate predictions of computation and data transfer times, that resources deliver constant performance over time without failing, or that network topologies are simple (e.g. fully connected). For this reason, practical solutions for Grid scheduling generally involve the development of adaptive scheduling and/or runtime re-scheduling techniques, rather than the direct use of existing scheduling algorithms [8, 10, 12, 15, 18, 34, 37, 39]. In the specific context of high throughput computations, a number of tools are available that automatically schedule Grid applications using such techniques (e.g., Condor [36], Nimrod/G [1], APST [14]).

When deploying a Grid application, these questions of efficient resource use may be dwarfed by the overwhelming practical problems caused by the *heterogeneity* of the resources, which can vary in operating system, local schedulers, and other factors. The Globus Alliance [21] has strived to reduce some of the barriers raised by resource heterogeneity by providing a single security model, and a consistent interface for accessing various batch schedulers. However, Globus has not yet achieved ubiquity, and it suffers from the rapid rate of change in its APIs over the last several years. Therefore, for the time being, Globus must be viewed as one system among a variety of systems the application developer must deal with. In their more recent work, the Globus Alliance has focused on creating standards and prototypes to allow distributed scientific applications to be built using Web Services, which are important emerging enterprise technologies [31]. The use of these standards will require major changes in the software infrastructure for computational centers, which will in turn imply major changes in the overall approach to application and system design. While there is good reason to believe that these last developments will provide a ubiquitous, standardized Grid software infrastructure in the medium-term, our goal is to deploy and run the EOL application on as many systems as possible today.

A second challenge is that of managing the *logistics* of application deployment. When running large applications (e.g., many thousands of tasks, input files, and output files), it is advantageous to use a scalable and robust (i.e., resilient to software and hardware crashes) bookkeeping system to track resource status, computation status, file transfer status, and file (and replica) locations. Providing this type of functionality in a scalable way raises complex computer science questions, as it often requires a distributed implementation and distributed algorithms (e.g., at the Virtual Organization level [23]). In our context however, the system merely needs to keep track of data relevant to one application execution, which can be done in a centralized fashion, for instance in a centralized bookkeeping server using a relational database. The main requirements are that the bookkeeping must be completely transparent to the user, efficient, and resilient to server failures. Furthermore, it is clearly necessary that all application deployment (e.g., job launching, monitoring, fault-detection and restart, data movements) be automated.

2.2 The APST Software

Since our goal in this work is to execute a large number of essentially independent tasks to generate a biology result, rather than to conduct research on novel software infrastructures, we chose to build on an existing platform that already meets many of the criteria required for our system. We selected the AppLeS Parameter Sweep Template (APST) [14, 4], which has already demonstrated success in managing heterogeneous Grid resources for real-world applications [13]. The initial goal of the APST project was to address challenge (i) in Section 2.1, and thus APST currently uses intelligent adaptive scheduling heuristics.

APST runs as a user agent, and is able to run unmodified applications (as long as these applications perform I/O via files). APST can use a variety of methods to access computational resources (Globus [21], Condor [36], NetSolve [2], batch schedulers – LoadLeveler, LSF, PBS –, or even simple SSH connections), and to store and transfer data (GridFTP [3], SRB [5], SCP). In addition, APST embeds an intelligent scheduler that makes use of Grid resource information (available via MDS [17], NWS [38] and Ganglia [24]) to make appropriate decisions when mapping computation and data to heterogeneous Grid resources. This is all done “under the hood” by a software layer that abstracts away the differences among the resources and the way in which they are accessed, thereby addressing challenge (ii) in Section 2.1. Note that this software abstraction layer is a completely separate and reusable module, Elagi [19], that we have developed and now distributed along with APST.

As depicted in Fig. 1, the main APST program runs as a background process or daemon that can receive commands from an APST client (APST provides both a command-line client and a Java API). These commands are structured with an XML syntax and make it possible to make APST aware of new available resources, new application tasks to execute, and new application data files, and to check on the status of the computation. APST uses the aforementioned

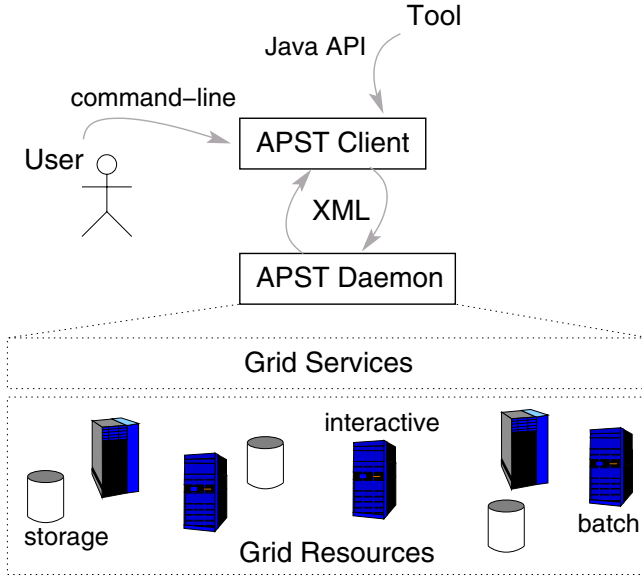


Fig. 1. APST Software Architecture

Grid services to transparently deploy application tasks and data onto compute resources (both batch and interactive) and storage resources.

In the next two sections we discuss the way in which APST manages batch-scheduled resources, and how APST addresses only parts of challenge (iii).

2.3 Batch-Scheduled Resources in APST

The computing resources available to the EOL project consist of a mixture of interactive workstations and large systems and clusters controlled by batch schedulers. Since queue wait times on batch systems vary unpredictably, this mixture poses a particular challenge to the APST scheduler. Any EOL tasks placed into a batch queue may delay dependent tasks if the queue wait is long. On the other hand, rescheduling such tasks to interactive resources and submitting different ones in their place would mean losing our place in the queue.

To provide the APST scheduler with flexibility in allocating tasks to batch resources, we submit agent processes instead of individual EOL tasks to the batch scheduler. When the batch job runs, these agents contact the APST daemon to let it know that the resource is available for use. This contact is normally made through a proxy running on the submission node of the batch system (i.e., the “front-end”), to accommodate systems where batch nodes cannot contact outside machines directly. We call this architecture RASH (Reverse Access SHell) and it is depicted in Fig. 2. When notified that batch resources are available, the APST scheduler recomputes its EOL task schedule and begins sending commands to the agents, directing them to run particular tasks. APST continues to give the

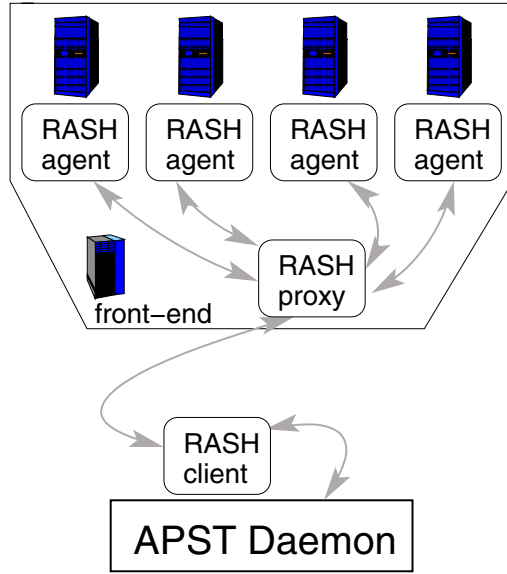


Fig. 2. RASH: software architecture supporting the use of batch-scheduled resources in APST

agents work as long as the batch resources are available and there are unassigned tasks to do. When the batch job times out, the APST scheduler detects the loss of the resources and computes a new EOL task schedule without the batch nodes.

Earlier versions of APST did not support batch systems. However, they did allow the user to add new computing resources during the run and provided fault recovery mechanisms under which the APST scheduler would recompute its task schedule when a computing resource failed. The combination of these two features allowed us to add support for batch systems with very little modification to the APST scheduler. Computing resources that become available when a batch job begins running are treated identically to those added by the user. When a batch job times out, the loss of resources is handled by the existing fault recovery mechanisms.

In the present implementation, the APST user determines the parameters of batch submissions—the number of processors, the amount of time, etc. When a batch job times out, APST submits a new one with the same parameters. In future work we plan to investigate how we might anticipate job time-out to minimize the delay between one batch job finishing and the next one starting. We also plan to investigate whether APST can efficiently take over responsibility for determining the parameters of batch submissions.

Due to the lack of a global Grid queue for job submission, several authors have explored ways in which applications can take advantage of multiple batch-scheduled system. For instance, the work in [35] explores the notions of submitting multiple requests for the same job to different batch systems and then using

only the request that acquires processors first. Our work bears a fundamental difference with such approaches as we allow for “late binding” of computation to processors: when a request is submitted to a batch queue one does not know which application tasks will be executed on the request processors once they are acquired. This is not a new concept and the Condor “Glide-In” [16] functionality makes this possible in the specific context of Condor and Globus. Most related to our work is the “placeholder scheduling” work [32], which we contemplated reusing within APST. However, one key difference in our work is that we run an agent on each acquired processor. This is necessary to assign different sequential application tasks to individual processors (as opposed to, say, a single MPI job to all processors in a batch), and to assign multiple sequential application tasks to a processor throughout time while the processor is available. These two capabilities are key for the EOL application as many of the tasks are sequential.

2.4 APST Usage

In a typical APST usage scenario, a user first assembles a large XML file that describes all of the tasks that must be run, and a set of resources available for executing these tasks. The user then uses the APST command-line client to send this XML to the APST daemon, which proceeds to automatically match up tasks with resources. At that point the daemon manages all of the logistics of application deployment, providing feedback to the user about task completions and generated output files. While this provides a solution for a large portion of challenge (iii) in Section 2.1, it is not sufficient. In the case of EOL, where we typically have millions of tasks running over a period of months, the above scenario would prove to be prohibitive. The situation clearly required the development of a system to manage the overall long-term state of our calculations, automatically generating XML and adding it to APST, and automatically monitoring APST to determine when tasks have been completed in order to store their output in appropriate databases and archives. In other terms, we need a higher level of logistics management that builds on APST and is closer to the application. APST cannot provide such functionality itself as it is designed to be completely application generic. This was our main motivation for developing and higher-level tool, BWMS, described in the next section.

3 Bioinformatics Workflow Management System

The EOL application is a “workflow” application, in the sense that it consists of many software components that have data dependencies (i.e. it is a directed acyclic graph). While APST manages data dependencies automatically, it does not provide any higher-level workflow management capabilities; for example, it does not address application-level abstractions such as the status of genomes or proteins. The creation of flexible systems for the management of complex computational workflows is an important focus in research and in standardization efforts. In the specific context of Grid computing, a Grid Workflow management

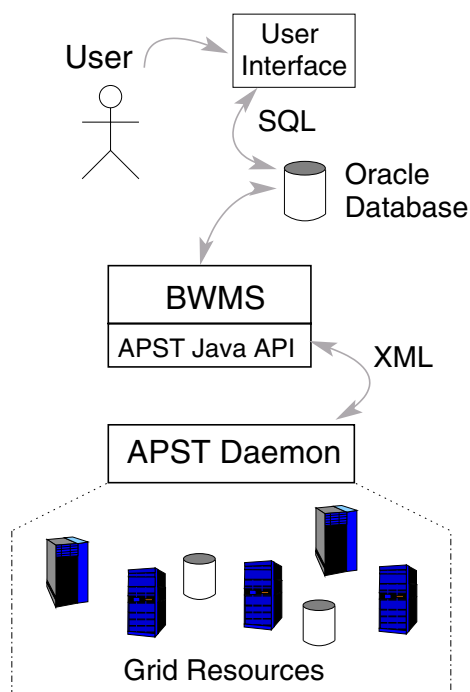


Fig. 3. Bioinformatics Workflow Management System Software Architecture

architecture is currently being developed as part of the Global Grid Forum [29] to define standards that address workflow problems particular to Grid applications. However, no implementation of Grid workflow standards were available at the time of our research. Consequently we chose instead to develop a highly focused workflow system, called the Bioinformatics Workflow Management System (BWMS), that builds on top of APST's distributed resource management capabilities. This integration of BWMS and APST is depicted in Fig. 3.

The overall purpose of the BWMS is to bundle groups of tasks into XML documents for execution by APST. In other words, BWMS translates domain-specific information regarding genomes and proteins, into generic descriptions of tasks that APST is able to execute on remote resources. The overall system state is stored in a relational database, which includes all of the protein sequence information, as well as bookkeeping data that indicates which genomes and proteins have been submitted, are in process, or have been completed by APST. An important advantage of this architecture is that multiple user interfaces may be developed that depend only on the table structure of the database, without having to define or implement wire protocols by which a user interface would have to glean information from the BWMS.

We implemented the BWMS in approximately 5000 lines of Java. Communication with the APST daemon is accomplished using a Java client class included

in the free distribution of APST [4]. We will comment on initial performance results in the following section.

4 Results

We executed a major test of our integrated system at the Supercomputing Conference (SC'03) in November, 2003. We ran the workflow system continuously for approximately 4 days during SC'03, and provided annotation for 36,104 proteins from more than 70 proteomes, with more than 54,000 tasks completed. The computational resources that we used, and the number of tasks completed by each, are depicted in Table 1.

Note the BWMS is able to tolerate considerable heterogeneity in accessing resources. The system used a wide variety of methods to access these machines, which ran a variety of batch schedulers; this heterogeneity was made entirely transparent by APST. The number of CPU's and the wallclock time used on the batch systems indicates the number of CPU's and amount of wallclock time requested in each submission of agents (see Section 2.3). We do not show measures of actual CPU cycles delivered, as many of the resources we used had heterogeneous architectures, thus making comparison difficult. For systems without any batch scheduler (e.g. the various workstations), we achieved a relatively higher throughput per CPU, since we did not have to pay the overhead of waiting through batch queues.

A notable feature of this data is the outstanding performance delivered by the BII Viper Cluster, and the dedicated EOL cluster. This performance is due to the EOL application's exclusive access to these resources during SC'03. Also notable is the relatively poorer utilization of several of the resources, particularly the SDSC Teragrid cluster, and the Titech condor pool. In fact, the SDSC cluster delivered performance comparable to the NCSA cluster, but we often had to manually disable APST's submission to the SDSC cluster to allow other

Table 1. Results for Random Grid/Application pairs

Host	CPU's	Walltime (min)	OS	Scheduler	Access	CPU Hrs
BII Viper Cluster	58	120	Linux	PBS	Globus	6255.3
EOL Cluster	36	120	Linux	PBS	local	3831.3
NCSA Teragrid	32	60	Linux	PBS	Globus	1380.8
UMich Morpheus	16	10	Linux	PBS	SSH	752.1
SDSC Teragrid	32	60	Linux	PBS	Globus	472.2
NBCR Workstation	4	-	Sun	none	SSH	357.5
Titech Condor pool	32	120	Linux	Condor	SSH	340.6
Monash U.	2	-	Linux	none	SSH	152.9
UFCG workstation	1	-	Linux	none	SSH	69.7
BeSC workstation	2	-	Linux	none	SSH	58.1
Total	215	-	-	-	-	13670.5

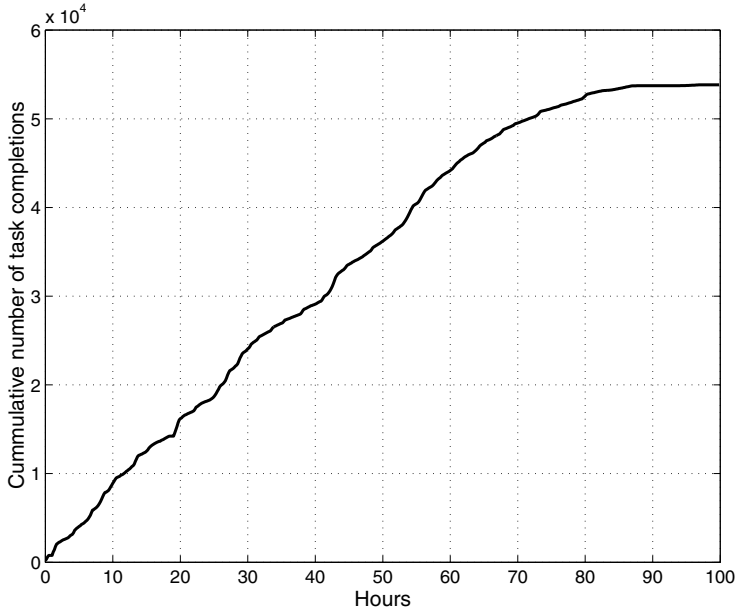


Fig. 4. Cumulative task completions throughout time

applications to access this resource during the conference. For resources with batch schedulers, the amount of wallclock time per RASH request was also a significant factor. As described in Section 2.3, when a request expires, running tasks are abandoned; for shorter requests, a greater fraction of the overall CPU time is lost in this way. Scheduling algorithms for managing the size of these requests, and for allocating tasks to these resources with a limited time duration, are both subjects that merit further study.

The aggregate performance of these resources is depicted in Fig. 4, which shows the cumulative number of tasks completed. While this graph does not exhibit any unexpected behavior, it makes it possible to quantify the throughput of the system in steady-state in terms of task completions: approximately 750 tasks per hour.

The most noteworthy feature of the curve is the plateau effect at the end of the graph. This is due to two factors. First, as we stopped the experiment at the end of the SC'03 conference, we suffered from the common “wait for the last task” syndrome. In the case of EOL this is insignificant as the goal is to achieve high throughput for an application that runs over a long period of time (e.g., weeks). Second, after inspecting the APST logs, we found that the slow down is also due to the increasing workload associated with file transfers. As the jobs completed the APST daemon spent an ever-greater fraction of its time fetching output files. In the long run, file transfer activities may delay the launching of new tasks, as computational resources can not receive needed input files until other file transfers were completed. Since this experiment, we have optimized file

transfers. To reduce the overhead associated with establishing a connection and negotiating security protocols, we now batch multiple files with the UNIX `tar` command, thus decreasing the number of separate transfers.

Another thing we learned during this experiment is that our implementation of the RASH architecture presented in section 2.3 had a scalability problem. Indeed, we used an earlier version in which there was one process on the APST daemon host corresponding to each RASH agent process, which could have totalled to as many as 215 processes (and many more for larger platforms). In our new design, there is only one RASH client process per front-end node, as explained in Section 2.3. We expect this change to lead to significant improvements in performance and scalability of the APST daemon, and thus of the system overall.

In summary, our integration of EOL with APST and BWMS proved to be functional, able to harness a wide range of Grid resources including both batch and interactive systems, and our experimental runs during the SC'03 conference allowed us to identify and address two scalability bottlenecks in the system. We will report on new experiments with this improved implementation in a future paper.

5 Conclusion

In this paper we have presented an integrated software pipeline for deploying the Encyclopedia of Life (EOL) application on large-scale Grid platforms. In our architecture, the Biology Workflow Management System (BWMS) makes use of the AppLeS Parameter Sweep Template (APST) in order to manage the execution of the EOL workflow at the application level. The value of APST is that it provides a simple abstraction layer that it makes the logistics of application deployment and the heterogeneity of Grid resources transparent. In addition, APST uses intelligent scheduling strategies to improve performance (e.g., by improving locality, reducing data movements, avoiding loaded resources). We realized a prototype implementation of this integration and reported on its performance for large runs (more than 54,000 application tasks) over a large-scale Grid platform (10 institutions) during the SC'03 conference. Based on this early experience we identified scalability bottleneck in our system, which we have addressed in the latest version. Our future work is along two fronts. First, we will improve our scheduling and resource management strategies. Indeed, APST's scheduling heuristics were initially developed for completely task parallel applications, and not workflows. For instance, an extremely promising lead is the recent work in [6], which provides new optimality results for the scheduling of multiple workflows in a high-throughput scenario. Second, we will integrate our work with the emerging Grid/Web service architecture and adopt workflow standards as they become available.

Acknowledgements

The authors wish to thank Greg Bruno, Mason Katz, Philip Papadopoulos, Robert W. Byrnes, Greg B. Quinn at SDSC, Atif Shabab, Danny Chuon, Boon

Tuck, Stephen Wong, Yong How Choong, Larry Ang at BII, Toyotaro Suzumura, Kouji Tanaka, Satoshi Matsuoka at TiTech, Colin Enticott, David Abramson at University of Monash, Australia, David R. Simpson, Terence Harmer at Belfast e-science center, UK, Cliane Cristina de Araujo, Zane Cirne at UFCG, Brazil for making the demo at SC'03 a success.

References

1. D. Abramson, J. Giddy, and L. Kotler. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, Cancun, Mexico, pages 520–528, May 2000.
2. S. Agrawal, J. Dongarra, K. Seymour, and S. Vadihyar. *Grid Computing: Making The Global Infrastructure a Reality*, chapter NetSolve: Past, Present, and Future - A Look at a Grid Enabled Server. John Wiley, 2003. Hey, A. and Berman, F. and Fox, G., editors.
3. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke. GridFTP: Protocol Extension to FTP for the Grid, March 2001. Grid Forum Internet-Draft.
4. APST Homepage. <http://grail.sdsc.edu/projects/apst>.
5. C. Baru, R. A. Rajasekar, and M. Wan. The SDSC Storage Resource Broker. In *Proceedings of the CASCON'98 Conference*, November 1998.
6. O. Beaumont, A. Legrand, and Y. Robert. Static scheduling strategies for heterogeneous systems. Technical Report LIP RR-2002-29, École Normale Supérieure, Laboratoire d'Informatique du Parallélisme, July 2002.
7. F. Berman, G. Fox, and T. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley Publishers, Inc., 2003.
8. F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 14(4):369–382, 2003.
9. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.
10. M. 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, October 2001.
11. T.D. Braun, D. Hensgen, R. F. Freund, H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A. Reuther, J.P. Robertson, M.D. Theys, and B. Yao. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837, 2001.
12. R. Buyya, M. Murshed, and D. Abramson. A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas*, June 2002.
13. H. Casanova, T. Bartol, J. Stiles, and F. Berman. Distributing MCell Simulations on the Grid. *International Journal of High Performance Computing Applications (IJHPCA)*, 14(3), 2001.

14. H. Casanova and F. Berman. Parameter Sweeps on the Grid with APST. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. Wiley Publisher, Inc., 2002.
15. H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *Proceedings of the 9th Heterogeneous Computing Workshop (HCW'00)*, Cancun, Mexico, pages 349–363, May 2000.
16. Condor Version 6.2.2 Manual. <http://www.cs.wisc.edu/condor/manual/v6.2/>.
17. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. In *Proceedings of the 10th IEEE Symposium on High-Performance Distributed Computing (HPDC-10)*, August 2001.
18. H. Dail, D. Berman, and H. Casanova. A Decoupled Scheduling Approach for Grid Application Development Environments. *Journal of Parallel and Distributed Computing*, 63(5):505–524, 2003.
19. Elagi. <http://grail.sdsc.edu/projects/elagi/>.
20. EOL Homepage. <http://eol.sdsc.edu/>.
21. I. Foster and C. Kesselman. Globus: A Toolkit-Based Grid Architecture. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 259–278. Morgan Kaufmann, 1999.
22. I. Foster and C. Kesselman, editors. *Grid 2: Blueprint for a New Computing Infrastructure*. M. Kaufmann Publishers, Inc., second edition, 2003.
23. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), 2001.
24. Ganglia. <http://ganglia.sourceforge.net>.
25. Joint Center for Structural Genomics. <http://www.jcsg.org>.
26. Y. Kwok and I. Ahmad. Benchmarking and Comparison of Task Graph Scheduling Algorithms. *Journal of Parallel and Distributed Computing*, 59(3):318–422, 1999.
27. W.W. Li, R.W. Byrnes, J. Hayes, A. Birnbaum, V.M. Reyes, A. Shabab, C. Mosley, D. Perkurowsky, G. Quinn, I. Shindyalov, H. Casanova, L. Ang, F. Berman, P.W. Arzberger, M. Miller, and P.E. Bourne. The Encyclopedia of Life Project: Grid Software and Deployment. *New Generation Computing*, 2004. in press.
28. W.W. Li, G.B. Quinn, N.N. Alexandrov, P.E. Bourne, and I.N. Shindyalov. A comparative proteomics resource: proteins of *Arabidopsis thaliana*. *Genome Biology*, 4(8):R51, 2003.
29. D. Marinescu. A Grid Workflow Management Architecture, August 2002. Global Grid Forum White Paper.
30. National Center for Biotechnology Information. <http://www.ncbi.nlm.nih.gov/>.
31. Open grid service architecture. <http://www.globus.org/ogsa/>.
32. C. Pinchak, P. Lu, and M. Goldenberg. Practical Heterogeneous Placeholder Scheduling in Overlay Metacomputers: Early Experiences. In D.G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, volume 2537, pages 202–225. Springer Verlag, 2002.
33. Pacific Rim Applications and Grid Middleware Assembly. <http://www.pragma-grid.net/>.
34. O. Sievert and H. Casanova. Policies for Swapping MPI Processes. In *Proceedings of the 12th IEEE Symposium on High Performance and Distributed Computing (HPDC-12)*, Seattle, June 2003.

35. V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests. In *Proceedings of the 11th IEEE Symposium on High Performance and Distributed Computing (HPDC-11)*, Edinburgh, 2002.
36. D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid. In F. Berman, A.J.G. Hey, and G. Fox, editors, *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley, 2003.
37. S. Vadhiyar and J. Dongarra. A Performance Oriented Migration Framework for The Grid. In *Proceedings of the 3rd IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, Tokyo, May 2003.
38. R. Wolski, N. Spring, and J. Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, 1999.
39. A. Yarkhan and J. Dongarra. Experiments with Scheduling Using Simulated Annealing in a Grid Environment. In *Proceedings of the 3rd International Workshop on Grid Computing*, Baltimore, 2002.

Genome-Wide Functional Annotation Environment for *Thermus thermophilus* in OBIGrid

Akinobu Fukuzaki^{1,2}, Takeshi Nagashima¹, Kaori Ide^{1,2}, Fumikazu Konishi^{1,2},
Mariko Hatakeyama^{1,2}, Shigeyuki Yokoyama^{2,3,4}, Seiki Kuramitsu^{2,5},
and Akihiko Konagaya^{1,2}

¹ Bioinformatics G., RIKEN GSC, 1-7-22,
Suehiro-cho, Tsurumi, Yokohama, Kanagawa, Japan
{akki, nagashima, kaoide, fumikazu, marikoh, konagaya}@gsc.riken.jp
² RIKEN Harima Inst., 1-1-1, Kouto,
Mikazuki-cho, Sayo-gun, Hyogo, Japan
³ Protein Res. G., RIKEN GSC, 1-7-22,
Suehiro, Tsurumi, Yokohama, Kanagawa, Japan
yokoyama@gsc.riken.jp
⁴ Grad. Sch. of Sci., Univ. of Tokyo.,
7-3-1, Hongoh, Bunkyo-ku, Tokyo, Japan
⁵ Grad. Sch. of Sci., Osaka Univ.,
1-3, Machikaneyama-cho, Toyonaka, Osaka, Japan
kuramitu@bio.sci.osaka-u.ac.jp

Abstract. We developed OBITco (Open BioInformatics *Thermus thermophilus* Cyber Outlet) for gene annotation of *T. thermophilus* HB8 strain. To provide system services for numbers of researchers in the project, we adopted Web based technology and high-level user authentication system with three functions which are rollback function, hierarch representative function and easy-and-systematic annotation. The robust and secure network connection protects the confidential information within the project, thus, researchers can easily access real-time information on DNA sequences, ORF annotations or homology search results. *T. thermophilus* HB8 possesses 2,195 ORFs, 1156 Intergenic regions, 47 putative tRNA regions, and 6 rRNA regions. BLAST against nr/nt database and InterProScan for all ORFs were used to get homology hit records. The system provides an ORF viewer to show basic information of ORFs and database homology hit records. Researchers can update annotation information of ORF by simple operation, and then new annotation is applied to central database in real-time. Latest information can be utilized for lab experiments such as functional analysis, network analysis and structural analysis. The system can be also utilized as data storage/exchange place for the researchers for everyday experiments.

1 Introduction

T. thermophilus is a gram-negative, aerobic thermophilic eubacterium grew at optimum temperatures of 65-85 °C[1]. Addition to thermostable enzyme productions *T. thermophilus* HB8 has been particularly acknowledged as a valuable model organism for a systematical study of protein structures and functions in Structurome project[2]. 1.8 Mb genome of *T. thermophilus* has sequenced, and it is estimated to encode ca. 2000 proteins. This small genome size and excellent crystallizability of the protein are most suitable to characterize a whole organism as a single system operated by genes and proteins. At present, more than 20 organizations are involved in the studies of protein structure elucidation, proteome, microarray and metabolome analysis in the Structurome project. For this purpose, construction of a common secured working environment is necessary to centralize and analyze vast datasets for the project researchers located in various institutions. Based on this policy, we developed OBITco: Open BioInformatics *T. thermophilus* Cyber Outlet for genome annotation and functional analysis of system wide study of *T. thermophilus*. OBITco enables the researchers broad genome annotation with gene prediction, structure elucidation, metabolic pathway analysis, gene expression analysis and network analysis.

2 System Architecture

The system architecture of OBITco aims at information sharing and genome wide functional annotation for the project. The gene annotation is done by sequence homology search of DNA and proteins. Since more than one researcher are assigned for the annotation, all of the opinions do not always match. And, information should be always updated according to the public database update. To solve such problems, OBITco has a rollback function for annotation information and a hierarchical representative function.

2.1 Rollback Function

The rollback function is a function to return to the past information, in other words, a function to record a tracking. We can reconsider the divisive opinion from the past annotation records. Such a function bestows diversified views and makes it possible to record an intellectual new idea. (Fig. 1)

2.2 Hierarchical Representative Function

A hierarchical representative function allows the expansion of the information for annotation with an inheritance of structure by an object-oriented approach. The annotation system must gather the various information such as a genome sequence, ORF information and information on the homologs. So there is a need for high-speed bioinformatics analysis to generate the data from such information. This system store information which can normalize genome sequence and

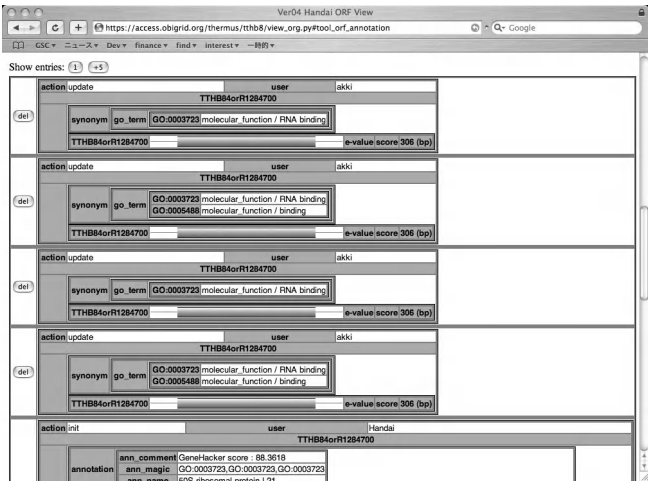


Fig. 1. Screen shot of the rollback function. The past annotation records are iterate in counter chronological order

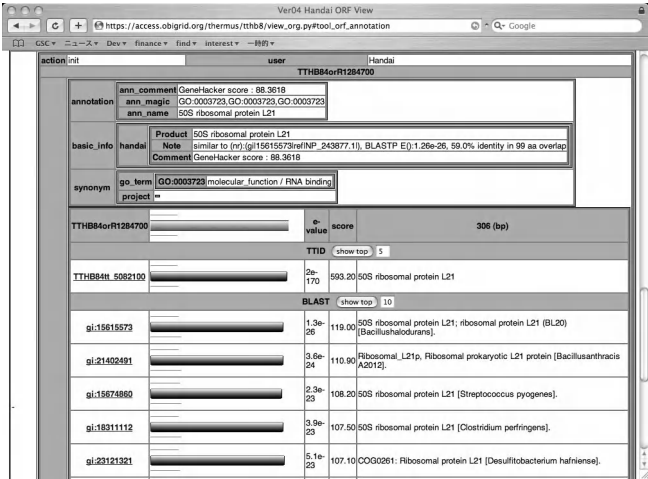


Fig. 2. A screen shot of the hierarchical representative function. One hierarchical object contains variety annotation data. The view of the annotation object is automatically dominated by hierarchical representative function. For example, homology hit data is shown with a graphical alignment of homology

ORF position into relational database (RDB). And the annotation system keeps results of bioinformatics analysis and annotation histories into a dictionary structure object, and stores the object into the RDB tables. Therefore, the annotation can be done with high flexibility and high speed. (Fig. 2)

2.3 Easy and Systematic Annotation

Many annotation data can be given to the regions by adopting the hierarchical representative function. However, on the other hand, the wide variety of information handling tends to cause a nonuniformity in the annotation quality and to force users heavy operational loads when one handles many annotation regions at the same time (Fig. 3(A)). To reduce number of operational loads and to facilitate easy and systematic procedures, a concept of Workflow is introduced and implemented into the annotation system. The Workflow bestows a series of simple operation interface that focused on individual annotation themes (Fig. 3(B)). It has three phases: region searching phase, region annotation phase and consum-

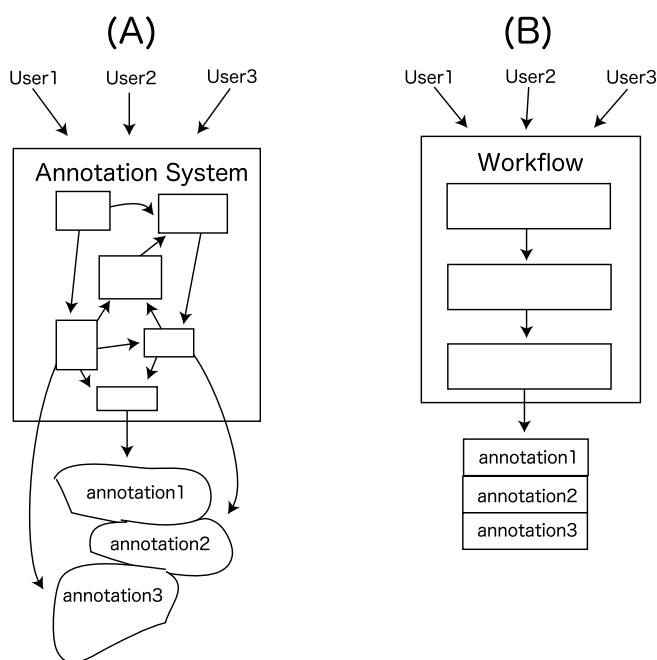


Fig. 3. When users make annotation with general annotation system, they would be face to too much operational load. Choosing an affirmative data field to enter new information would cause miss-input of data, or discriminating of an applicable information to refer would slow the progress of annotation. Thus, their output would have a fluctuation of annotation quality and the lead-time for annotation would be longer. (A) Workflow provides a well-regulated set of user interfaces that give finite information to users, thus, the system realizes a reducing of operation load and regulation of annotation quality. (B)

mating phase. Region searching phase bestows user interface to search genome regions by keyword, genome locus or its homolog entries, then put them to a region list. At region annotation phase, annotation user interface repeats over

all regions in the region list. Then finally, consummating phase bestows a table of summary of whole annotations for this task. For example, on annotation for PDB information, *T. thermophilus* HB8 putative ORFs are blasted against known PDB entries to append an annotation information for protein structures. First phase of the Workflow, researcher searches regions that have over 90% se-

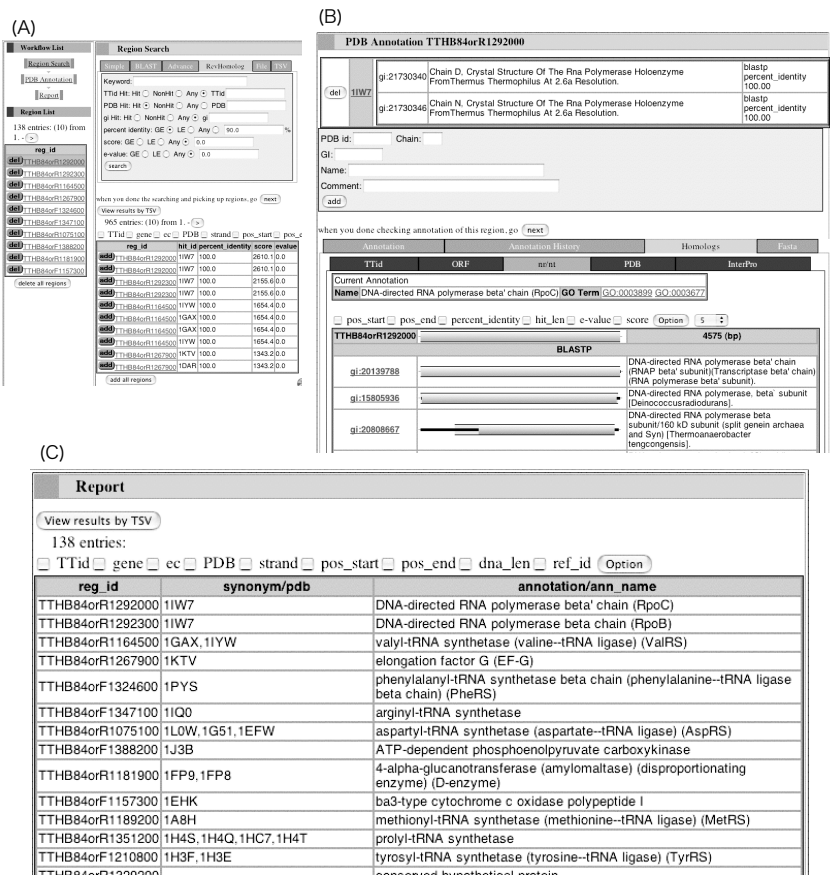


Fig. 4. A screen shot of the PDB annotation. (A)Workflow List shows flow of annotation task, they are Region Search as region searching phase, PDB Annotation as region annotation phase and Report as consummating phase. Region List shows a table of region list which selected to annotate in this task. Region Search is graphical user interface to search regions by keyword, sequence homology or type of homology hit. This example shows the condition to find regions which have over 90% sequence identity homolog. There are 965 homolog hits in this condition (B)PDB annotation is to append the PDB ID and its chain name, gene id, name and comment to the region. One region could make several protein structure, this example shows that the region has 4 structures 1H4Q, 1H4S, 1H4T and 1HC7 (C)At consummating phase, all region ID and its annotated PDB are shown as table

quence identity in PDB entries by RevHomolog search interface, then append it to the region list (Fig. 4(A)). Second phase of the Workflow, annotation interface shows homologies and current annotations for the first region of the region list. A researcher refers these information to select the best PDB information. The annotation interface that specialized to PDB annotation helps the researcher to register annotation data correctly. When the researcher confirmed a registration of new annotation, she/he goes to next region (Fig. 4(B)). A user interface which has optimum and simple operation allows the researcher to annotate adequately and speedily. Finally, the report interface shows the latest annotation list for all regions in the region list, so researcher can confirm annotations at a glance (Fig. 4(C)). Thus, Workflow realizes the easy and systematic annotation.

3 Implementation

We chose Python (<http://www.python.org>), an object oriented language, for the system development. Python is a highlevel, dynamic, general purpose programming language that can be applied to different problems in bioinformatics study. And PostgreSQL database is also employed as a backend subsystem for data access. PostgreSQL is an opensource, a descendant of the original Berkeley code. PostgreSQL supports SQL92 and SQL99 and offers various modern features (complex queries, foreign keys, triggers, views, transactional integrity, multiversion and concurrency control). All these systems are constructed in the OBIGrid environment. The annotation system is composed of several API sets which are described with Python and provides service for the user by CGI. Before to develop OBITco, the genome database software Ensembl[3] has tested. Ensembl has implemented with four layer architecture which are viewer, biologically meaningful objects, database connectivity objects and data sources. This architecture makes it easier to evolve the schema to address new data types. However, *T. thermophilus* has circular chromosomes, so several modifications

View	View Composition API		
	HTML Rendering Object		
Control	Workflow API		
Model	Data access API		
	User Object	Annotation Object	Region Object

Fig. 5. OBITco three layer architecture. The model layer has Data access API which contains User Object to control user access, Annotation Object to handle the annotation information, and Region Object is representative for regions to annotate. Control layer is implements as Workflow API. The view layer has HTML Rendering Object to make graphical user interface within View Composition API

Table 1. OBITco implemented classes

View	
Component.py	Component root class
Component_SubComponentWrapper.py	Make SubComponent class as Component
Component_ToolGeneralAnnotation.py	For annotation
Component_ToolRegionView.py	For region view
Component_ToolReportTable.py	For report table
Component_ToolSearchRegion.py	For region search
SubComponent_AnnoobjectAnnotation.py	For annotation of annotation object
SubComponent_AnnoobjectSynonymGo.py	For synonym/go.term of annotation object
SubComponent_AnnoobjectSynonymPDB.py	For synonym/pdb of annotation object
SubComponent_AnnoobjectSynonymProject.py	For synonym/project of annotation object
SubComponent_BasicInformation.py	Show basic information of the region
SubComponent_CommonObject.py	Show table view of general object
SubComponent_Entries.py	Table view for entries such as search result
SubComponent_FileUpload.py	User interface to handle file uploading
SubComponent_GeneralGenomeBrowse.py	View of genome wide browse
SubComponent_GeneralHomologs.py	Table of homologies
SubComponent_GeneralRegionFasta.py	To show FASTA format data
SubComponent_GeneralRegionView.py	To show sequence level region view
SubComponent_GeneralSearch.py	Root class for searching interface
SubComponent_SearchAdvance.py	For advanced keyword search
SubComponent_SearchBlast.py	For blast base search
SubComponent_SearchFile.py	To search regions in uploaded file
SubComponent_SearchRevHomolog.py	To search regions in homolog condition
SubComponent_SearchSimple.py	Simple keyword search
SubComponent_Tab.py	To make tab sub component
PObject.py	Base classes to compose user interface parts
Control	
WorkFlow.py	WorkFlow framework
Viewer.py	WorkFlow framework but one step operation
SessionParams.py	CGI parametors and session management
Model	
DataConnection.py	API for data access
etc	
DataFetch.py	To fetch the genbank entry data
OBIShare.py	Shared sub routines
ObjectDict.py	General object operation
workflow001.py	CGI script for functional annotation
workflow002.py	CGI script for go term annotation
workflow003.py	CGI script for PDB annotation
workflow004.py	CGI script for project ID annotation
view.py	CGI script for viewer

on source code was required. In addition, the data set of *T. thermophilus* was not fixed yet, therefore it is necessary to be annotated by many researchers in the project. The system architecture and API sets are designed to capsule the data access and to be task flow oriented system. The API sets are categorized by its role into three layers, data access, view composition and work flow. (Fig. 5) Table 1 shows implemented classes on OBITco. View Composition API is coded as 26 python script files. There are three amass of files Components, SubComponents and PObjects. A Component makes one operation in the Workflow as like a one web page. Component class is directly interacting with Workflow API to complete the chain of operations on the annotation task. The Workflow requires the management of region list, the dominating of operation flow and the handling of CGI parameters to the component. A SubComponent makes more tangible parts of user interface. SubComponent_AnnoobjectAnnotation bestows a composition of a user interface and Data access API call to update the functional annotation of the region. SubComponent_AnnoobjectSynonymGo, SubComponent_AnnoobjectSynonymPDB and SubComponent_AnnoobjectSynonymProject bestow a composition of user interface

and data access to add and delete synonym terms for GO term, PDB and project original region ID. `SubComponent_BasicInformation` bestows fixed format table of basic information for the region such as region ID, genome locus, functional annotation and synonyms. `SubComponent_CommonObject` is for a basis class for object view. `SubComponent_Entries` bestows `Entries` classes to compose the table view for list of search results, homolog hits or so. `Entries` class contains inner operation for entries object which handles the list of result or homolog entries and its property to control paging or sorting of the list. `Entries` class completely include the method of composition and operation for entries object, thus, `Component` and `SubComponent` don't need to take care about entries inside. Because, the interaction between `Entries` class and `Component/SubComponent` is defined by `Workflow`. `SubComponent_FileUpload` bestows a framework to accept tab separated value type file uploading. File uploading corrects a list of region IDs from the file instead of searching. This mechanism allows the researcher to make region list from the file which exported by other analysis tools. `SubComponent_GeneralGenomeBrowse` bestows the genome browse interface which shows regions loci on entire genome. `SubComponent_GeneralHomologs` is a class to show homologs which hit to the region. This class extends the `Entries` class to realizes graphical alignment view. `SubComponent_GeneralRegionFasta` shows FASTA format sequence of the region for DNA sequence and Amino Acid sequence. `SubComponent_GeneralRegionView` bestows sequence level view of the region. The researcher needs sequences information on the region or its UTRs to confirm its transcription or translation. `SubComponent_GeneralSearch` is for basis class of `SubComponent_Search` series to bestows region searching interface. `SubComponent_Tab` bestows the view which contains a row of tabs that gives the appearance of folder tabs above the content area. `PObject` bestows base classes to compose user interface parts such as text field, button, check box, radio button and or so. `Workflow API` is coded as 3 python script files. `Workflow` is a `Workflow` framework class to control whole task flow of the annotation. `Viewer` is a `Workflow` framework class but without workflow control. `OBITco` uses `Viewer` class to realizes general information viewer. `SessionParams` is a management center of CGI session parameters. Data access API is coded by a python script `DataConnection`. `DataConnection` is a class which includes the set of data access method. Any data access procedure pass through this class, thus, `View` and `Control` layer is completely separated from back end data base system. `OBITco` genome wide annotation system is developed by these internal API sets.

4 Summary

The breakdown of current annotations is shown on table 2. The annotation means functional annotation's update. The synonym means there were adding or deleting of synonym entry. The computational annotation is done before the annotation by researcher. The researcher requires to make annotation only if original annotation might be wrong or there are additional information, thus, a number of annotation by researchers are getting small. Therefore, synonym/pdb

Table 2. Latest annotation counts

annotation target	count
annotation	25
synonym/gene	67
synonym/ec	46
synonym/go_term	2
synonym/pdb	80
synonym/project	2097

Table 3. Aspect of OBITco registered user

organization	sites	users
Institute	2	39
University	7	13
Company	2	2
Org.	1	1
	12	55

has no precedent computational annotation, so a number of annotation by researcher for it has a larger count. OBITco current version which includes Workflow concept first time has been released at February, 2004. We have observed users access and maintained for minor version up. Table 3 shows an aspect of registered user and its organization. Organizations are calculated by their registered e-mail address domain name. All of users were asked to agree with non disclosure contract for our database which includes non published materials. When the materials are ready to publish, OBITco will be open to public. No any registration would be required. But, there are some problem to be public. We should also consider about technical matters of public service. The very clear matter is computational power. OBITco serves many services which based on database and homology searching. These programs need a lot of cpu power and memory spaces. So, we will use a grid technology to solve this problem. OBITco is a framework utilized not only for genome annotation but also for candidate selection for structure determination, function prediction of hypothetical proteins based on the information such as protein structure and mRNA expression profiles. The function to handle these data would be available on the future release.

References

1. Borges, K.M. and Bergquist, P.L. (1993) Genomic restriction map of the extremely thermophilic bacterium *Thermus thermophilus* HB8. J. Bacteriol., 175, 103–110.
2. Yokoyama, S., Hirota, H., Kigawa, T., Yabuki, T., Shirouzu, M., Terada, T., Ito, Y., Matsuo, Y., Kuroda, Y., Nishimura, Y., Kyogoku, Y., Miki, K., Masui, R. and Kuramitsu, S. (2000) Structural genomics projects in Japan. Nat. Struct. Biol. 7 Suppl:943–945.

3. T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik1 and M. Clamp The Ensembl genome database project. (2002) *Nucleic Acids Research* 30: 38–41
4. Shibuya. T. and Rigoutsos, I. Dictionary-driven prokaryotic gene finding. (2002) *Nucleic Acids Res.* 30, 2710–2725.
5. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410
6. Sonnhammer, E.L.L., von Heijne, G. and Krogh, A. (1998) A hidden Markov model for predicting transmembrane helices in protein sequences. In Glasgow, J., Littlejohn, T., Major, R., Lathrop, F., Sankoff, D. and Sensen, C. (eds), *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, CA, pp. 175–182.
7. Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H. and Kanehisa M. (1999) KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.* 27, 29–34.
8. Schomburg, I., Chang, A. and Schomburg, D. (2002) BRENDA, enzyme data and metabolic information. *Nucleic Acids Res.* 30, 47–49.
9. Pearson, W.R., Wood, T., Zhang, Z. and Miller, W. (1997) Comparison of DNA sequences with protein sequences. *Genomics*, 46, 24–36.
10. Hirokawa, T., Boon-Chieng, S. and Mitaku, S. (1998) SOSUI: classification and secondary structure prediction system for membrane proteins. *Bioinformatics*, 14, 378–379.
11. Apweiler, R., Attwood, T.K., Bairoch, A., Bateman, A., Birney, E., Biswas, M., Bucher, P., Cerutti, L., Corpet, F., Croning, M.D. et al. (2000) InterPro—an integrated documentation resource for protein families, domains and functional sites. *Bioinformatics*, 16, 1145–1150.
12. Zdobnov, E.M. and Apweiler, R. (2001) InterProScan—an integration platform for the signature-recognition methods in InterPro. *Bioinformatics*, 17, 847–848.
13. Kimura, S., Hatakeyama, M., Konagaya, A., (2004) Inference of S-system Models of Genetic Networks from Noisy Time-series Data. *Chem-Bio Informatics Journal*, Vol.4, No.1.

Parallel Artificial Intelligence Hybrid Framework for Protein Classification

Martin Chew Wooi Keat¹, Rosni Abdullah², and Rosalina Abdul Salam³

¹ Faculty of Computer Science,
Universiti Sains Malaysia, Penang, Malaysia
`martin.wooi.keat.chew@intel.com`

² Faculty of Computer Science,
Universiti Sains Malaysia, Penang, Malaysia
`rosni@cs.usm.my`

³ Faculty of Computer Science,
Universiti Sains Malaysia, Penang, Malaysia
`rosalina@cs.usm.my`

Abstract. Proteins are classified into families based on structural or functional similarities. Artificial intelligence methods such as Hidden Markov Models, Neural Networks and Fuzzy Logic have been used individually in the field of bioinformatics for tasks such as protein classification and microarray data analysis. We integrate these three methods into a protein classification system for the purpose of drug target identification. Through integration, the strengths of each method can be harnessed as one, and their weaknesses compensated. Artificial intelligence methods are more flexible than traditional multiple alignment methods, and hence, offers greater problem-solving potential.

1 Introduction

This concept paper relates to the field of protein classification, for the purpose of structural and functional determination, in order to assist the process of drug target discovery. Given an unlabeled protein sequence S and a known superfamily F , we wish to determine whether or not S belongs to F . We refer to F as the target class and the set of sequences not in F as the non-target class. In general, a superfamily is a group of proteins that share similarities in structure and/or function. If the unlabeled sequence S is detected to belong to F , then one can infer the structure and function of S . This process is important in many aspects of bioinformatics. For example, in drug discovery, if sequence S is obtained from some disease X and it is determined that S belongs to the superfamily F , then one may try a combination of existing drugs for F to treat the disease X . Our approach combines three methods (Hidden Markov Model, Neural Network and Fuzzy Logic) into a single integrated hybrid system. Furthermore, the core of this system (the Neural Network portion) can be parallelized for better performance, especially when very large data sizes have to be processed. We begin by

reviewing prior work done using Hidden Markov Models, Neural Network and Fuzzy Logic in the field of bioinformatics. The paper first looks at a review of existing protein classification techniques and their limitations. It then proposes a new classification method, based on a hybrid of existing techniques. The theoretical workings of the conceptual system are explained, its benefits anticipated, and its contributions to the field of bioinformatics outlined. The paper ends with a discussion on future work.

2 Literature Review

There are two types of protein sequence analysis tools: (a) alignment tools, and (b) consensus identification tools. The purpose of alignment tools is to optimally align two or more protein sequences. This serves to help identify sections of similarities across the sequences presented to the tool. An alignment tool does not make any decisions about the relatedness of the sequences, functional or otherwise, other than highlighting sections of similarities among the sequences presented to it. One example of an alignment tool is BLAST [1]. Tools such as BLAST are based on conventional string processing algorithms, and makes use of insertions of gaps, in order to derive a more optimal alignment. Other than string processing algorithms, multiple alignment tools based on other techniques such as Genetic Algorithms have also been developed. An example of this is SAGA (Sequence Alignment by Genetic Algorithm) [2]. A major drawback of such multiple alignment tools is the arbitrary scoring or fitness function employed to rank the optimality of a particular alignment. Furthermore, when a very large number of sequences are processed, the resulting multiple alignment could potentially be rather taxing for the human eye to interpret. Hence, multiple alignment tools may not be very suitable to be used to determine the relatedness of an unknown sequence with respect to a large set of sequences.

Consensus identification tools are used to determine the relatedness of a given sequence, with respect to a particular cluster of sequences. First, the cluster has to be abstracted into a model or a representation. Once this is done, an unknown sequence is compared against the model/representation, in order to determine how well the fit is. An unknown sequence may be compared to a range of models/representations, in order to ascertain the best fit. A commonly used modeling/representation technique is Hidden Markov Model [3]. A Hidden Markov Model is built to represent protein motifs (recurrent substring patterns in a protein sequence string), rather than the entire protein sequence. Related proteins normally share similar, or near-similar motifs. Hidden Markov Models are able to cope with noises in the motif patterns with the use of insert or delete states. As a result of this, Hidden Markov Models are able to handle near-similarities very well. When an unknown sequence is submitted to the Hidden Markov Model, the model is able to return a probability value, indicating the consensus of the unknown sequence, with respect to the cluster of sequences the model represents. A major drawback of Hidden Markov Models is that, if the relationship among the sequences are not explicit (or near explicit), we would

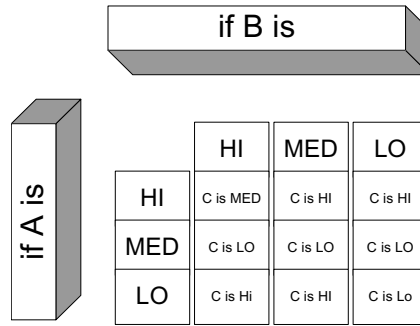
not be able to construct an accurate model. Hence, Hidden Markov Models are suitable for coping with explicit, or near explicit (i.e. noisy) similarities, but not suitable for implicit similarities.

To cope with implicit similarities, neural networks are used [4]. A sequence is transformed into a set of input values, based on a particular encoding scheme (e.g. 2-gram encoding scheme which counts the frequency of unique character pairs in the protein sequence). These input values are then fed into a neural network. A typical usage model requires a network to be trained on a collection of data pre-determined to be of similar functionality. Once the properties of the collection have been abstracted by the network, it is ready to be used in a predictive mode. An unknown sequence is encoded using the same encoding scheme that the network was trained on, and then fed into the network. A positive output would indicate that the unknown sequence is in consensus with the collection. The drawback of encoding schemes, such as 2-gram encoding, is that local similarities (e.g. motif information) could be lost during the transformation. Regular expression based functions has been used to detect local similarities and return a real value, to be used as a bias input to the neural network. However, such an approach still relies on arbitrary score assignments, and could result in a potentially misleading bias.

Hidden Markov Models and neural networks have been combined before into a hybrid to analyze DNA sequences. One example searches for potential TATA-boxes in a DNA sequence [5]. A TATA-box is a sequence rich in adenine (A) and thymine (T), located upstream of a gene. The TATA-box acts as a promoter. A promoter is a DNA sequence that enables a gene to be transcribed. The promoter is recognized by RNA, which then initiates transcription (i.e. generation of protein). TATA-boxes vary in structure. To detect potential TATA-boxes, hidden Markov models are used to abstract known TATA-boxes found along a particular DNA sequence. One model per box. A neural network is then used to abstract the dependencies across the models. The outputs of the hidden Markov models are used as the inputs into the neural network. This hybrid is then used to analyze DNA sequences to detect potential TATA-boxes. The presence of TATA-boxes indicates the presence of genes in the DNA sequence.

Differing from the three categories of applications mentioned above, is a fourth category consisting of applications which uses Fuzzy Logic to abstract microarray data for prediction purposes [6]. Microarrays are genes inlaid on a piece of silicon, and stained to highlight the presence of selected genes. These genes govern the production of proteins. A sequence of microarrays, illustrating gene expression over a period of time, is abstracted using Fuzzy Logic. Once the knowledge base is built, the system is then used to anticipate which proteins might be produced, based on incomplete gene expression information.

We can derive numerous decision matrixes from microarray observations such as the example in Figure 1. From these decision matrixes, when we are presented with a situation where we only know about the quantity of a limited number of elements (A & B only for example), we would be able to extrapolate additional information about other elements (such as the quantity of element C).



	if B is		
if A is	HI	MED	LO
	HI	C is MED	C is HI
	MED	C is LO	C is LO
	LO	C is HI	C is LO

Fig. 1. Decision matrix derived from knowledge base

We propose a hybrid protein classification and consensus identification system, which will combine the best features of some of the methods mentioned above. The core of our system is an array of neural networks, but we use Hidden Markov Models to compensate for the lost of local similarity information. Hidden Markov Models are based on probability theory and is less arbitrary than regular expression functions. The outputs from the array of neural networks are then post-processed using Fuzzy Logic in order to yield new information. The neural network core of the system easily lends itself to parallelism. Parallelization is important in order for the system to cope with very large data sizes. We have identified two approaches used to parallelize neural networks: (a) parallel neural network execution environment [8], and (b) parallel neural network programming language [9].

For the parallel neural network execution environment approach, it is the general case that neural network algorithms share similar phases of execution. For example, loading of patterns, propagation, error calculation, weight correction, and so on. A set of standard function headers, governing the operation of a neural network, is defined, as well as the mode of interaction among those functions. An end-user has to “override” those functions, to fit the problem being solved. Once this is done, the execution environment will run those functions in the correct order, and automatically parallelize the functions which have been identified to be parallelizable. The drawback of this approach is that the development of the application is constrained by the pre-defined function headers. This inflexibility might impact the efficiency of the application, if not outright impede its development.

For the parallel neural network programming language approach, a programming language specialized for the description of neural networks and its parallel operation is used. Such a language contains specialized constructs to describe nodes and connections, as well as parallel operations. The drawback of this approach is that such languages are relatively obscure, and hence, has limited libraries, as well as being unable to access libraries written in more popular languages.

We intend to develop our system using parallel libraries, such as Message Passing Interface (MPI), which is available for popular languages such as C++.

Low-level libraries such as MPI will give better speedup performance, and C++ has re-usable libraries, such as Standard Template Library (STL) which will help speed up implementation. More details on our system will be provided in the next section.

3 System Description

The system first has to be trained to abstract the properties of as wide a range of protein superfamilies as possible. Each protein superfamily is represented by its own neural network. A protein sequence in a particular protein superfamily must first be encoded to yield an array of real values (e.g. between 0.0000000000000000 to 1.0000000000000000). This array of real values is an abstraction of the properties of that particular protein sequence, brought to surface by the encoding scheme used.

A protein consists of a string of amino acids. Up to 20 unique amino acids (each represented by a unique character) may make up a protein sequence. Hence, a protein sequence is represented as a string of characters. An encoding scheme is a function which takes in a string of characters (i.e. protein sequence) and returns an array of real values based on a pre-determined algorithm. An encoding scheme which may be good at abstracting global similarities may not be able to adequately cope with local similarities.

Global similarity refers to similarity observed when the entire protein sequence is taken into account. Local similarity refers to similarity observed at only a certain section of the protein sequence. An example of an encoding scheme which captures global similarity is 2-gram encoding. 2-gram encoding captures the frequency of unique character (i.e. amino acid) pairs in the protein sequence. The various frequency values are normalized to be between 0 to 1. To overcome the problem of inadequate abstraction, multiple encoding schemes are used, and their outputs made coherent through normalization.

For abstracting local similarity, we propose the use of Hidden Markov Models. One model for one particular local similarity. Local similarity can be observable with the aid of multiple alignment tools. Once a protein superfamily is multiply-aligned, a section of interest is identified and delineated. A Hidden Markov Model is then generated to represent this window.

To train a particular neural network, the protein sequences in a superfamily are each encoded using one or more global similarity encoding schemes, and is window-scanned against one or more local similarity Hidden Markov Models. During window-scanning against a particular Hidden Markov Model, the highest probability value returned is used as the indicative value.

The real values returned from the various schemes (i.e. encoding and Markov) are normalized according to a pre-determined algorithm, in order to prevent over-bias in the network training. Once the values are normalized, they are fed into the neural network. The neural network has an input layer, a hidden layer, and a single output node. This single output node is defaulted to 0 (i.e. false). The purpose of the training is to condition the network to respond with an output

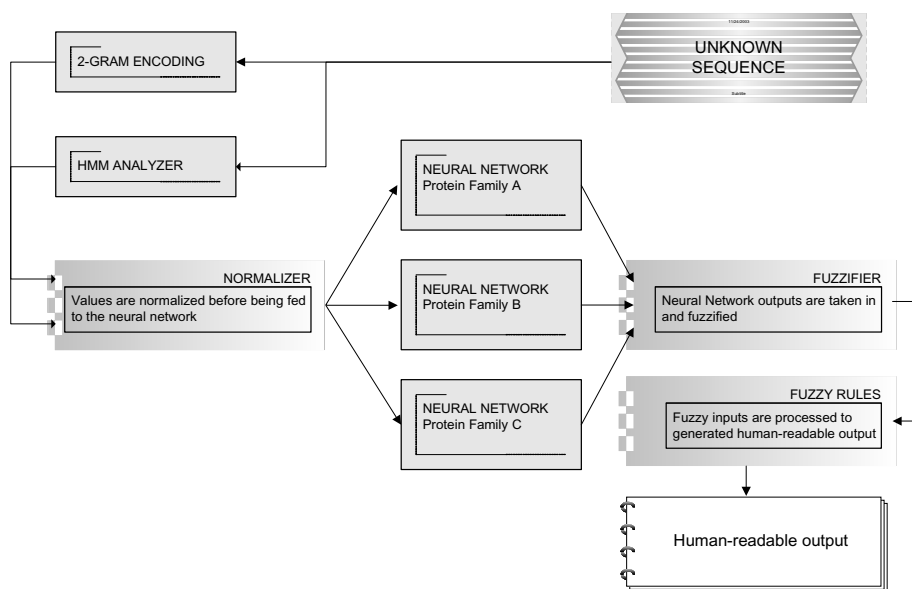


Fig. 2. Schematic illustration of the hybrid system

value of 1 (i.e. true), or as close to 1 as possible, whenever a sequence of similar characteristics (with respect to the combination of encoding and Markov schemes used) is presented to it.

Once each protein superfamily is abstracted to its own neural network, the system is ready to analyze unknown sequences. The unknown sequence is first transformed into a set of real values inputs, using the same combination of encoding and Markov schemes that was used to train the array of neural networks. Once transformed, the values are fed into each neural network. The neural network returning the highest value (i.e. nearest to 1) is deemed to be in resonance with the unknown sequence. As such, the unknown sequence is considered to exhibit the same structural/functional characteristics of the protein superfamily represented by the resonant neural network. A schematic of the system described above is given in Figure 2.

However, the output of the other neural networks could also yield potentially useful information. To analyze these residue data, fuzzy logic is used. The outputs from the neural networks are arranged in a matrix (refer to Figure 3), and then fuzzified based on pre-determined membership classes. When a particular unknown sequence S_i (refer to Figure 3) is submitted for analysis, it will generate an array of signals of varying intensities (HI, MED, LO). This “fuzzy signature” can be compared to the “fuzzy signature” of other known sequences. The purpose of this comparison is to determine which known sequence, or combination of sequences, is best able to mimic the functionality of the unknown sequence. Such information may be useful for drug synthesis and drug substitution purposes, as well as drug side effect analysis.

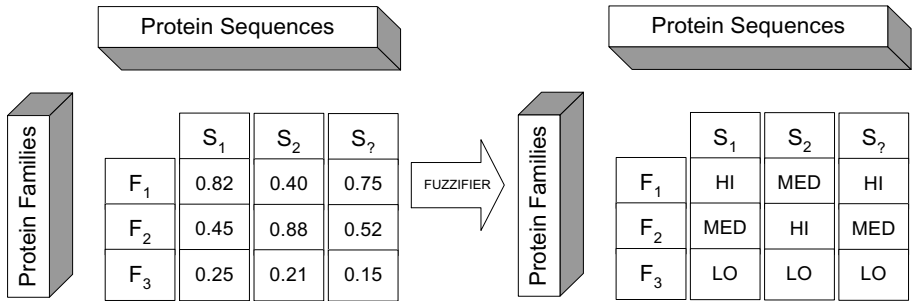


Fig. 3. Schematic illustration of the fuzzifier process

This system could be used to organize existing protein data in a manner which will enable structural/functional inferences to be made on unknown, as well as synthetic sequences. This will assist the process of drug target discovery, as well as drug synthesis. A drug target could be patented and licensed to drug manufacturers.

Experimental variations can be easily introduced into this framework to determine if any new knowledge could be generated. For example, a particular protein family (i.e. neural network) could be represented by its own unique encoding technique, or may share an encoding technique with other families. A particular protein family will also be represented by hidden Markov models based on motifs unique to that family. The general idea is that whichever encoding method and hidden Markov model used for a particular family must be able to abstract the defining properties of that family. When an unknown protein sequence is presented to a particular neural network, it is encoded using the same method that was used to encode the data that the neural network was trained on. Furthermore, that unknown sequence is window-scanned with respect to the hidden Markov model accompanying that neural network, and the highest probability value obtained is used.

Considering the very large protein data size that the system has to process, parallel processing offers an effective way to address computation performance issues. Parallel processing can most contribute to performance improvement in the neural network training section of the system. Neural network training is time consuming especially when the training data set is very large. The neural network used by this system has one input layer, one hidden layer, and a single output node (refer to Figure 4).

To parallelize the training of the neural network, independent computation paths are first identified. For example, the lines of computation shown in Figure 5 are independent of the lines of computation shown in Figure 6. The only commonalities are the input values and final output value.

The commonalities (i.e. the input values and the final output value) will be handled by the master computer. The independent paths of computation will be distributed to worker computers, to be processed in parallel. The system will alternate between a sequential state and a parallel state. The initial state will

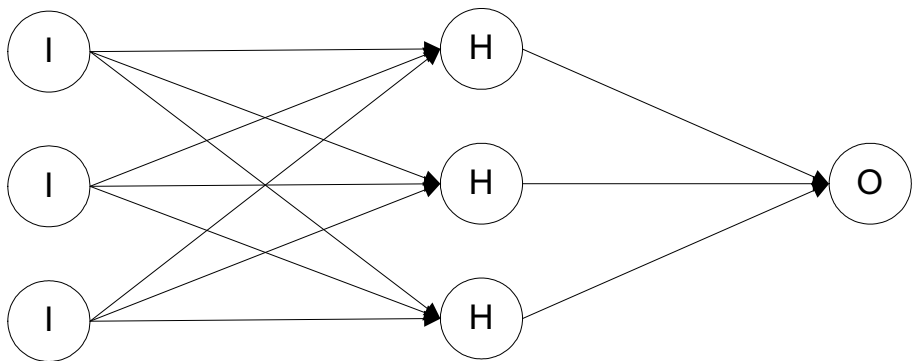


Fig. 4. Neural network with one input layer, one hidden layer, and one output node

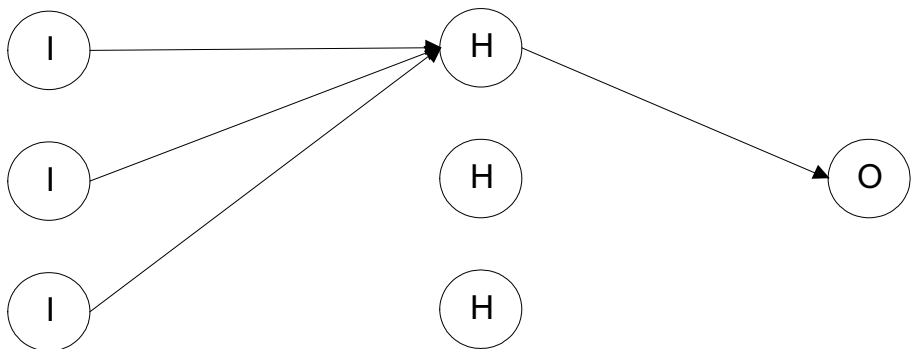


Fig. 5. Example of an independent computation path

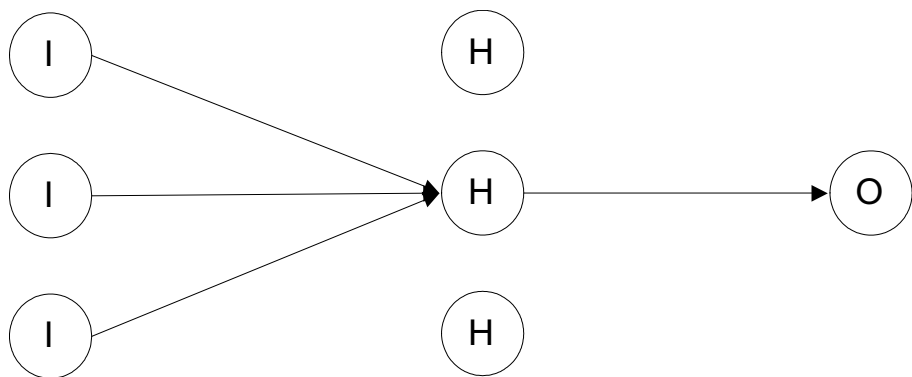


Fig. 6. Another example of an independent computation path

be sequential, as the input values are gathered, and distributed to the worker computers by the master computer. Once this is done, each worker computer

performs its assigned path of computation independently, and in parallel. The master computer then gathers the results from each worker, and sums up into an output value. This cycle repeats itself, once for each training datum. Protein consists of up to 23 different amino acids, and with 2-gram encoding, we will have 529 (23×23) nodes for both the input and hidden layers. The number of nodes, as well as connection weights linking those nodes will increase exponentially if we move on to 3-gram encoding ($23 \times 23 \times 23$). Furthermore, the system has an array of neural networks, one for each protein family. The training of one neural network is independent of the other, and hence, could be carried out in parallel as well. Therefore, parallel processing is necessary, not only because of large protein data sizes, but also, due to the size of the neural network as well. Parallel processing will enable this system to be scalable, not only in terms of data size, but also, in terms of complexity.

4 Anticipated Advantages Over Current Work

Currently, alignment tools are commonly used to help determine homology among sequences [1], [2]. Homology among proteins is assumed to be a result of those proteins having a common evolutionary origin. The main drawback of current alignment tools is that they are based on the identity (i.e. $A = A$, $B = B$, etc.) concept of similarity, or the “point accepted mutation” concept of similarity (i.e. $A = A$ and $A = B$, $C = C$ and $C = D$, etc.). Gaps have to be artificially inserted to obtain a subjectively defined optimal standard based on arbitrarily assigned scores and penalties. Furthermore, when the number of sequences that has to be compared is very large, the presentation of the end result may be humanly indecipherable. Our proposed hybrid system supports multiple encoding schemes for global similarity abstraction. Therefore, it is not restricted to only one concept of similarity. With the use of Hidden Markov Models, we replace arbitrary scoring with the concept of probability, for a more realistic assessment of local similarity. The overall system does not require artificial gaps to be inserted. Furthermore, with post-processing done by the fuzzy logic component of the system, we will be able to filter the results of the analysis into a more humanly decipherable summary. Systems based on Hidden Markov Models have been used to represent sequence collections before [3]. However, the main drawback is that it relies on explicit similarity. If the similarity of the sequences is implicit, Hidden Markov Models would not be able to model the collection. Our proposed hybrid system supports encoding schemes which will be able to extract implicit similarities, therefore, overcoming the limitation faced by Hidden Markov Models systems. Neural Networks have been used to classify proteins before [4]. However, to capture local similarity, a regular expression based scoring function was used. The drawback of this is that (a) regular expressions merely match a string with another without indicating the probability of the consensus, and (b) arbitrary scoring is involved. Furthermore, the neural network output spectrum was divided into four regions, to cater for only four protein superfamilies. This output spectrum division approach may not be feasible, as more and

more protein superfamilies come into the picture. Our proposed hybrid system attunes one neural network to one protein superfamily. As more and more protein superfamilies come into the picture, the system is easily scalable – requiring only the further addition of neural network components.

5 Contributions to the Field of Bioinformatics

This system has four main contributions to the field of bioinformatics: (a) it provides a framework for the integration of various sequence encoding schemes with one or more Hidden Markov Models; (b) the framework supports variations additions and modifications to the manner it processes protein similarities by remaining essentially unchanged; (c) applies fuzzy logic in a novel manner as a data post-processor; and (d) was designed to be able to incorporate parallelization, in order to for the system to be scalable, and hence, practical.

This system allows values generated from different sequence encoding schemes and probability values generated from Hidden Markov Models to be analyzed in an integrated manner. Values from Hidden Markov Models have been used as inputs to neural networks before (e.g. in speech recognition applications [7]). However, our system extends this concept by including values from sequence encoding schemes, along with a normalizer to prevent over-bias. Our system supports additions and modifications to the encoding schemes used, as well as additions of Hidden Markov Models, in order to facilitate the incorporation of new discoveries into our system. The framework of the system is robust enough to remain essentially unchanged as new discoveries are seamlessly incorporated into it. Other systems which are hard-coded on a specific algorithm may become redundant as new discoveries on protein similarities are made. Fuzzy logic has been used in bioinformatics before, to study gene expression [6]. This system applies fuzzy logic in a novel manner, in order to post-process the outputs from the array of neural networks that makes up the system, in order to help yield new information on potential drug side effects. We do not know of any other application of fuzzy logic in this manner.

We also developed an approach on how to parallelize this system, in order to make it scalable. This will help make the system a practical protein data mining system, able to cope with very large data sizes, as well as the incorporation of more complex encoding schemes. Parallelization occurs on two levels. On the first level, the training of a particular neural network alternates between a sequential state and a parallel state. On the second level, the system has one neural network for one protein family, and the training of one network can be conducted independently (i.e. in parallel) of another.

6 Future Work

For the system described above, instead of using conventional neural networks, we could explore the use of weightless neural networks. Weightless neural networks have been typically used for image recognition problems [10]. We could

adapt the weightless neural network concept to cater for the problem of protein classification [11], and compare its results against the results obtained from conventional neural networks based on the delta learning rule or backpropagation. Weightless neural networks only require one pass of the training data, and offer an attractive alternative in terms of improved training performance.

References

1. Altshul, S., Madden, T., Schaffer, A.: Gapped BLAST and PSI-BLAST: A New Generation of Protein Search Programs. *Nucleic Acids Research* (1997).
2. Notredame, C.: SAGA: Sequence Alignment by Genetic Algorithm. *Nucleic Acids Research* (1996).
3. Krogh, A.: An Introduction to Hidden Markov Models for Biological Sequence. Technical University of Denmark (1998).
4. Wang, J., Ma, Q., Shasha, D., Wu, C.: New Techniques for Extracting Features from Protein Sequences. IBM Corporation (2001).
5. Ohler, Uwe., et. al.: A Hybrid Markov Chain – Neural Network System for the Exact Prediction of Eukaryotic Transcription Start Sites, University of Erlangen, Nuremberg (2000).
6. Woolf, P., Wang, Y.: A Fuzzy Logic Approach to Analyzing Gene Expression Data. *Physiol Genomics* (2000).
7. Cohen, M., Rumelhart, D., Morgan, N.: Combining Neural Networks and Hidden Markov Models for Continuous Speech Recognition. Stanford University (1992).
8. Vuurpijl, L., Schouten, T., Vytöpil, J.: PREENS : Parallel Research Execution Environment for Neural Systems. University of Nijmegen (1992).
9. Hopp, H., Prechelt, L.: CuPit-2: A Portable Parallel Programming Language for Artificial Neural Networks. Karlsruhe University (1997).
10. Burattini, E., DeGregorio, M., Tamburrini, G.: Generating and Classifying Recall Images by Neurosymbolic Computation. Cybernetics Institute, Italy (1998).
11. Chew, Martin Wooi Keat., Rosni Abdullah., Rosalina Abdul Salam.: Weightless Neural Network Array for Protein Classification (unpublished), Universiti Sains Malaysia, Malaysia (2004).

Parallelization of Phylogenetic Tree Inference Using Grid Technologies

Yo Yamamoto¹, Hidemoto Nakada^{1,2}, Hidetoshi Shimodaira¹,
and Satoshi Matsuoka^{1,3}

¹ Department of Mathematical and Computing Sciences,
Tokyo Institute of Technology 2-12-1 Oo-okayama,
Meguro-ku, Tokyo, Japan 152-8550
yamamoto@matsulab.is.titech.ac.jp,
hide-nakada@aist.go.jp,
{shimo, matsu}@is.titech.ac.jp
<http://www.is.titech.ac.jp>

² National Institute of Advanced Industrial Science and Technology,
Grid Technology Research Center, Central2,
1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568 Japan

³ National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan 101-8430

Abstract. The maximum likelihood method is considered as one of the most reliable methods for phylogenetic tree inference. However, as the number of species increases, the approach quickly loses its applicability due to explosive exponential number of trees that need to be considered. An earlier work by one of the authors [3] demonstrated that, by decomposing the trees into fragments called splits, and calculating the individual likelihood of each (small) split and combining them would result in a very close approximation of the true maximum likelihood value, as well as achieving significant reduction in computational cost. However, the cost was still significant for a practical number of species that need to be considered. To solve this problem, we further extend the algorithm so that it could be effectively parallelized in a Grid environment using Grid middleware such as Ninf and Jojo, and also applied combinatorial optimization techniques. Combined, we achieved over 64 times speedup over our previous results in a testbed of 16 nodes, with favorable speedup characteristics.

1 Introduction

All form of life today on earth originated from a common biological ancestor; so, any species may be placed as some leaf node of some gigantic phylogenetic tree. One valuable endeavor is to infer a phylogenetic tree given a set of a variety of various species to determine how the individual species have exactly evolved and relate to each other during the course of evolution, in particular when a particular split branching has occurred given a pair of different species. Such

research is quite important to reveal the mechanism of how evolutions have and will occur for various life forms.

Traditional biology mostly inferred the phylogenetic relationships amongst the species by their external features. However, such comparisons often tend to lack precision and objectiveness, and in fact sometimes lead to inconsistent results. With the discovery of DNA, it is now becoming possible to infer phylogenetic trees using mathematical models of evolution constructed on genetic DNA sequences. However, in practice straightforward inference algorithms built on such models have substantial computational complexity, and have remained applicable only to very small problems.

Based on our past work that aimed to reduce the complexity in tree inference without losing precision [3], we further improve the algorithm by applying both numerical optimization and parallelization techniques on a cluster/Grid environment., using task parallel Grid middleware Ninf[2] and Jojo[1]. We obtained nearly 64-fold speedup over our earlier results as a combined effect of both on a small cluster test environment of 16 nodes, allowing us to scale the problem significantly.

2 Inferring Phylogenetic Trees — The Complexity Problem

A sample phylogenetic Tree tree is illustrated in Fig. 1. The maximum likelihood method that will compute such a tree will compute the likelihood value of x_k at a locus k , and consider the product of all such likelihood $\prod_k L(x_k)$ as the likelihood value induced from the particular genomic DNA sequences. $L(x_k)$ will be obtained typically via a non-linear optimization process involving considerable iterations, and as will be computationally non-trivial, as shown in Fig. 2. After obtaining all the likelihood values of candidate phylogenetic trees, we consider the one with the largest one likelihood value to be more most trustworthy. However, the number of phylogenetic trees is quite large in itself, or more precisely, for n specifies the number of tress is $((2n - 5)!)/(2^{n-3}(n - 3)!) = O(2^n n!)$. As a result, computing the likelihood values for all possible candidate trees becomes quickly impractical, even for a relatively small n .

To cope with such massive computational complexity, one of the authors proposed an approximate method for computing the likelihood value of a given tree. We call a branch of a phylogenetic tree a *split*, and given n species we can divide a given phylogenetic tree into $(n - 3)$ set of splits. We then compute the likelihood values of a given splits using the maximum likelihood method, and derive an approximate value of the likelihood value using matrix manipulations. This method has considerable computational complexity advantage without losing much precision. Since the total number of possible splits is $2^{n-1} - (n + 1) = O(2^n)$, we can significantly reduce the overall computational cost. However, the number of phylogenetic trees is still significant, and this method, although a definite improvement, still was too expensive of realistic values of n .

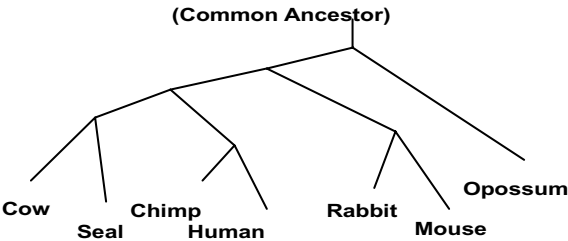


Fig. 1. An Example Phylogenetic Tree

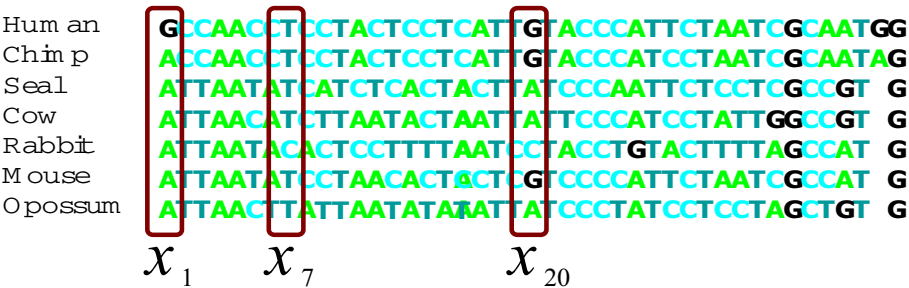


Fig. 2. DNA Sequence, and computing of the likelihood values

3 Overview of Our Proposed Improvements to the Split Method

Our improvements over the previous proposal are twofold. One is to apply combinatorial optimization techniques to reduce the search space for the trees. Another is to parallelize the search effectively over the Grid using appropriate task-parallel Grid middleware, Ninf[2] and Jojo[1]. The resulting program was shown to execute efficiently and with significant speedup, even for a relatively small number of nodes on a small-scale cluster. Figure 3 shows the overall workflow; here, we see that the program largely consists of two phases, the first phase being “computing the likelihood value of each split using the maximum likelihood method”, and the second phase being the “combining the splits and searching the optimal results using combinatorial optimization techniques and their parallelization”. The former will perform parameter-sweep parallelization of likelihood values of each possible split, either sequentially or in parallel on the Grid, and output the results in files for the second phase. The second phase in turn will either directly obtain the likelihood values of all combinations of splits, or use combinatorial optimization techniques such as branch-and bound or simulated annealing, and obtain the optimal likelihood value from the “more likely” candidates, again possibly in parallel on the Grid.

For both phases, we parallelize the computation using master-worker scheme, and implement the former using the Ninf GridRPC system, whereas for the latter

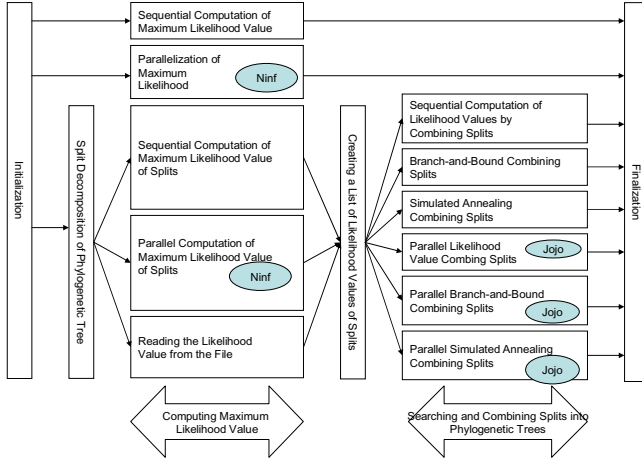


Fig. 3. The Overall Workflow of Deriving the Phylogenetic Tree with Maximum Likelihood Value

we perform further hierarchical master-worker parallelization using a Java Grid parallelization system Jojo. There are various reasons we employ two different Grid middleware systems; the primary reason for using Ninf GridRPC is that, it is easy to integrate existing maximum likelihood numerical packages, while the reason we employ Jojo for the second phase is that, the latter involves hierarchical parallelization, in particular branch-and-bound computation. From a pure Grid middleware research perspective it is also interesting to investigate how the two different middleware will interoperate smoothly on the Grid.

4 Optimizing Phylogenetic Tree Inference

In order to reduce the number of candidate phylogenetic trees., we employ combinatorial optimization techniques.

4.1 Using Branch-and-Bound

Since each phylogenetic tree corresponds to $(n - 3)$ sets of splits, we obtain a new tree by combining a split onto a star-shaped phylogenetic tree one by one, as shown in Figure 4. Since there are multiple ways how split can be combined at each stage, the search space branches out into a search tree as in Figure 4, with the leaves being the candidate phylogenetic tree. We then apply branch-and-bound technique onto this search tree, allowing us to prune the search space significantly with an appropriate bounds function.

In order to prune the branches, we compute the upper bound of the likelihood value for each node in the tree, and compare the value against the current solution; if the upper bound is greater, we continue the search by expanding the node; otherwise, we prune the branch of the search tree under that node.

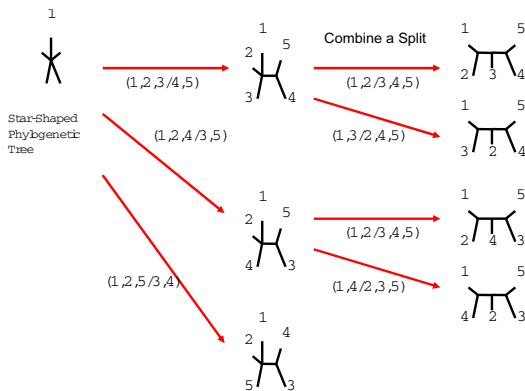


Fig. 4. Formulating a Search Tree of Combining Splits

As an upper bound, we employ the combined likelihood value where we combine all possible splits onto that (pair of splits) node.

4.2 Using Simulated Annealing

For simulated annealing, we obtain the neighboring solutions based on splits, as we see in Figure 5. Firstly, from the set of splits that signifies the current solution, we arbitrarily remove one of the splits. Then, of the three possible splits that could be combined with the current set, we remove the split that

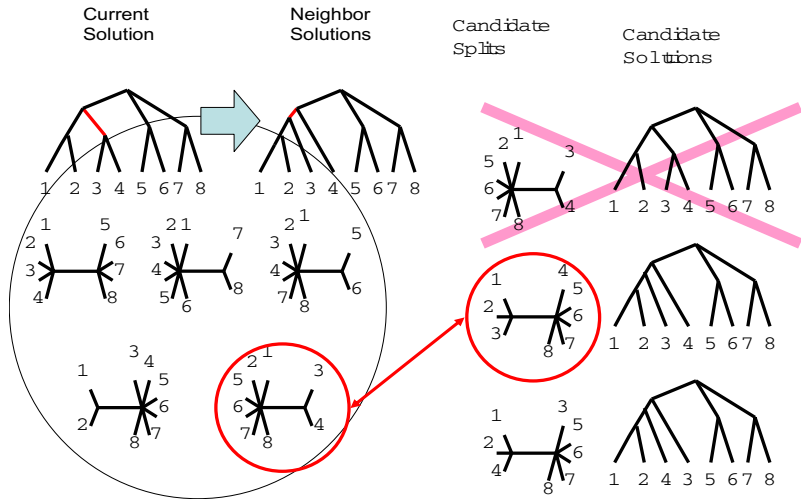


Fig. 5. Creating Neighboring Solutions and Candidates in the Simulated Annealing Scheme

would result in idempotent return to the original before split removal. Then, from the remaining two we pick one at random, and combine with the set of splits, deriving the neighboring solutions. The “cooling” function we employed was, given the cooling parameter α we simply perform exponential degradation $T_{next} = \alpha T_{current}$ ($0 < \alpha < 1$).

5 Parallelizing Phylogenetic Tree Inference on the Grid

If you wish to include color illustrations in the electronic version in place of or in addition to any black and white illustrations in the printed version, please provide the volume editors with the appropriate files.

If you have supplementary material, e.g., executable files, video clips, or audio recordings, on your server, simply send the volume editors a short description of the supplementary material and inform them of the URL at which it can be found. We will add the description of the supplementary material to the online version of LNCS and create a link to your server. Alternatively, if this supplementary material is not to be updated at any stage, then it can be sent directly to the volume editors, together with all the other files.

5.1 Using Simulated Annealing

The computation in the first phase continues in a simple master worker style. The master first creates a pool of phylogenetic trees or splits subject to further computation, and sends them as jobs off to the worker nodes on the Grid one by one. The worker computes the maximum likelihood and returns the value; the worker aggregates the returned result and resends another job to an idle worker, until the pool of trees and/or splits is exhausted.

5.2 Parallelizing the Split Method Directly

The second phase can also be parallelized in a straightforward fashion in a similar manner using master-worker style computation. This time the master sends the phylogenetic tree and the corresponding split pair to the workers. The workers in turn compute the approximate likelihood values of the combined phylogenetic tree based on the likelihood values obtained in phase one, and returns the result to the master. The master keeps track of the process, continuing until all the phylogenetic trees that can be generated are covered, and picks the tree with the largest likelihood value as the result.

5.3 Efficient Parallelization of the Split Method Using Branch-and-Bound

With branch-and-bound, parallelization is performed somewhat differently for phase two. However, a common problem with parallelizing branch-and-bound is that, there could be considerable load imbalance depending on how the search tree is divided, and moreover, some computation may go to waste if the bound

value of some branch turns out to prune a computation ongoing on some other processor. The shape and the depth of subtrees of a search tree may greatly differ amongst one another, and whether or not a search should be conducted on a subtree is runtime dependent. As such, naive master-worker subdivision of (sub) search tree may turn out to be quite inefficient.

In order to avoid this problem, we set a limit on the number of “problems” (i.e., computing the likelihood value of each split) individual workers will compute for each subtree. The master maintains a pool of problems, and distributes each one by one to the workers. The worker then proceeds to solve the problems in the manner similar to Section 5.2. If the number of problems that the worker has solved exceeds some threshold value, the problem is returned to the master. The master in turn re-adds the problem to the pool for subsequent re-allocation to some worker.

5.4 Parallelizing Simulated Annealing

For parallelizing simulated annealing, we employed the replica exchange method as outlined below. Each worker maintains an independent simulated annealing process, each with a different temperature parameter. At some periodic *exchange interval* the worker sends the current likelihood value to the master, and requests for exchanging the temperature value. Let us label this worker i . When the master receives this request, the master notifies worker j , where $j < i$ and holds the maximum temperature value, that an exchange request has been made. The worker j in turn judges whether the exchange of the temperature should take place according to the temperature and the likelihood value it has received. If it decides to accept the exchange it performs the temperature exchange and notifies the master its previous temperature and the current likelihood value; the master then notifies worker i that the exchange was successful, the worker i performs its own exchange internally, and the entire exchange process completes. Otherwise, if the worker j decides to deny the request, it notifies the master who in turn notifies worker i that the exchange was unsuccessful. Irrespective of whether the exchange was successful or not, the workers continue with their search until the next exchange period is encountered.

6 Evaluation of Our Proposed Scheme

6.1 Evaluation Criteria and the Target Problem

As the evaluation criteria, we employed and measured the followings:

- **Evaluating the original approximation algorithm using splits:** We initially investigate the effect of our original scheme of deriving the approximate likelihood value by combining splits to form the phylogenetic tree, by comparing its precision and compute cost
- **Evaluating the reduction of the search space using combinatorial optimization techniques:** We next evaluate how much the search space

has been reduced by employing branch-and-bound and simulated annealing techniques

- **Evaluating the parallelization efficiency and scalability:** Although in theory master-worker computation could approach near-perfect speedup, in reality we may observe loss in efficiency as we scale the problem larger due to various factors including communication overhead between the master and the worker, as well as the load of the master increasing and becoming the sequential bottleneck. The metrics we employ are firstly the speedup value due to parallelization, i.e.,

$$Speedup = \frac{T_{serial}}{T_{parallel}}$$

and in particular we measure the parallelization overhead as when we have just one master and one worker. We also compute the scalability as:

$$Scalability = \frac{Speedup}{N_{workers}}$$

where $N_{workers}$ is the total number of workers.

We employed paml[4] as the program that computes the likelihood value given a particular base sequence. For our sample problem we used the following 9 species: seal, cow, rabbit, opossum, mouse homo-sapience, dugong, armadillo, and rat. The sequence data for each species are those for mitochondria downloaded from NCHI, and the sequence length is 3392.

6.2 Evaluation Environment

As a preliminary evaluation environment, we employed a cluster as a controlled platform rather than to employ a full-fledged distributed Grid. The employed Abacus cluster consists of dual Athlon MP 2000+ (1.67 Ghz) nodes with 1Giga-Byte of memory, and interconnected via a 100Base-T network. Both the master and the worker have been allocated on the nodes, and we measured using 2, 4, 8, and 16 nodes.

6.3 Evaluating the Original Approximation Algorithm Using Splits

Figure 6 shows the graph of the results of evaluating our original approximation algorithm using splits. We employed 6 species from our 9, and from the total of possible 105 phylogenetic trees, on the horizontal axis we plot the exact maximum likelihood value, versus the approximated value computed by combining splits. As we observe from the graph, most of the values lie on the line $y = x$: as such, we observe that our original approximation method is quite good.

Table 1 shows the effect on the computational costs using our approximation method. For all possible phylogenetic trees for n species, we compare the actual computation time of the approximation method, denoted as T_{comp} , versus the projected computation time using the maximum likelihood method, denoted

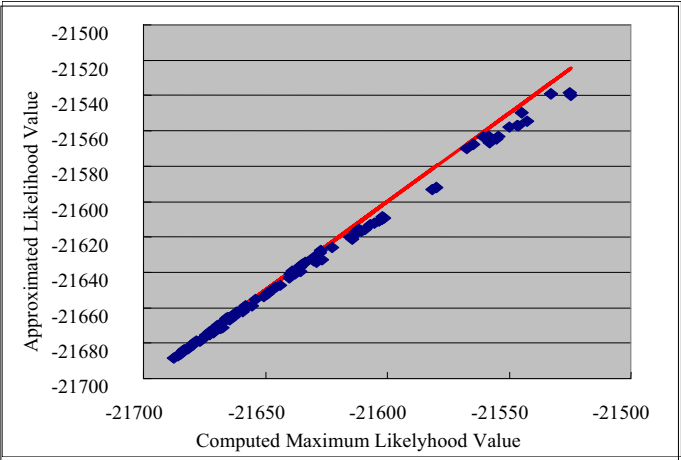


Fig. 6. Comparing Approximated Likelihood Value with the Real Maximum Likelihood Value

Table 1. Effect of Computational Cost Reduction Using the Approximation Method

n	Ntree	Tave (sec)	Tpaml	Tcomp
5	15	37	9 min 30 sec	3 min 7 sec
6	105	85	2 hours 30 min	6 min 55 sec
7	945	149	1 day 15 hours	35 min 22 sec
8	10,395	241	29 days	2 hours 44 min
9	135,135	330	1 year 5 months	18 hours 41 min

as T_{paml} (computed as $T_{ave} \times T_{ave}$, where T_{ave} is the sampled average value of computing the maximum likelihood value). As we observe in the table, we drastically reduced the overall compute time.

The approximation method internally consists of two phases, where we compute the maximum likelihood value of each split, and the second phase where we combine the splits to form each phylogenetic tree. Figure 7 shows the breakdown of the compute time for these two phases. As we can see, the time to combine the split becomes more dominant as n grows larger. This is because the number of splits is $O(2^n)$ whereas the number of possible phylogenetic trees is $O(2^{2^n}n!)$. Based on this observation we simply parallelized the first phase whereas we performed parallelization as well as combinatorial optimization for the second phase.

6.4 Evaluating the Reduction of the Search Space Using Combinatorial Optimization Techniques

Branch-and-Bound. For branch-and-bound, we measured how much pruning of the search tree is achieved. Since on every interior search nodes we need to

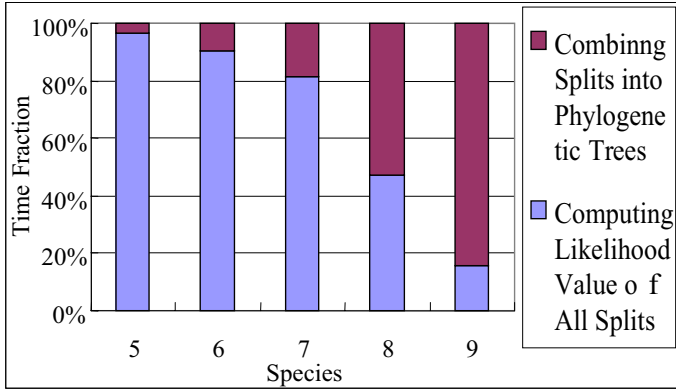


Fig. 7. Time Fraction of Two Phases in the Approximation Algorithm

combine all possible splits combinable with the current split that represent the search status, the interior nodes take just as much time to compute the likelihood value as the leaves; because of this we need to account for this cost and not just account for the computational cost at the leaves. In total, we were able to prune the search space by 83.4%, 93.5%, and 95.3% for 7, 8, and 9 species, respectively.

Simulated Annealing. For simulated annealing, the number of times the search is repeated depends on the termination condition. Here, for benchmarking purposes, we precompute the likelihood values of all the phylogenetic trees beforehand, and have the termination condition be such that the result falls within 1% of the precomputed results for 10 consecutive times. Also, since there will be effects of randomness in the results, we perform the experiment 3 times and take the average of the results (Note that these are purely for evaluation purposes of the obtained results, and not something to be used in practice.). We alter the initial temperature and the cooling parameters in various experiments.

For 7 species, we found that, when we set the initial temperatures to be low, the number of search repetitions will be smaller, but we also have had to enlarge the cooling parameters, or otherwise we may not achieve convergence in the results. Experimenting on various parameters, we found that we could converge and reduce the search space by up to 95.2% similar to the results obtained with branch-and-bound.

6.5 Evaluation of Scalability with Parallelization

On evaluating simple parallelization of maximum likelihood value computation for 5–6 species, we found the overhead to be about 12% for 6 species. This overhead increases with larger n instead of decreasing, being counterintuitive since the granularity does not decrease with a larger n . We currently consider that the overhead is largely due to I/O of the phylogenetic tree itself, and working to resolve the issue. The results that follow must be regarded with this issue in mind.

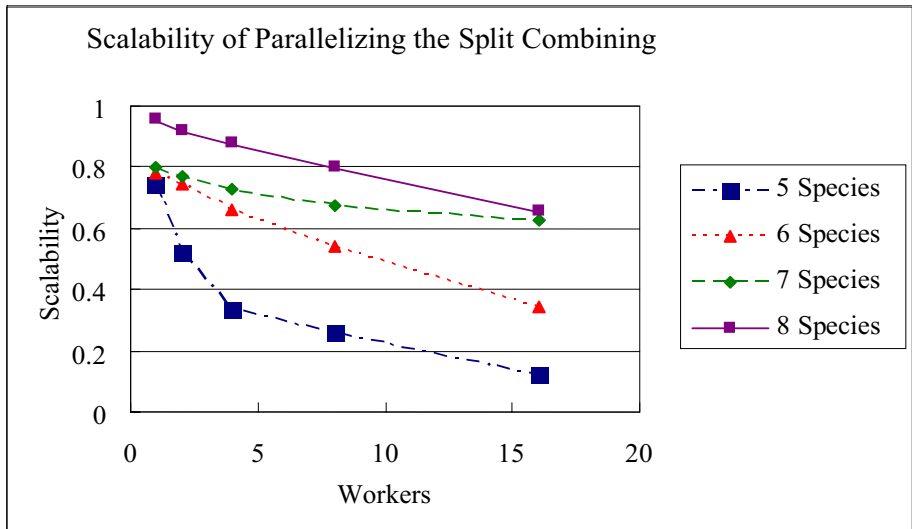


Fig. 8. Scalability of Parallelizing the Combining of the Splits in our Original Approximation Algorithm

For scalability, we obtained approximately 6.5 to 7.5 times speedup with 8 workers, and 12.7 times speedup with 16 workers. Given that the overhead is 12%, these are nearly ideal results.

For parallelizing our approximation scheme without combinatorial optimizations, the overhead ranged from 30% for 5 species, to about 5% for 8 species. Figure 8 shows the scalability graph, where we obtained 10.5 times speedup for 8 species on 16 nodes.

Evaluating Parallelization under Branch-and-Bound. We varied the number of species by 5–9 for evaluating parallelized branch-and-bound. The overhead due to parallelization is less than 3–4% for over 5 species, indicating effective parallelization. Scalability is quite excellent as seen in Figure 9. We need to conduct experiments with a significantly larger n to observe scalability of our parallelization on a larger number of nodes on the Grid, however.

Evaluating Parallelization of Simulated Annealing (Replica Exchange Method). Finally, for simulated annealing parallelized with replica exchange method, we set the maximum temperature values to 5, 10, 20, 50, 100, 200, and 500, and the minimum temperature to be 1. Each worker was assigned an initial temperature value so that the intervals of the temperature values will have constant ratio with its neighbors, and cover the range from the minimum to the maximum. For example, if the maximum temperature is 8, and the number of workers is 4, the ratio will be 2, and the assigned values will be 1, 2, 4, and 8. We experimented with 4 to 16 workers and 7 species, with the termination

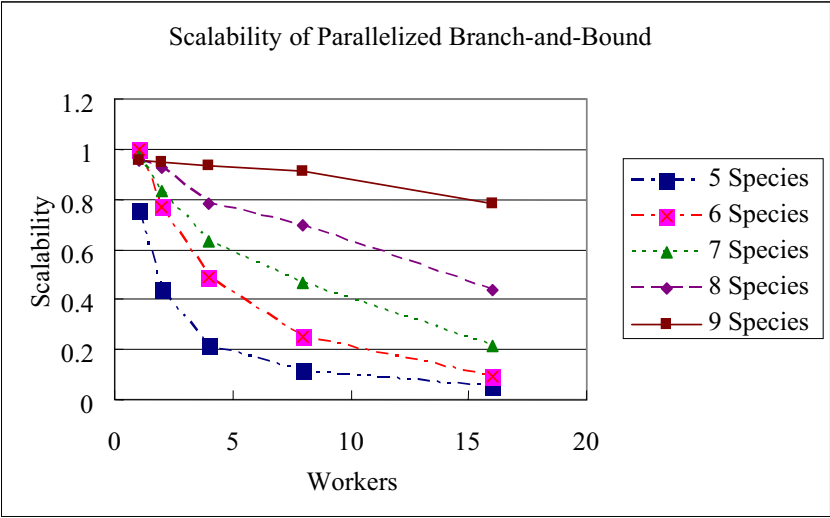


Fig. 9. Scalability of Parallelizing the Branch-and-Bound Optimization

condition be such that if one of the workers satisfies the same condition as the sequential case, the entire system is terminated.

Figure 10 shows the results: here, we observe that (as expected) the replica exchange method derives no speedup, but rather contributes to stability of the results convergence. Irrespective of the initial temperature value, the results converge after about 60 iterations, achieving 94% pruning of the search space.

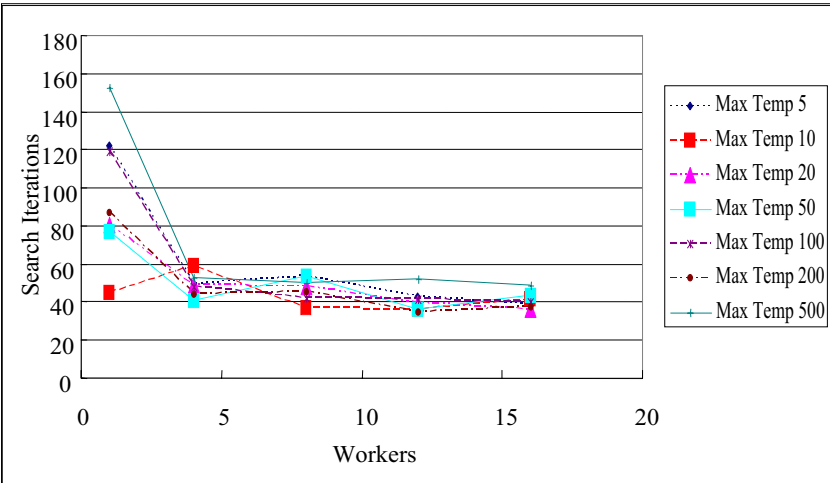


Fig. 10. Convergence Characteristics of Simulated Annealing / Replica Exchange Method

Although subject to benchmarking in a larger system, the initial results are favorable in that the user of our scheme may be relieved of truing the parameters appropriately to achieve fast convergence, when simulated annealing is advantageous over branch-and-bound.

7 Conclusion and Future Work

We demonstrated that, by combining our approximation scheme of the likelihood value of phylogenetic trees with appropriate combinatorial optimization techniques and parallelization techniques on the Grid, we obtain substantial speedups with good efficiency, pruning the search space as much as 95% for both branch-and-bound as well as simulated annealing techniques. The net effects of all the approximations and optimizations was 64 times over our original approximation method, or over 50,000 fold speedup over the straightforward sequential algorithm for obtaining maximum likelihood value for each candidate tree, with 16 processors. This is a promising result that allows comparison of fairly large number of n in a realistic timeframe.

As a future work, we need to further investigate methods for coping with scaling up the computation for a larger n . In particular, we need to experiment with other combinatorial optimization techniques, such as Genetic Algorithm, and/or other algorithms for computing the likelihood values more efficiently. We also need to reduce the parallelization overhead as well as matching the more hierarchical nature of the resources on the Grid. Here, the hierarchical organization features of Jojo could be of good help, but we need to validate the scalability in a real Grid. Another issue under a practical setting is fault tolerancy, which has not been built into our system. Since master-worker style computation is somewhat amenable to easier checkpointing, we are planning to use some of the checkpoint as well as recovery features in the new versions of Ninf and Jojo. Finally, we need to make both our code as well as our environment available, possibly in the form of portals so that computed results could be systematically stored for access by phylogenetic researchers.

Acknowledgements

This research was partially funded by the Japanese ACT-JST “Terascale large-scale optimization using commodity Grid technologies” project, as well as the Japanese Scientific Grant-in-Aid (A) 14702061.

References

1. Hidemoto Nakada, Satoshi Matsuoka, and Satoshi Sekiguchi. “Java-Based Programming Environment for Hierarchical Grid: Jojo”, in *Proceedings of IEEE Computing in Clusters and the Grid (CCGrid) 2004*, the IEEE Press, April, 2004.

2. Hidemoto Nakada, Mitsuhsa Sato, and Satoshi Sekiguchi. “Design and Implementation of Ninf: Towards a Global Computing Infrastructure”. In *Future Generation Computing Systems, Metacomputing Issue*, Vol. 15, pp.649–658, 1999.
3. Hidetoshi Shimodaira. “Multiple Comparisons of Log-Likelihoods and Combining Nonnested Models with Applications to Phylogenetic Tree Selection”, in *Comm. In Statistics, Part A-Theory and Meth.*, Vol. 30, pp. 1751–1772, 2001.
4. Z. Yang. “Paml: A Program Package for Phylogenetic Analysis by Maximum Likelihood”, in *CABIOS*, Vo. 13, pp. 555–556, 1997.

EMASGRID: An NBBnet Grid Initiative for a Bioinformatics and Computational Biology Services Infrastructure in Malaysia

Mohd Firdaus Raih^{1,2}, Mohd Yunus Sharum¹,
Raja Murzaferi Raja Moktar¹, Mohd Noor Mat Isa¹,
Ng Lip Kian³, Nor Muhammad Mahadi^{1,2}, and Rahmah Mohamed^{1,2}

¹ Interim Laboratory, National Institute for Genomics and Molecular Biology,
Heliks Emas Block, UKM-MTDC Smart Technology Centre,
43600 UKM Bangi, Malaysia

<http://genome.ukm.my/>

² School of Biosciences and Biotechnology, Faculty of Science and Technology,
Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Malaysia

<http://www.ukm.my/biosains/>

³ Asia Pacific Science and Technology Centre, Nanyang Technological University,
School of Mechanical and Production Engineering,

50 Nanyang Avenue, Singapore 639798

<http://apstc.sun.com.sg/>

Abstract. The plethora of bioinformatics tools currently available to the biologist, and the diversity of problems that these applications were designed to solve, has necessitated a look at providing a single environment which can serve as an interface to these many applications. At the same time, this environment should also be able to function as an infrastructure resource with adequate computational capacity to solve the data volume which is currently available from numerous genomics projects. We discuss the setting up of a national level infrastructure initiative which utilizes grid computing technology to serve geographically distributed users in Malaysia. This infrastructure was designed to provide access to high performance computational resources made available by Sun Microsystem's Sun Grid Engine (SGE) using different interfaces which access a pipeline of bioinformatics software. The underlying computer system, such as operating systems and high performance computing applications, were bypassed with the creation of an application layer of bioinformatics tools (BioGrappler) or by accessing the compute resources by the Grid Engine Portal (BioBox).

1 Introduction

The concept of a wrapper program or a web accessible interface of computational biology software to interact with a compute grid is not new. Applications which are sequence alignment based, such as BLAST [1, 2] and CLUSTAL W [3, 4] have been deployed using high performance computing technologies numerous times.

A web accessible interface to bioinformatics applications which run on a compute grid is also not a new development. Many developments were however tailored for computational biologists or bioinformaticists as the technologies used relied heavily on the user's ability to work in a Unix operating system environment as well as writing scripts to assist in job submission and control. For large research groups with adequate human resources and support in bioinformatics, this scenario does not present a major problem. In developing countries and smaller research groups, where the researcher population in a particular group may consist of mainly molecular biologists or biochemists, with almost no understanding of Unix, scripting or the concept of high performance computing (HPC) and otherwise trivial technicalities such as operating systems can be considered an important limiting factor.

Another important factor to consider is the serial nature of analysis for biological data. DNA sequencing results are usually filtered, screened for vector sequences, edited, assembled and submitted to similarity searching. Similarity searching results are then used for multiple sequence alignments, protein structure prediction and phylogenetics. Currently, there is no one software that will fill all the analysis requirements available. There are however many well proven software which will fill a particular niche requirement such as BLAST for similarity searching, Clustal W for multiple sequence alignment and Modeller [5] for protein structure prediction. Arising from this serial analysis is the problem of format multiplicity. Each application may have input and output formats which are different resulting in the result of one analysis not being immediately readable as the input for a subsequent analysis. As such there is a requirement that for high throughput data cases such as genome projects, the software functions mentioned will have to work in a seamless pipeline.

1.1 NBBnet and EMASGRID

The National Biotechnology and Bioinformatics Network, Malaysia (NBBnet) is a virtual resource and capacity building network tasked at providing and managing administrative services as well as scientific computational biology services for the Malaysian biotechnology community through the use of informatics [6]. Via this virtual network, physical sharing of resources are conducted. Amongst the resources shared are geographically distributed clusters of computational resources. By utilizing available grid technology, heterogenous hardware in a single physical location, are clustered into a single cluster grid by SGE. Job submissions between cluster grids are executed by queue transfers using the Globus toolkit [7]. EMASGRID, an acronym for Extensible Malaysia (NBBnet)-SGE Grid is an infrastructure concept developed to serve geographically distributed users in harnessing a centralized compute grid operated by SGE. This centralized infrastructure model is however quite suitable to be operated at a much smaller departmental or campus level as well.

1.2 Objectives for Biological Compute Grid Infrastructure

In Malaysia, the population of scientists involved in biotechnology research consists of mainly molecular biologists or basically biochemistry trained personnel. These researchers churn out diverse data sets from small volume data on specific proteins of interests to high volume genome sequencing data. The unavoidable fact under present circumstances is that probably all of them will need to utilize some form of informatics assistance in managing and analyzing data. Another unavoidable factor when taking into context the current scenario in high performance computing is the usage of Unix based operating systems and highly technical interfaces to HPC resources. The infrastructure initiative discussed here addresses this problem by providing easy to use interface presented in the 'language' of the biologist as an application layer to a heterogeneous high performance computing infrastructure using Sun Microsystems' Sun Grid Engine technology (SGE). This in effect provides an avenue for non-bioinformatics savvy biologists to operate and run high volume jobs on a compute grid with almost no retooling or retraining exercises for understanding the infrastructure involved being carried out.

2 Organization and Deployment

EMASGRID resources are accessible via multiple interfaces. Currently users are able to access the computational infrastructure via web (two access interfaces) or by remote Unix login via secure shells (ssh) (Fig. 1). Web browser access is achieved by the Grid Engine Portal (GEP) BioBox initiative or via submissions through the Bioinformatics Tools @ NBBnet (BT@N) interface. The BioBox interface is essentially targeted for user balancing and ease of access to a wider range of tools (17 applications are integrated - <http://apstc.sun.com.sg/biobox.php>) as opposed to the BT@N interface with only BLAST integrated. Operationally, this means that user volume will be distributed to the grid as opposed to input volume. Here, we define user volume as the number of users simultaneously logged in. Input volume is defined as the number of jobs required to be distributed by SGE irregardless of the number of users. This access is suitable for geographically distant users who need access to the diverse set of tools available via BioBox and may need to submit many jobs to different applications. Processing time depends on the volume of the data and complexity of the input as only one node is utilized for one job.

2.1 Bioinformatics Tools @ NBBnet

The BT@N interface differs slightly in function when compared to the GEP interface. Via this interface data in the form of FASTA formatted sequences are contained in a MySQL database (Fig. 1). The BT@N interface was originally developed as a personal database and meta-server system for limited volume data. In this case, the limit defined as low volume was set at 100 entries with each entry having a possibility of 2 sequences for nucleotide and its protein

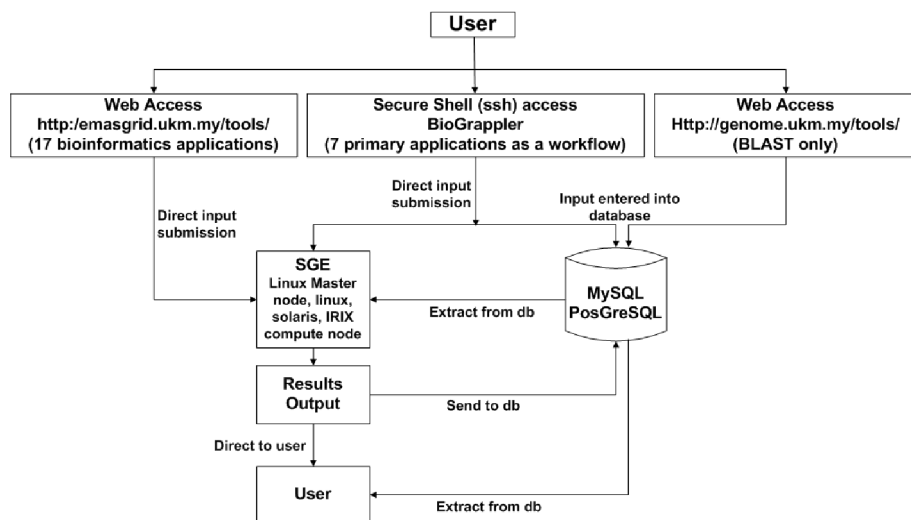


Fig. 1. Overview of the organization for the EMASGRID deployment

translation. Via this interface users can select sequences of interest and submit them to SGE for BLAST. The system creates a file of the selected sequence and submits the sequences for BLAST via SGE. Volume balancing of the input is enabled via this interface. In the scenario above, if 50 sequences were selected from the user database, the files would be split into user defined parameters of 10 to 20 sequences for each available CPU to process. Results are collected and can be either reintegrated as either a single output file or as a corresponding BLAST result for each sequence submitted (in this case 50 output files). Submission via BT@N can also accept FASTA formatted sequence files via http uploads. At this point in time, this system is only able to submit BLAST searches.

2.2 BioGrappler

The Bioinformatics Grid Applications Pipeline Environment, acronymed as BioGrappler, and developed as an NBBnet initiative to provide its members with access to HPS resources, was designed as an application layer or wrapper program to compute resources available to SGE (Fig. 1). As the name implies, its primary objective was to provide a continuous flow of biological sequence analysis resources, which are distributed over a heterogenous compute grid. Users are able to access the interface via ssh logins. BioGrappler provides a utility where a user can achieve the following:

- edit and ascertain sequencing chromatogram quality using phred [8, 9]
- screen for vector sequences (Cross_match – <http://www.phrap.org>)
- database similarity searching (BLAST)
- sequence clustering by BLAST (blastclust [10]) and CD-HIT [11]

- multiple sequence alignment by ClustalW
- protein structure prediction by Modeller.

BioGrappler which in essence is a bioinformatics wrapper program for SGE (ver. 5.3) was written entirely in Perl. This allows flexibility and quick deployment capability to different platforms. In this laboratory, it is currently implemented as an agent to SGE in a master-slave grid model with centralized control and centralized data. It can easily be adapted to a cooperative model with centralized control and distributed data. The cooperative model would be better equipped to enhance the performance of the grid processing for cases where the nodes have ample storage space for the multiple databases required. To date, the developments have been extensively tested on Linux (kernel 2.4.26). However, this does not restrict the compute nodes to just Linux as BioGrappler need only run on the master node. The major advantages which BioGrappler has over the previously discussed web accessible systems is the suite of applications which are arranged as a workflow and the capability to better tap SGE for a more efficient workload distribution. This gives the effect of a very customized job submission such as those which can be achieved by scripts. BioGrappler accepts 2 sets of formats as the primary input namely either ABI files (chromatograms) or FASTA formatted sequence files. The program phred is used for quality editing of the ABI files. The output of phred is prepared to be submitted as the input for either a vector screening application (Cross_match) or a sequence database similarity search (BLAST). The applications can still be operated as standalone job submissions. Sequences which need to be submitted to Modeller for protein structure prediction can be prepared as Modeller input files directly and will be submitted to SGE for balancing of the queue and processing load.

BioGrappler consists of three basic levels of data flow management, namely, (1) the input management system, (2) the submission and SGE management module and finally (3) the output management module. The input management module controls the splitting of queries to available resources. A simple approach of splitting high volume input into smaller chunks of input for submission was done. For example, a BLAST search for 10 000 sequences as input would be organized into input throughput of 20 to 50 sequences per submission. EMASGRID is operated using 10/100 Mbps ethernet. Therefore, we observed that smaller numbers of sequences per submission, which means a higher number of submissions per session, would quickly exhaust the network bandwidth, resulting in loss of communications with networked file systems and causing the problem of runaway nodes. By simply limiting CPU submissions to 20 to 50 sequences, and thereby creating bandwidth ‘breathing space’, there was a decrease in the problem of runaway nodes. The queries were not split for multi processor processing as BLAST could sufficiently process the submission volumes tested. This application was roughly able to speed up an unbalanced job by approximately 98 hours. However, no benchmarks were done due to the heterogenous and operational nature of the grid. Furthermore, CPU resources were not uniformly available for every submission. As a result only an approximate estimate of the increased processing capacity is available. Optimizations and tuning were pri-

marily done on the side of the wrapper program as opposed to optimizing from SGE. Most optimizations to data processing jobs were done for high volume case scenarios using available CPUs as distributed single units. BioGrappler has currently been tested as able to submit and manage an average of 10 000 to 20 000 sequences as multiple submissions to SGE (20 CPUs) in a single session. BioGrappler is currently not optimized to run multiprocessor jobs using MPI.

3 Discussion and Future Directions

We view the collaborative development of wrapper applications to existing and proven bioinformatics software as an important aspect of grid computing in the life sciences. The developments can be approached as collaborative but segmented projects. This approach to life sciences grid proliferation can take place by utilizing any existing grid technology available as in the case of this paper, SGE was used. The main difference is that extensive work is done on the wrapper application to optimize the deployment of the SGE instead of vice versa. We hope that by this approach, a more biologist friendly application could be produced which could appeal to a wider user base.

Development of interfacing applications should be as uniform as possible besides being operable in multiple platforms. As a solution to this, we have approached the developments as web accessible applications. This is true for the BioGrappler program as well. This interface, even though implemented in a Unix environment, can bypasses almost all Unix commands except the most basic such as directory changing, listing and file concatenation. Users will still need to have basic comprehension of the Unix file system. As a result, we hope to be able to deploy a web enabled version of BioGrappler in the near future. It is also important to note that there should be clear identification of redundancies in these developments. In this case, implementations and software which are already available can be integrated into the system without creating new ones. Perl has been an ideal tool for this approach as it enables quick multi platform deployment. The problem of file format incompatibilities should also be taken into account. These new developments should either make use of existing file formats with the aid of a format integration agent. We have implemented this agent by integrating it into a pipelined workflow as discussed. What this translates into are easy data sharing between application which produce or require initially incompatible file formats. The limitations of such an agent lies mainly in the fact that it needs to be constantly updated to recognize corresponding input or output files for software which have been integrated into the grid. We have overcome this by mainly identifying the format and setting a template of the input and output in the integration process. This enables continuous integration of applications into the grid which is accessible to BioGrappler. The deployed web accessible BioGrappler, will adhere to a web services model which will allow the biological data to be fully exploited as proposed by Lincoln Stein [12]. The advantages of enclosing all computational analysis in a single environment are manifold. One of the most important is the ability to reorganize the outputs for

mapping of gene regulatory networks, an important step to approach genomic data as a complete system. A systems approach such as this, or systems biology, would permit the study of all elements in a system in response to genetic (which can be in digital format) or environmental perturbations [13].

4 Conclusion

The EMASGRID initiative stems from a need to address the problem of providing high performance computational infrastructure to a wide user base of geographically dispersed Unix illiterate bench biologists. As such, our approach has chosen to make use of available HPC technology as they are and concentrate development efforts towards providing high performance capacity to the users by limiting the interfaces to biologically comprehensible terminology in what is an essentially ‘fire and forget’ approach to high throughput sequence analysis. We believe that the model we have developed will be a useful infrastructure deployment to other developing countries with the ambition for genomics and high throughput biology but without the high throughput funding. Such a development model could better promote a borderless integrative approach to solving biological question without requiring over crossing of traditional discipline boundaries. This is hoped to result in parallel advancement of HPC technology as the requirements are observed and at the same time create a more efficient way of tapping these HPC advancements.

Acknowledgements

We thank the National Biotechnology Directorate, Ministry of Science, Technology and Innovation, Malaysia for the funding and support to NBBnet and microbial genomics research grant IRPA 09-02-02-002 BTK/TD/003; the NBBnet R&D team at the Interim Lab, National Institute for Genomics and Molecular Biology, Malaysia; Sun Microsystems for a hardware grant in grid computing and Asia Pacific Science and Technology Centre, Singapore for the technical assistance in grid computing.

References

1. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Research* 25 (1997) 3389–3402.
2. Hokamp, K., Shields, D.C., Wolfe, K.H., Caffrey, D.R.: Wrapping up BLAST and other applications for use on Unix clusters, *Bioinformatics* 19 (2003) 441–442.
3. Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research* 22 (1994) 4673–4680.

4. Li, K.B.: ClustalW-MPI:ClustalW analysis using distributed and parallel computing, *Bioinformatics* 19 (2003) 1585–1586.
5. Sali, A., Blundell, T.L.: Comparative protein modelling by satisfaction of spatial restraints, *Journal of Molecular Biology* 234 (1993) 779–815.
6. Firdaus Raih, M., Harmin, S.A., Ahmad, H.A., Isa, M.N.M., Mahadi, N.M., Ibrahim, A.L., Mohamed, R.: NBBnet — The National Biotechnology and Bioinformatics Network: A Malaysian initiative towards a national infrastructure for bioinformatics. *Electronic Journal of Biotechnology*, 6 (2003) Available online: <http://ejbiotechnology.info/content/vol6/issue1/issues/03/>
7. Foster, I., Kesselman, C.: Globus: a metacomputing infrastructure toolkit, *International Journal of Supercomputer Applications* 11 (1997) 115–128.
8. Ewing, B. and Green, P.: Basecalling of automated sequencer traces using phred. II. Error probabilities, *Genome Research* 8 (1998) 186–194.
9. Ewing, B., Hillier, L., Wendl, M. and Green, P.: Basecalling of automated sequencer traces using phred. I. Accuracy assessment, *Genome Research* 8 (1998) 175–185.
10. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignments search tool, *Journal of Molecular Biology* 215 (1990) 403–410.
11. Li W., Jaroszewski, L., Godzik, A.: Clustering of highly homologous sequences to reduce the size of large protein databases, *Bioinformatics* 17 (2001) 282–283.
12. Stein, L.: Creating a bioinformatics nation, *Nature* 417 (2002) 119–120.
13. Hood, L., Galas, D.: The digital code of DNA, *Nature* 421 (2003) 444–448.

Development of a Grid Infrastructure for Functional Genomics

Richard Sinnott¹, Micha Bayer¹, Derek Houghton¹, David Berry²,
and Magnus Ferrier²

¹ National e-Science Centre, University of Glasgow, G12 8QQ
{ros, bayermm, derekh}@dcs.gla.ac.uk

² National e-Science Centre, University of Edinburgh, EH8 9AA
{daveb, magnus}@nesc.ac.uk

Abstract. The BRIDGES project is incrementally developing and exploring database integration over six geographically distributed research sites with the framework of a Wellcome Trust biomedical research project (the Cardiovascular Functional Genomics project) to provide a sophisticated infrastructure for bioinformaticians. Grid technologies are being used to facilitate this integration. Key issues to be investigated in BRIDGES are data integration and data federation, security, user friendliness, access to large scale computational facilities and incorporation of existing bioinformatics software solutions, both for visualisation as well as analysis of genomic data sets. This paper outlines the initial experiences in applying Grid technologies and outlines the on-going designs put forward to address these issues.

1 Introduction

Hypertension affects a quarter of the adult population in western societies and is the major cause of cardiovascular mortalities. It is believed that hypertension is caused by a combination of factors including both genetic and environmental influences. The Wellcome Trust has funded a large (£4.34M) collaborative project (Cardiovascular Functional Genomics - ‘CFG’ [1]) to investigate the causes of hypertension. This five year project involves five UK and one Dutch site (depicted in Fig. 1). It is pursuing a strategy combining studies on rodent models of disease (mouse and rat) contemporaneously with studies of patients and population DNA collections. The project is a prime example of the large-scale computational problems associated with modern biology, with requirements to combine vast arrays of heterogeneous information about three species, human, mouse and rat.

Currently many of the activities that the CFG scientists undertake in performing their research are done in a time consuming and largely non-automated manner often requiring navigation to many different data resources web sites and following multiple links to potentially relevant information. Similarly, in their pursuit of novel genes and understanding their associated function the scientists often require access to large scale compute facilities to analyse their data

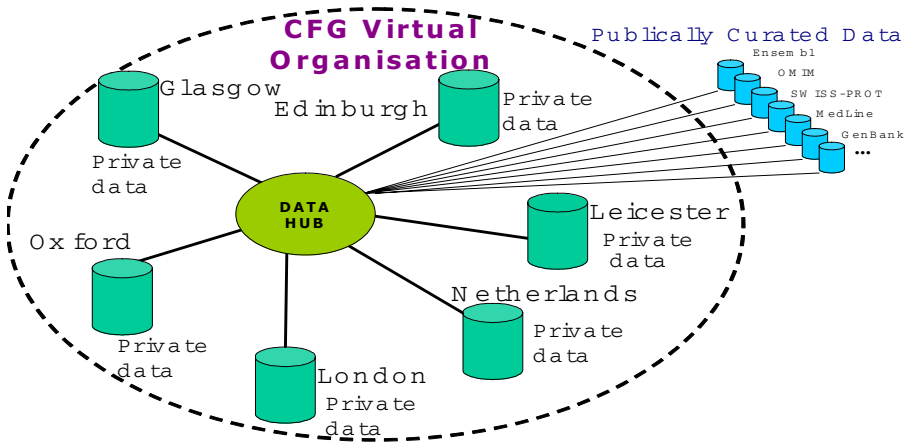


Fig. 1. Data Distribution and Security of CFG Partners

sets, e.g. in performing large scale sequence comparisons or cross-correlations between large biological data sources.

The Biomedical Research Informatics Delivered by Grid Enables Services (BRIDGES) project [2] has recently been funded by the UK Department of Trade and Industry to directly address the needs of the CFG scientists and provide a thorough investigation of relevant technologies for this purpose. Specifically, BRIDGES will investigate the application of Open Grid Services Architecture – Data Access and Integration (OGSA-DAI) [3] and IBM’s Information Integrator product [4] to deal with federation of distributed biomedical data. Evaluation and benchmarking of these technologies is an important component of the BRIDGES work. In addition security is extremely important for the scientists. The scientific data itself may have different characteristics. We consider three primary data kinds based upon their security characteristics:

- Public data: including public genome databases such as Ensembl [29] and GenBank [35], gene function databases such as LocusLink [36] and OMIM [37] and relevant publications databases such as PubMed [39] and MedLine [40];
- CFG specific data: that is to be shared between the CFG consortia only, or subsets of the CFG consortia;
- Private data: including potentially patient records and animal experiment data. This data has strict requirements on its access and usage which the Grid infrastructure must adhere to.

To meet these security requirements, Grid technology is being employed to establish a CFG *virtual organisation*. Virtual organisations provide a framework through which the rules associated with the participants and resources are agreed and enforced – especially those specific to security requirements. The distribution of CFG partners and the data security needs are depicted in Fig. 1. A central

component to this virtual organisation is the notion of a Data Hub which is described in detail in section 2.

The Grid infrastructure to be deployed by BRIDGES should address all of the security concerns and interlinking of the different data sets in as transparent, and user friendly a manner as possible. The overall architecture and Grid related technologies themselves used to ensure this are discussed in section 2 (Overall Bridges Architecture), section 3 (Data Access and Integration), section 4 (Security Considerations) and section 5 (Portal Technologies). Issues and experiences in Grid enabling bioinformatics tools that the scientists use for visualisation of genomic data sets (and between data sets), as well as for analysis of these data sets on high throughput computational farms are discussed in section 6 (Grid Service Development). Finally in section 7, we draw conclusions and outline plans for the future, and provide acknowledgements.

2 Overall Bridges Architecture

The architecture of the Bridges infrastructure is depicted in Fig. 2. A key component of this architecture is the Data Hub which represents both a local, DB2 based, data repository, and data made available via externally linked data sets (through Information Integrator federated views as described in section 2.2). These data sets exist in different remote locations with differing security requirements. Some data resources are held publicly whilst others are for usage only by specific CFG project partners, or in some instances, only by the local scientists. It is especially important that local security issues are considered. Hence this architecture assumes the existence of multiple different institutional firewalls. The Data Hub itself makes use of two key technologies: OGSA-DAI and IBM's Information Integrator.

2.1 Grid Enabled Data Access and Integration Solutions

The Grid community is currently developing appropriate specifications for data access and integration in a Grid environment through the Data Access and Integration Service working group [20] at fora such as the Global Grid Forum [21]. Much of this work is driven by results from the OGSA-DAI project [3] and the recently funded follow up project, Data Access and Integration 2 (DAIT) [3]. OGSA-DAI/DAIT is a collaborative programme of work involving the Universities of Edinburgh, Manchester and Newcastle, the National e-Science Centre, with industrial participation by IBM and Oracle. Their principal objective is to produce open source database access and integration middleware which meets the needs of the UK e-Science community for developing Grid and Grid related applications. Its scope includes the definition and development of generic Grid data services providing access to and integration of data held in relational database management systems, as well as semi-structured data held in XML repositories.

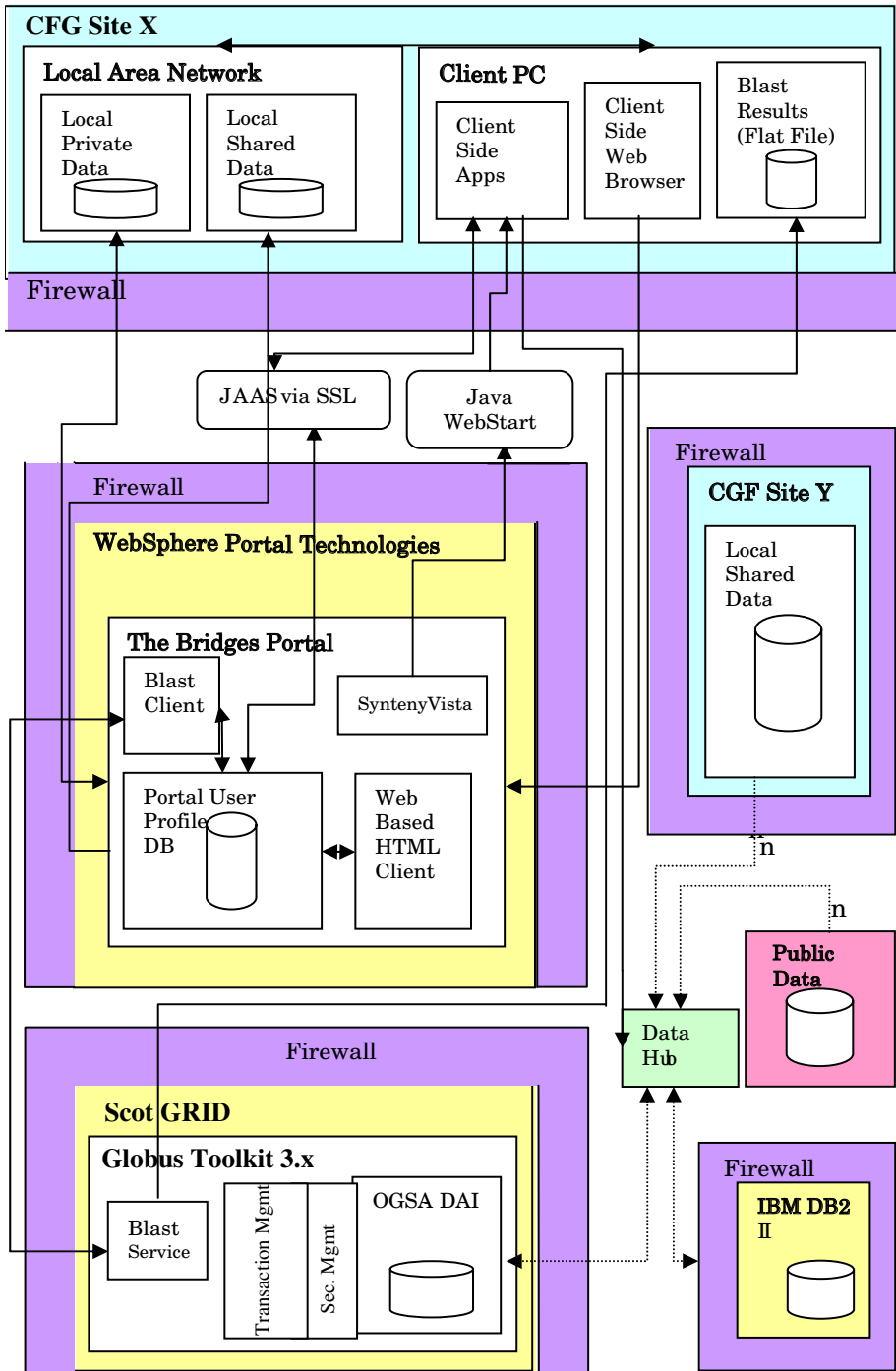


Fig. 2. Overall Bridges Architecture

OGSA-DAI have focused upon making these data resources available within an OGSA compliant architecture. The OGSA-DAI Grid services themselves provide the basic operations that can be used to perform sophisticated operations such as data federation and distributed queries within a Grid environment, hiding concerns such as database driver technology, data formatting techniques and delivery mechanisms from clients. This is achieved by the provision of a Grid-enabled middleware reference implementation of the components required to access and control data sources and resources. OGSA-DAI itself can be considered as a number of co-operating Grid services. These Grid services provide a middleware layer for accessing the potentially remote systems that actually hold the data, i.e. the relational databases, XML databases or, as planned for the near future, flat file structures. Clients requiring data held within such databases access the data via the OGSA-DAI Grid services. The precise functionality realised by OGSA-DAI is described in detail in the Grid Data Service Specification [22]. A typical scenario describing how this functionality might be applied to find, access and use (remote) data sets involves a persistent DAI Service Group Registry (made available via a Grid services hosting container such as Apache Tomcat [23]) offering persistent service factories (used for creating services to access and use specific data resources). Clients would contact the DAI Service Group Registry to find out what data sets are available, and once a required data source was found, create an instance of the Grid data service (via the appropriate factory) that would give access to this resource. The client can then issue queries (submit *Perform* operations via XML documents) to this Grid data service which extracts the queries and submits them to the appropriate databases, e.g. as SQL queries, before results are finally returned in XML documents. Extensions to this scenario to have multiple Grid data services supporting multiple, parallel queries executing through a given client query are possible.

2.2 Commercial Data Access and Integration Solutions

Information Integrator – which was previously known as DiscoveryLink – has been developed to meet the challenge of integrating and analyzing large quantities of diverse scientific data from a variety of life sciences domains. IBM Information Integrator offers single-query access to existing databases, applications and search engines. The Information Integrator solution includes the combined resources of Information Integrator middleware and IBM Life Sciences services. Using this software, IBM Life Sciences services can create new components that allow specialized databases—for proteomics, genomics, combinatorial chemistry, or high-throughput screening—to be accessed and integrated quickly and easily. This is depicted in Fig. 3. At the far right of Fig. fig:ibm-ii-dai are the data sources. Information Integrator talks to the sources using wrappers, which use the data source's own client-server mechanism to interact with the sources in their native dialect. Information Integrator has a local catalogue in which it stores information (metadata) about the data accessible (both local data, if any, and data at the backend data sources). Applications of Information Integrator

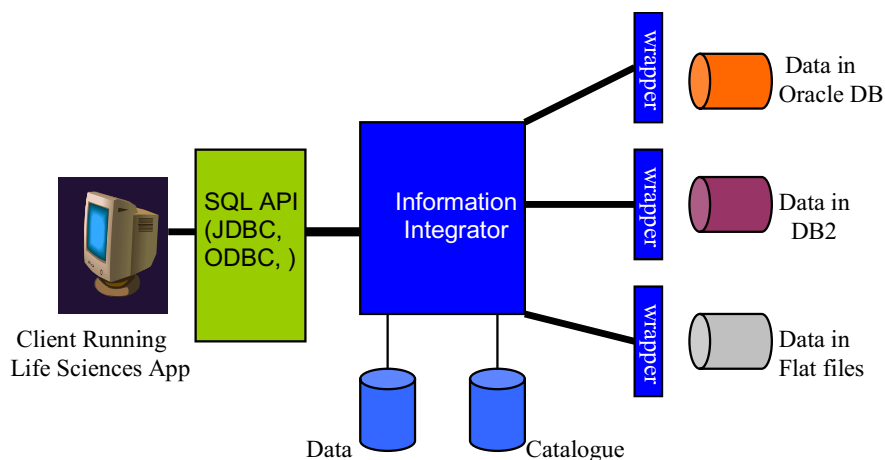


Fig. 3. IBM Information Integrator Data Access and Integration

manipulate data using any supported SQL API, for example, ODBC or JDBC are supported, as well as embedded SQL. Thus an Information Integrator application looks like any normal database application.

OGSA-DAI and Information Integrator have broadly similar aims: to connect distributed heterogeneous data resources. It is important that these two offerings are compared in a realistic life science environment. BRIDGES investigations will provide valuable information on the benefits of both of these solutions which will inform the wider Grid and life science communities. Currently the Information Integrator product for example requires programmatic access to different data repositories. This is not always the case - indeed it is normal for the life science public resources to *not* offer programmatic APIs where for example SQL based queries can be issued. Instead, these resources will generally offer only a web based front end for query submission, or make available their databases as compressed downloadable files. Similarly, open issues are being discovered with the current OGSA-DAI implementation, e.g. the ability to query resources offered as flat files, e.g. bioinformatics resources such as SWISS-PROT, and perform distributed joins across multiple remote databases.

The focus of OGSA-DAI and Information Integrator is primarily upon access and integration of data and not specifically upon security concerns. Security in the context of the Grid is an area that is currently receiving much attention since it is a crucial factor in the wider uptake of the Grid. There are numerous standards under development addressing aspects of security [10, 24]. Within BRIDGES we are considering two security aspects: authentication and authorisation. These are of course considered in conjunction with existing best practice security, e.g. firewalls.

3 Security Considerations

Authentication is widely recognised as being only a starting point in establishing the security of a given Grid based system [3]. Authentication allows establishment of the identity of Grid users. The UK e-Science community has established a public key infrastructure (PKI) based upon X.509 certificates [2] for authentication which are issued through a central Certificate Authority (CA) at Rutherford Appleton Laboratories (RAL) in the UK [13]. These certificates are used to maintain a strong binding between a user's name and their public key when accessing remote Grid resources. It is recognised [3] however, that this approach to certification is unlikely to scale to a wider community, e.g. once Grid technologies are rolled out to the (life sciences) masses.

In the context of the Grid, X.509 based certificates are used to support the establishment and management of virtual organisations (VOs). Currently, however, existing solutions to establishing VOs do not adequately address the security needs of VO members. A fundamental requirement in establishing a VO is to ensure that efficient access control is achieved. Access control is usually done by comparing the authenticated name of an entity to a name in an Access Control List. In the UK e-Science Level 2 Grid [25] for example, which is based upon Globus toolkit version 2 [17], statically maintained "gridmap files" are used to limit who may or may not gain access to remote resources. This is achieved through so called Grid *gatekeepers*. This approach lacks scalability, manageability and does not meet the needs of dynamicity inherent to VOs. It is also limited in the level of granularity of the security model, e.g. in supporting fine grained authorisation to shared resources by potentially, dynamically changing collaborating VO members. We note that the CFG VO is unlikely to be especially dynamic however.

To improve this situation, authentication should be augmented with authorisation capabilities, which in this context can be considered as what Grid users are allowed to do on a given Grid end-system. This "what users are allowed to do" can also be interpreted as the privileges users have been allocated on those end-systems. The X.509 standard [2] has standardised the certificates of a privilege management infrastructure (PMI). A PMI can be considered as being related to authorisation in much the same way as a PKI is related to authentication. Consequently, there are many similar concepts in PKIs and PMIs. An outline of these concepts and their relationship are discussed in detail in [6].

A key concept from PMI are attribute certificates (ACs) which, in much the same manner as public key certificates in PKI, maintain a strong binding between a user's name and one or more privilege attributes. The entity that digitally signs a public key certificate is called a Certification Authority (CA) whilst the entity that signs an AC is called an Attribute Authority (AA). The root of trust of a PKI is sometimes called the root CA – which in terms of the UK e-Science community is given by the Grid Support centre at RAL [13]. The root of trust of the PMI is called the source of authority (SOA). CAs may have subordinate CAs whom they trust and to which they delegate the powers of authentication and certification. Similarly, SOAs may delegate their powers

of authorisation to subordinate AAs. If a user needs to have their signing key revoked, a CA will issue a certificate revocation list. Similarly, if a user needs to have authorisation permissions revoked, an AA will issue an attribute certificate revocation list (ACRL). Typically, a given users' access rights are held as access control lists (ACLs) within each target resource. In an X.509 PMI, the access rights are held within the privilege attributes of ACs that are issued to users. A given privilege attribute within an AC will describe one or more of the user's access rights. A target resource will then read a user's AC to see if they are allowed to perform the action being requested.

The Privilege and Role Management Infrastructure Standards Validation (PERMIS) [15,16] is a role based authorisation infrastructure that realises a PMI – indeed the PERMIS project built and validated the world's first X.509 attribute certificate based authorisation infrastructure. Role Based Access Control (RBAC) models have been designed to make access control manageable and scalable [8]. To cater for RBAC solution for many applications, the PERMIS access control module has been developed [6,7]. It is a standards-based Java API that allows developers of resource gateways (gatekeepers) to enquire if a particular access to the resource should be allowed. PERMIS RBAC uses XML based policies defining rules, specifying which access control decisions are to be made for given VO resources [9]. These rules comprise:

- definitions of subjects that can be assigned roles;
- definitions of Source of Authority (SOA) - trusted to assign roles to subjects;
- definitions of roles and their hierarchical relationships;
- definitions of what roles can be assigned to which subjects;
- definitions of targets that are governed by the policy;
- the conditions under which a subject can be granted access.

The roles are assigned to subjects by issuing them with a standard X.509 Attribute Certificate [2]. The PERMIS team are currently working closely with the Globus team to design a standard Security Assertion Markup Language (SAML) [17] interface to any authorisation infrastructure. This will allow Grid applications to plug and play any authorisation infrastructure. As a result, the BRIDGES project has agreed to work with the PERMIS team and provide a rigorous investigation of security authorisation in a Grid biomedical context. Currently PERMIS has been extended with the SAML API and work is ongoing to extend Globus Toolkit version 3 with a similar API [21]. The expected date for release of the extended GT3 is April 2004. Currently the BRIDGES team is involved in defining suitable XML based policies suitable for the security authorisation requirements of the CFG project consortia, and identifying policy decision and enforcement points to be used when accessing the Grid services and associated CFG specific data sets. We are also involved in discussions in how remote CFG sites might make available their data sets in as secure a manner as possible, without compromising their local security requirements, e.g. through restricted and controlled opening of firewalls. Avenues incorporating existing solutions like *ssh/scp* are being explored.

In addition to authentication and authorisation security aspects, a key requirement of the CFG scientists is related to privacy. Privacy relates to the use of data, in the context of consent established by the data owner. There is little prior art in privacy Grid science, although there is useful UK background in privacy including hospital systems [18]. Web based standards such as P3P [19] may contribute to only a small fraction of the necessary security mechanisms. The area of privacy of biomedical data will be investigated as work on BRIDGES evolves.

4 Portal Technologies

There are various possibilities available for hosting the services to be made available to the CFG scientists. Given that user friendliness is a key aspect, development of a project portal was made. This portal should provide a personalisable environment that the scientist is offered to explore all of the (Grid related) software, data resources and general information associated with the BRIDGES, and hence the CFG projects. Portals in general offer several key advantages as a hosting and delivery mechanism for Grid services. They are:

- Highly flexible and extensible solutions
- Support content and information delivery suitable to users role
- Standardized look and feel across application suite
- Single entry point for services, data resources

Arguably the most mature portal technology on the market and the market leader is IBM WebSphere Portal Server, which has been used to develop the BRIDGES portal, although we note that several other solutions were also investigated including GridSphere [26] and the Commodity Grid toolkit [27]. WebSphere Portal Server runs as another layer on top of the highly developed WebSphere Application Server. Since this provides a fully functional enterprise Java hosting environment it is possible to deploy a Java based Grid service instance within the same virtual machine container.

The BRIDGES portal is shown in Fig. 4. This portal provides an integrated and personalisable environment through which the scientists have access to the various Grid services that they need. This will include Grid data services for the various information repositories of interest to the scientists; Grid services for visualisation of genomic data sets, and Grid services for analysis of genomic data sets. Currently the portal supports Grid enabled visualisation and analysis tools. Work is on-going in development of the Grid data services making use of the Data Hub and hence remote federated data sets.

Integral to the portal is security. The scientists have been issued (by the UK e-Science Certification Authority) with X.509 certificates which are embedded in their browsers. Depending upon the role of the portal user (e.g. scientist, systems administrator, principal investigator etc) the X.509 certificate is used to limit what services the portal user sees and subsequently is allowed to invoke. Of course only certificates for scientists involved with the CFG virtual organisation

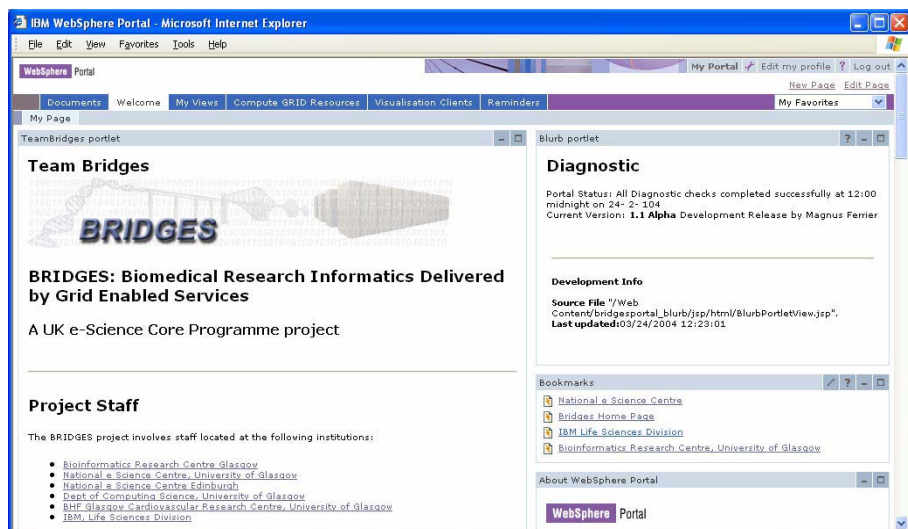


Fig. 4. BRIDGES Portal

are recognised, hence non-authorised access to the portal and the services/data sets available there is not possible. A generic mapping between certificates used for authentication and certificates used for fine grained authorisation is currently under development.

5 Grid Service Development

At the time of writing several trial-Grid services have been engineered and made available through the BRIDGES portal. We describe each of these in turn.

5.1 Grid Enabled Synteny Visualisation Services

Synteny is the condition of two or more genes being located on the same chromosome. Of particular interest to the CFG scientists is conserved synteny which may be defined as the condition where a syntenic group of genes from one species have orthologues (similar genes, where the similarity itself can be ascertained through a combination of approaches such as protein sequence similarity, structure, function etc) in another species.

The analysis of conserved synteny between the different organisms (e.g. mouse, rat and human), in combination with quantitative trait loci (QTL) data [28] and microarray experiments, is one of the main methods used by the CFG scientists in investigating hypertension. Their aim is to discover genes responsible for hypertension in rat or mouse organisms and translate these findings into knowledge about the mechanisms for hypertension in human. It should be noted that knowledge of syntenic relationships and of known QTLs between organisms provides

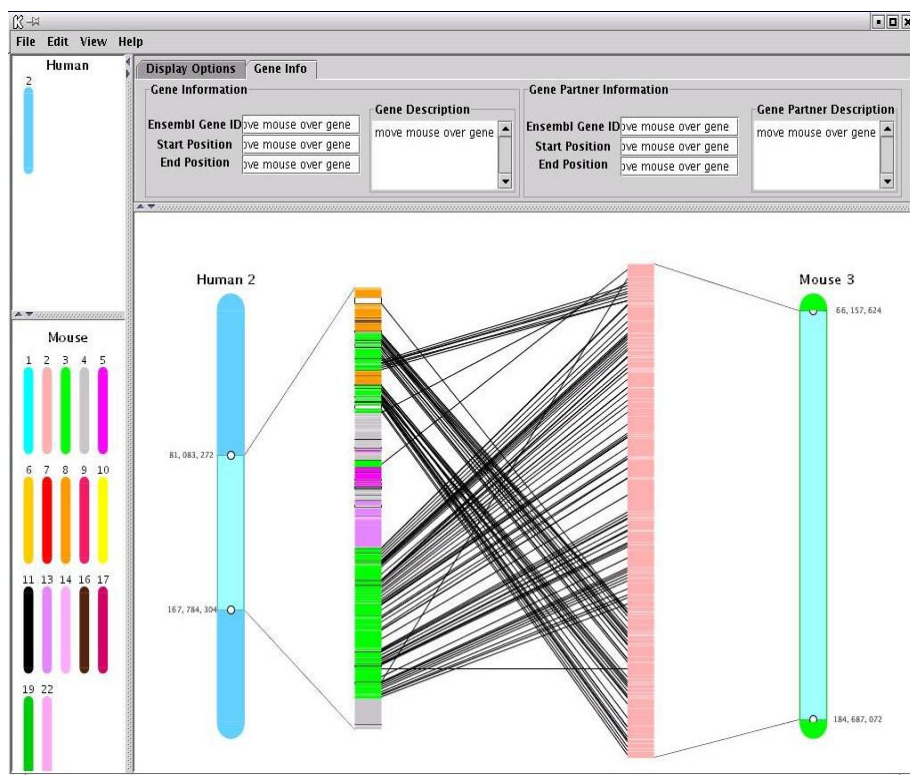


Fig. 5. Grid Enabled Syntenic Visualisation Tool

supporting, but not necessarily guaranteed, evidence about the location and functional role of candidate genes causing hypertension between species.

In displaying conserved synteny, two or more chromosomes need to be shown simultaneously. SyntenyVista was developed for this purpose as shown in Fig. 5. Originally SyntenyVista was developed under the assumption that the relevant chromosome data sets were locally available, e.g. as files on the same machine where SyntenyVista itself was running. This has numerous limitations. Firstly, the user must manually download the relevant data sets, and possibly the complete databases from public resources offering syntenic information, e.g. Ensembl [29]. Secondly, each time the user wants to visualise syntenic data sets, they need to manually check, e.g. by visiting the public data repository, whether a newer more up to date version of their syntenic data set is available. Grid technology offers a mechanism to overcome these restrictions, through automatically accessing and pulling down relevant remote data sets as and when needed. To achieve this SyntenyVista has been augmented with OGSA-DAI capabilities. Specifically, SyntenyVista is now able to access remote syntenic data sources and pull down (cache) them as needed. The portal delivery mechanism for SyntenyVista itself is via Sun's WebStart [30]. The Java WebStart technology simplifies de-

ployment of Java applications and enables launch of full-featured applications through single clicks from web browsers. Upon launching SytenyVista from the portal, automatic checks are done to ensure that the latest version of the software is available, and if not, automatically downloaded and installed.

The Grid enabled version of SytenyVista automatically checks on syntenic data sets that might have been cached already. When these are available, they are loaded (onto the pallet on the left hand side of Fig. 5). Users are then able to drag these onto the main window to see where syteny might be present between different chromosomes. When these data sets are not cached locally, remote resources accessible via OGSA-DAI are accessed and the data pulled down. Currently the Grid data services for finding syntenic data sets are under development, and as such the current implementation uses OGSA-DAI directly to connect to a single syntenic data source (Ensembl).

5.2 Grid Enabling BLAST Services

Bioinformaticians typically need to be able to find similarities between different genomic or protein sequences. The Basic Local Alignment Search Tool (BLAST) [31] has been developed to perform this function. Numerous versions of BLAST currently exist which are targeted towards different sequence data sets and offer various levels of performance and accuracy metrics. Typically full scale BLAST jobs across whole genomes is a highly compute intensive activity. As a result, large scale compute farms are often required.

The ScotGrid computational resource at the University of Glasgow offers precisely such a high throughput compute facility [32]. It is the e-Science resource at the University of Glasgow and represents a consolidation of resources across a variety of research groups and departments. It is used by varieties of scientists across the university and internationally including particle physicists, electronic engineers, computer scientists and bioinformaticians. ScotGrid itself offers a Beowolf Linux cluster with the equivalent of 330 1GHz processors and 15TB disk space comprised of IBM xSeries, Blade server, FAStT500 and Dell and Cisco technologies. It uses the Maui scheduling software [33] which itself is based on OpenPBS [34].

To provide Grid enabled BLAST services accessing and using ScotGrid, it was required that the BLAST software was ported into the Grid environment, i.e. made available as a GT3 based Grid service. The original version of BLAST we used was implemented in C hence this required writing appropriate Java wrappers and exploiting specific GT3 APIs. The current prototype for GT3 BLAST job submission is based on the GT3 core only, and involves a simple wrapper to OpenPBS commands, due to difficulties with the full GT3 installation.

BLAST takes in a search sequence and assumes a target sequence database. This requires that the source and target data sets were staged onto the ScotGrid infrastructure, and the results pulled off once the job itself had completed. It should be noted that in the current implementation, we simply keep a copy of the relevant target database on ScotGrid, however it is expected that this solution will be modified once we deal with more security dependent data sets. Job

monitoring services based on usage of the OpenPBS API have also been implemented. These Grid services and monitoring services are all available through the BRIDGES portal and provide a valuable resource to the CFG scientists.

6 Conclusions

The BRIDGES project began in October 2003 and is engaged in the evaluation of a wide variety of Grid technologies applied to the life science domain. Focus thus far has been oriented towards, mainstream implementations of Grid technology such as Globus toolkit, and its recent web service based version (GT3). GT3 usage has not been without issues however, and often workaround solutions have been necessary. For example, full installation of GT3 with GRAM for job submission/management capabilities was found to be especially problematic due to undocumented operating systems dependencies. Further, since we often required software to run on Windows OS flavours, e.g. for running WebSphere portal software, compromises had to be made between the architecture and design, and the final systems that have been implemented.

The current implementation has provided a proof of concept prototype. The next phase of the work will look in more detail at the key requirements of the CFG scientists. For example, the scientists are especially interested in microarray analysis. Tools and workflows that allow the scientists to take up/down regulated gene names from microarray experiments and garner further information are of special interest. This in turn requires that our Data Hub can connect to relevant data sites to pull down specific information on the genes themselves. The current focus of our Data Hub is on genome databases such as Ensembl [29], GenBank [35]; gene function databases such as LocusLink [36], OMIM [37], transcriptome databases such as Unigene [38], and relevant publications databases such as PubMed [39] and MedLine [40]. Currently, few of these resources provide the necessary programmatic access needed. Instead, they often only offer compressed files that can be downloaded. As a result, a local warehouse is being established and populated. This will make use of both IBM's Information Integrator technology and other Grid services, e.g. for replica location management [41]. This will support automated, dynamic updating of the local repository with remote, public data sets, as and when these change.

There exist numerous other visualisation and analysis tools that the CFG scientists would like to be supported and offered through the BRIDGES portal. These include sequence visualisation tools and multiple alignment tools. Work is currently on-going in Grid enabling these. Similarly, the scientists are keen to use numerous analysis and sequence comparison tools such as clustalw [42] and Smith-Waterman [43]. Their Grid enablement is also under way. We note that not all scientific compute demands are satisfied by farms such as ScotGrid, and often a large SMP machines are required. Where this is the case, large SMP resources offered through the UK e-Science Grid such as Blue Dwarf at Edinburgh University will be made.

Finally a role based authorisation infrastructure meeting the needs of the CFG scientists and their associated is under development. This will give valuable insight into the much needed security concerns raised by rolling out the Grid to the wider (life science) community.

Acknowledgements

This work was supported by a grant from the Department of Trade and Industry. The authors would also like to thank members of the CFG team including Prof. David Gilbert, Prof Malcolm Atkinson, Dr Ela Hunt and Dr Neil Hanlon. Drs Hanlon and Hunt are also acknowledged for their contribution to the original SyntenVista software. Acknowledgements are also given to the IBM collaborators on BRIDGES, notably Drs David White, Andy Knox and Jean-Christophe Mestres. The CFG project is supported by a grant from the Wellcome Trust foundation.

References

1. Cardiovascular Functional Genomics project, <http://www.brc.dcs.gla.ac.uk/projects/cfg/>
2. BioMedical Research Informatics Delivered by Grid Enabled Services (BRIDGES), www.brc.dcs.gla.ac.uk/projects/bridges
3. Open Grid Service Architecture – Data Access and Integration project (OGSA-DAI), www.ogsadai.org.uk
4. IBM Information Integrator, [http://www3.ibm.com/solutions/lifesciences/solutions/Information Integrator.html](http://www3.ibm.com/solutions/lifesciences/solutions/Information%20Integrator.html)
5. E-Science Security Roadmap: Technical Recommendations v0.5, UK e-Science Security Task Force, draft executive summary v0.51
6. ITU-T Rec. X.509 (2000) — ISO/IEC 9594-8 The Directory: Authentication Framework
7. C Adams and S Lloyd (1999), Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations, Macmillan Technical Publishing.
8. Adams, C., Lloyd, S. (1999). "Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations", Macmillan Technical Publishing, 1999
9. Austin, T. "PKI, A Wiley Tech Brief", John Wiley and Son, ISBN: 0-471-35380-9, 2000
10. Grid Security, <https://forge.gridforum.org/projects/sec>
11. L Pearlman, et al., A Community Authorisation Service for Group Collaboration, in Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks. 2002.
12. M Thompson, et al., Certificate-Based Access Control for Widely Distributed Resources, in Proc 8th Usenix Security Symposium. 1999: Washington, D.C.
13. VOMS Architecture, European Datagrid Authorization Working group, 5.9.2002.
14. Steven Newhouse, Virtual Organisation Management, The London E-Science centre, <http://www.lesc.ic.ac.uk/projects/oscar-g.html>

15. D. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure, in Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, Monterey, California, USA. 2002.
16. Privilege and Role Management Infrastructure Standards Validation project www.permis.org
17. P Hallem-Baker, E Maler, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML), OASIS, SAML 1.0 Specification. 31 May 2002. <http://www.oasis-open.org/committees/security/#documents>
18. I. Denley and S.W. Smith, Privacy in clinical information systems in secondary care. British Medical Journal, 1999. 318: p. 1328-1331.
19. Platform for Privacy Preferences (P3P) Project, W3C, <http://www.w3.org/P3P/>
20. Data Access and Integration Services working group, <https://forge.gridforum.org/projects/dais-wg>
21. Global Grid Forum, www.ggf.org
22. Grid Data Service Specification, https://forge.gridforum.org/docman2/ViewCategory.php?group_id=49&category_id=517
23. Apache web site, www.apache.org
24. Web Security Standards, http://www.oasis-open.org/committees/documents.php?wg_abbrev=wss
25. UK e-Science Engineering Task Force, www.grid-support.ac.uk/etf
26. GridSphere Portal, www.gridsphere.org
27. Commodity Grid toolkit, www-unix.globus.org/cog
28. An Overview of Methods for Quantitative Trait Loci (QTL) Mapping, Lab of Statistical Genetics, Hallym University, http://bric.postech.ac.kr/webzine/content/review/indivi/2002/Aug/1_08_index.html
29. EMBL-EBI European Bioinformatics Institute, <http://www.ebi.ac.uk/ensembl/>
30. Sun WebStart Technology, <http://java.sun.com/products/javawebstart/>
31. Basic Local Alignment Search Tool (BLAST), <http://www.ncbi.nlm.nih.gov/Tools/>
32. ScotGrid, www.scotgrid.ac.uk
33. Maui Job Scheduler, <http://www.supercluster.org/maui/>
34. Open Portable Batch System, www.openpbs.org
35. NCBI GenBank, <http://www.ncbi.nlm.nih.gov/Genbank/>
36. NCBI LocusLink, <http://www.ncbi.nlm.nih.gov/LocusLink/>
37. NCBI Online Mendelian Inheritance in Man, <http://www.ncbi.nlm.nih.gov/OMIM/>
38. NCBI Unigene, <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=unigene>
39. PubMed Central Home, <http://www.pubmedcentral.nih.gov/>
40. US National Library of Medicine, <http://www.nlm.nih.gov/>
41. Replica Location Service (RLS), www.globus.org/rls
42. EMBL-EBI European Bioinformatics Institute clustalw, <http://www.ebi.ac.uk/clustalw/>
43. EMBL-EBI European Bioinformatics Institute MPSrch, <http://www.ebi.ac.uk/MPSrch/>

Building a Biodiversity GRID

Andrew C. Jones¹, Richard J. White¹, W. Alex Gray¹, Frank A. Bisby²,
Neil Caithness², Nick Pittas¹, Xuebiao Xu¹, Tim Sutton², Nick J. Fiddian¹,
Alastair Culham², Malcolm Scoble³, Paul Williams³, Oliver Bromley⁴,
Peter Brewer², Chris Yesson², and Shonil Bhagwat³

¹ Cardiff University, School of Computer Science, Queen's Buildings,
5 The Parade, Cardiff CF24 3AA, UK

{Andrew.C.Jones, R.J.White, W.A.Gray, N.Pittas, X.Xu,
N.J.Fiddian}@cs.cardiff.ac.uk

² Centre for Plant Diversity & Systematics, School of Plant Sciences,
The University of Reading, Reading RG6 6AS, UK

{F.A.Bisby, N.Caithness, T.Sutton, A.Culham, P.W.Brewer,
C.Yesson}@reading.ac.uk

³ Department of Entomology,

The Natural History Museum, London SW7 5BD, UK

{M.Scoble, P.Williams, S.Bhagwat}@nhm.ac.uk

⁴ School of Biological Sciences, University of Southampton,
Southampton SO16 7PX, UK

O.Bromley@soton.ac.uk

Abstract. In the BiodiversityWorld project we are building a GRID to support scientific biodiversity-related research. The requirements associated with such a GRID are somewhat different from other GRIDs, and this has influenced the architecture that we have developed. In this paper we outline these requirements, most notably the need to interoperate over a diverse set of legacy databases and applications in an environment that supports effective resource discovery and use of these resources in complex workflows. Our architecture provides an invocation model that is usable over a wide range of resource types and underlying GRID middleware. However, there is a trade-off between the flexibility provided by our architecture and its performance. We discuss how this affects the inclusion of computationally intensive applications and applications that are highly interactive; we also consider the broader issue of interoperation with other GRIDs.

1 Introduction

Biodiversity informatics is developing as a distinct part of the field of bioinformatics. Many bioinformatics researchers concentrate on the proteomic and functional aspects of molecular sequence data. In contrast, the emphasis of biodiversity informatics is on the diversity of organisms and the evidence available to explore their interactions and relationships, their past history, and what might happen in the future. Molecular sequence data is still relevant, as one of many

factors, but in this context one is typically interpreting the data as historical evidence of phylogenetic relationship, rather than interpreting it functionally. Many relevant, diverse kinds of data sets have been assembled, and many analytic tools have been created for specific purposes. It is now being recognised that new scientific questions could be investigated if it were possible to interoperate over such resources, assembling workflows that use them. In this paper we discuss some aspects of the BiodiversityWorld (BDW) project, which is building a biodiversity GRID that provides the interoperation we require, and we concentrate particularly upon the architecture we are adopting and the implications of this for the ways in which BDW applications may be structured.¹

We commence by describing the project background, providing information about the application domain and relevant GRID developments. We then consider the distinct requirements associated with a biodiversity GRID, following which we explain the architecture that we have adopted to meet our requirements, and its implications. In the concluding section we describe the current project status and outline the ways in which we intend to develop BDW in the future.

2 Background

2.1 Application Domain

In the BDW project we have selected three exemplar biodiversity research areas to drive the development of our biodiversity GRID. These areas have been selected to fall within the team's areas of expertise, and to be representative of tasks that one might typically wish to perform using a biodiversity GRID. These research areas are as follows:

1. *Bioclimatic and ecological niche modelling.* This is a two step procedure. First, a *climate preference profile* is produced by cross-referencing the known localities of a species with present day climate data. This climatic preference is then used to locate other areas where such a climate exists indicating areas which are climatically suitable for the species, using present day climate data (e.g. to identify areas under threat from invasion by invasive species), or using climate model predictions for either the future or the past (e.g. to predict the possible effects of global climate change on the species distribution). Relevant existing modelling algorithms for this research include GARP (Genetic Algorithm for Rule-set Production) and CSM (Climate Space Model); we also need to be able to visualise the climate preference profiles and species distributions. Suitable species distribution data is required, and a catalogue of life such as that provided by Species 2000² can help in selecting suitable species names for querying the distribution data.

¹ BDW is one of a number of current e-Science projects funded by the UK Biotechnology and Biological Sciences Research Council (BBSRC). For an overview of these projects see [1].

² <http://www.sp2000.org/>

2. *Biodiversity modelling and the prioritisation of conservation areas.* This task involves using species distribution data in order to produce a species richness map, and then using it as a basis for proposing priority areas for biodiversity conservation. As a test study, we are using a database of Canadian butterflies, provided by the Canadian Biodiversity Information Facility (CBIF)³, and we are using Worldmap⁴ to perform the biodiversity modelling.
3. *Phylogeny and palaeoclimate modelling.* The purpose of phylogenetic analysis is to reconstruct the most likely model of historical relationships among species and to use this to explore scenarios that have led to the diversity we see. Typical applications of such an approach are to integrate multiple genome data to interpret certain kinds of changes (morphological or cytological) during evolution [2]. Recent developments in phylogenetic reconstruction, and calibration of phylogenetic trees against time, allow the exploration of the role of climate change in diversification of species through palaeohistory. The intention is to operate two converging streams of activity once the taxon set⁵ has been established. The purpose of stream one is to automate data gathering from the global DNA sequence databases EMBL⁶/GenBank/DBJ, enable sequence alignment and convert this to files ready for data analysis. We wish to perform analysis through a choice of proprietary software, leading to the output of phylogenetic trees. The purpose of stream two is to gather distribution data for these taxa and fit climate models to each taxon. These streams need to converge so that newly developed software can optimise the climate variables across the phylogenetic tree to give palaeoclimate scenarios through evolutionary time. When fully implemented, these scenarios can be cross-referenced to global palaeoclimate models for comparison. Simultaneous optimization of DNA sequence-based phylogeny, palaeoclimate and geographical data will allow, for the first time, explicit scientific interpretation of the role of climate in the evolution of biodiversity.

At present, tasks such as those enumerated above require substantial manual work on the part of the scientist. This is perhaps particularly evident in task (3). Current working practice is generally to bring prepared datasets to a software package (e.g. SAS for statistical analysis; PAUP for phylogenetic analysis) and to use the suite of proprietary analytical options made available in that package. Using a sequence of analytical tools selected from different packages is usually difficult as the data needs to be prepared in different ways, and most relevant software packages were not designed with this kind of use in mind.

A workflow-based approach to link tools in an analytic pathway, such as we adopt in BDW, provides a means of planning and executing complex tasks within a uniform environment. Furthermore, changes to these tasks (e.g. to use a

³ <http://www.cbif.gc.ca/portal/digir-toc.php>

⁴ <http://www.nhm.ac.uk/science/projects/worldmap/index.html>

⁵ Such as a set of species.

⁶ <http://www.ebi.ac.uk/embl/>

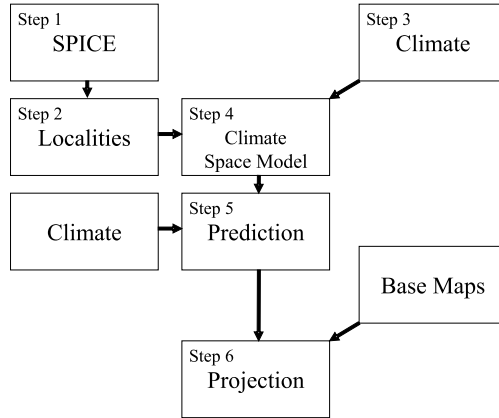


Fig. 1. A bioclimatic modelling workflow

different analytic tool) can easily be achieved. To illustrate the complexity of the above tasks, consider one aspect – phylogenetic analysis. This may include multiple sequence alignment, phylogenetic inference by several competing methods, followed by a comparison of results. This is a tedious task at present, involving up to a dozen competing and incompatible packages. In BDW we are making such packages available in a mutually compatible way, as ‘resources’ that can be discovered and assembled into workflows meeting the user’s requirements.

To further illustrate typical biodiversity-related tasks, consider Fig. 1. This figure illustrates a bioclimatic modelling workflow, and is one of the workflows we have currently implemented in BDW. In step 1, a scientific name is submitted to the SPICE catalogue of life and a list of alternative names for the chosen species is retrieved; in step 2, these names are used to retrieve distribution maps for specimens having any of these alternative names; in step 3, *climate surfaces* describing climate world-wide are retrieved; in step 4, these climate surfaces and distribution maps are used to generate a model of climatic conditions where the chosen species is currently found; in step 5, climate surfaces are used in conjunction with the generated climate space model to predict suitable regions for the species of interest, and in step 6, this prediction data is overlaid on a map of the world or some geographical region. Note that in step 5 we might use different climate surfaces from those used in step 3: for example, we might use predicted climate to assess the possible effects of climate change.

2.2 The GRID

Much of GRID research has focussed upon making large amounts of computational power readily available (see, e.g., [3]). But another important aspect of building a usable GRID is the ability to discover and reason with information in sophisticated ways. This is implied by the three levels (Data; Information; Knowledge) proposed by Jeffrey [4] and, to a certain extent, by the Knowledge

Grid [5]. This need has been recognised at a rudimentary level by the introduction of the Open Grid Services Architecture (OGSA) [6] and, more recently, by the Web Services Resource Framework (WSRF)⁷. But there is still much room for research into the effective use of metadata and ontologies within a GRID environment, particularly in our application area. Also, GRID software such as Globus is evolving rapidly, without maintaining full backward compatibility: this is a significant problem if a GRID that includes a large number of heterogeneous resources and tools is to be maintainable. Some projects are looking at the metadata & ontology problem in a context partially related to BDW (such as myGRID⁸). But one of the major goals for BDW is to address all the problems identified in the present section sufficiently well for it to be possible for an extensible biodiversity GRID to be developed and effectively maintained.

3 Biodiversity GRID Requirements

The nature and diversity of the resources to be used, and the kinds of problems that we wish to help scientists solve, present challenges to the effective management and use of resources within a biodiversity GRID. In relation to data sources, notable requirements are as follows:

- For some of the tasks of interest, such as bioclimatic modelling, our GRID must facilitate the use of a wide range of different kinds of data in a seamless manner.
- We need to support data source heterogeneity: for example, various database management systems have been used – indeed, several data sources are not held in database management systems, but merely in files – and data representation and structure will vary. The services which database owners are willing to publish will also vary: some databases may provide direct querying against a known schema; others provide only a small set of operations that map onto ‘canned queries’. Many of the databases of interest have been built for very specific purposes, which may be very different from how we would like to use them in BDW, and these purposes have influenced the design, both in terms of what information is stored and the organisation of this information. This contrasts sharply with many important data sets in other disciplines, which are frequently stored in agreed formats in public repositories.
- Data may vary in ways relating to scientific opinion. For example, not all scientists may use the same scientific name to refer to a given species, and scientific opinion may differ as to the circumscription (i.e. the boundaries of variation) of a species. So resources holding knowledge to assist in accurate retrieval are needed.

⁷ <http://www.globus.org/wsrf/>

⁸ <http://www.mygrid.org.uk>

- Some data are sensitive, such as detailed distribution information for rare species. Precise, dependable access control is therefore required to ensure that users see only the data to which they are entitled to have access.

In relation to analytic tools, various related requirements arise, including:

- Certain tools were written to solve very specific problems and assume data input/output in proprietary formats. We need to be able to accommodate these formats within our interoperation framework, applying appropriate transformations where necessary.
- We wish to make use of ‘packages’ such as those described in Section 2.1, in which a task that may be required is deeply embedded in a framework intended primarily, or solely, for access via a proprietary user interface.
- We want to support interaction with analytic tools of a range of types (command-line based; tools provided as program libraries; commercial and scientific packages as described in the previous requirement; open source code; etc.) within our architecture where possible. Where this is impracticable (e.g. a system may assume that the user will interact via a GUI on a particular platform and the effort involved in building software to simulate such a user is unjustifiable), sensible ways of using such external tools must be devised.

A further requirement is that it is essential for the composition of workflows, which bring resources together in flexible and powerful combinations, to be fundamentally efficient and straightforward for biodiversity application users to undertake readily themselves. This has implications for the HCI aspects of the user interface design (which we intend to discuss in a future paper), but also for the architecture of BDW, especially in regard to the provision of suitable metadata.

Thus, it is necessary to provide an architecture and middleware that support the above requirements. Moreover, because of the investment of time that is inevitably required in order to make these resources available within our GRID, the middleware must insulate our GRID from changes that may occur as the underlying GRID software (such as Globus) evolves.

4 Meeting the Requirements

4.1 BiodiversityWorld Architecture

An early proof-of-concept system was developed by some of the authors of this paper in order to explore the possibilities for using Globus software as the basis for a biodiversity GRID. The resulting software, GRAB (GRid And Biodiversity), was capable of performing some tasks relating to bioclimatic modelling, but in a fixed workflow with a limited set of resources. In Globus Toolkit 2 it was found that the facilities provided did not lend themselves well to implementation of services of the kinds needed in a biodiversity GRID [7]. These

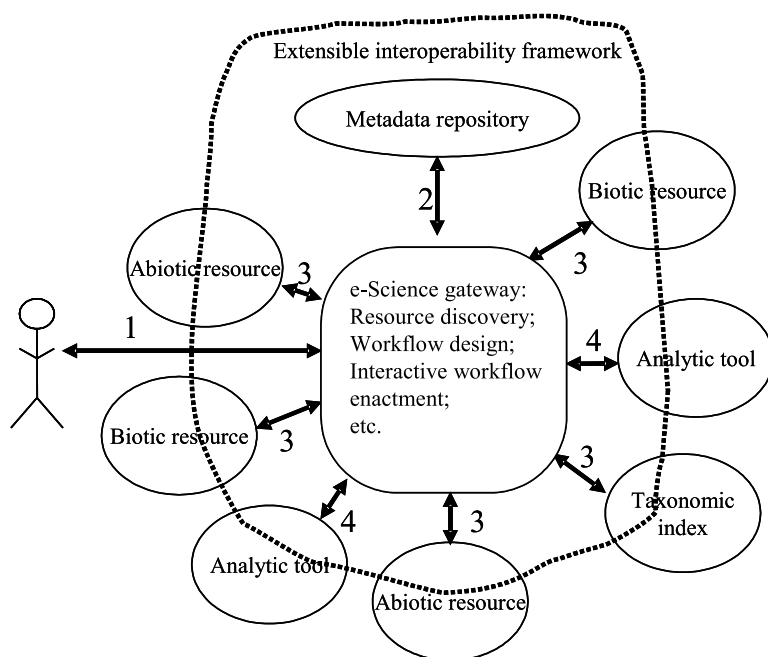


Fig. 2. BiodiversityWorld environment. 1: User exploits a wide range of facilities provided by the e-Science gateway, which can be extended to accommodate new resources, etc; 2: Resource discovery makes use of a metadata repository, in which resources are registered; 3: Resources of importance to Biodiversity Informatics include a Taxonomic Index (catalogue of names); biotic resources (containing information about organisms), and abiotic resources (containing other relevant information, e.g. climate data); 4: Analytic tools are also needed to form a problem-solving environment for biodiversity informatics

limitations, the requirements mentioned in the previous section, and the evolving nature of GRID software identified in Section 2.2, implied that a middleware layer providing a suitable invocation mechanism and insulating resources from the underlying GRID was required to provide a suitable interoperation framework for the subsequent BiodiversityWorld project. The environment in which resources are made available, and the elements of a typical interaction with the system, are illustrated in Fig. 2. The architecture we have adopted in order to achieve this interoperability is illustrated in Fig. 3. It has the following elements:

- The *abstraction layer*, which we refer to as the *BiodiversityWorld-Grid Interface* (BGI). This layer specifies a Java software interface that all resources must conform to, but the interface between this layer and the GRID is custom-built for whatever GRID software is being used at the time. The BGI provides an invocation mechanism and assumes that data is serialised for transmission, typically using XML;

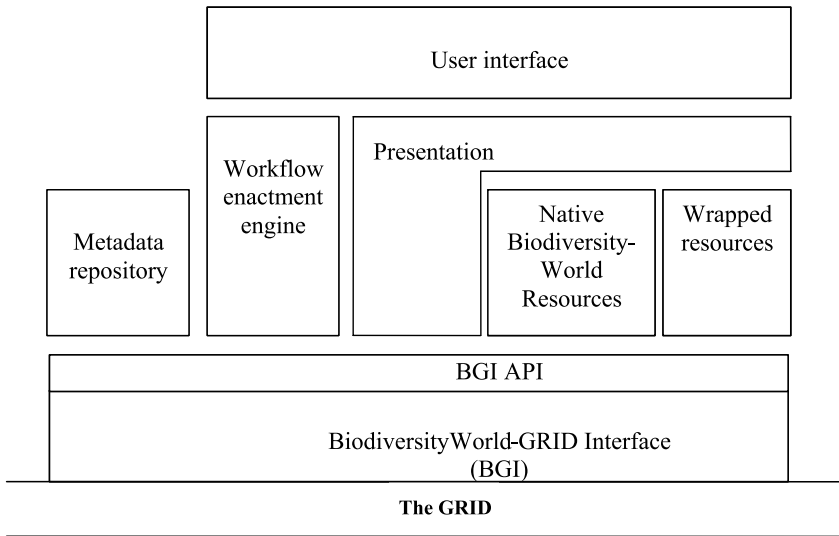


Fig. 3. BiodiversityWorld architecture

- *Resource wrappers*, where necessary, to wrap the resources to conform to the BGI and publish metadata concerning operations and associated data types supported by each resource, the nature and location of each resources, etc. It is a wrapper's responsibility to invoke the relevant operations on a resource, de-serialising the data that is to be used by that resource;
- A *metadata repository*, which holds this published metadata in order to support resource discovery and resource use within workflows. This repository is accessed via the BGI, as other resources are;
- A *workflow enactment engine* that can communicate via the BGI with resources as required;
- *Presentation* modules, and
- A *user interface* which provides facilities for workflow design, visualisation of results, etc.

This architecture is intended to be suitable for the kinds of scenarios envisaged for BDW. More detail of the BDW architecture may be found in [8, 9]. In the present paper we shall now concentrate on some important implications of having adopted this architecture that were not discussed in our earlier publications.

4.2 Implications of the BiodiversityWorld Architecture

There are a number of implications of the BDW architecture, with regard to how resources may be wrapped for use in BDW; how we may structure applications that are computationally intensive; how to support highly interactive applications; and how we can interoperate with other GRIDs. We shall consider each of these in turn.

Wrapping resources. There is a variety of ways in which resources can be made compliant with the BGI. As originally conceived, one of the main challenges for BDW was to make interoperation possible among resources that already existed and were highly heterogeneous. For these kinds of resource, wrapping them to conform to some uniform specification is essential in order to make the BDW architecture manageable. For resources to be wrapped using Java we have split wrappers into two parts with a standard interface defined between them. The resource-facing wrapper component performs all the resource-specific wrapping tasks; while the GRID-facing wrapper component performs the tasks specific to our chosen GRID implementation. This latter component is identical for all resources wrapped for a given GRID infrastructure; it could be replaced by instances of a component supporting a new GRID infrastructure if required, without necessitating changes to the resource-facing wrapper components.

Further re-usability has been achieved by noting that one particular class of resource is those resources that are normally invoked from the command line. At present we have implemented a number of distinct wrappers for these resources, but this is an obvious example of where some wrapper genericity can be achieved: we have implemented, and are currently testing, such a wrapper. But another important class of resource is those that are being newly implemented. We see this as an opportunity to implement software that directly conforms to the BGI, with no wrapping as such being needed. In contrast with the foregoing resource types, an alternative way of implementing resources is to design them to operate with some specific *implementation* of the BGI. For example, an OGSA-based GRID service could be incorporated if an OGSA version of the BGI were being used. This has the advantage that, if necessary, the entire resource could be implemented in some language other than Java; it also has the obvious disadvantage that the resource will only be compatible with one specific implementation of the BGI. But the main point arising from these differing resource types is that we are not restricted to a single way by which resources can be integrated into BDW: we can choose the most appropriate technique for individual circumstances.

Computationally intensive applications. It will be clear that our architecture has communications overheads when compared with architectures that make direct use of efficient mechanisms (for example, using GridFTP for data transfer). We have not as yet assessed the precise magnitude of the performance penalty, but it is obvious that this penalty will be significant for distributed applications requiring high bandwidth. But this performance penalty is a consequence of an architectural design which has benefits that are important in a situation like ours, where interoperation among a large number of heterogeneous resources is required, and maintainability of the system as GRID software evolves is important. Also, though some tasks relating to biodiversity studies are computationally intensive, they tend to be self-contained – for example, climate surface generation for bioclimatic modelling can be carried out as an independent task, given appropriate input data. Accordingly we assume that any computationally intensive task will be carried out within a *single* resource, as seen by the BDW system,

even if the resource is actually distributed over a cluster of computers. This approach is not dissimilar to that taken in the WhyWhere⁹ project, although it should be understood that WhyWhere is narrower in focus than BDW – it is a system specifically for reasoning with specimen location data and environmental data. In WhyWhere, high-performance GRID technologies are used to process the environmental data, but a separate mechanism is used to access the species data; hence, as in BDW, the high-performance computing tasks are partitioned off. If this approach should prove inadequate in some special circumstances, we shall have to consider some special high-throughput extension to our architecture, something we may also have to consider for highly interactive applications as we shall now see.

Highly interactive applications. In most cases, interaction with a resource within BDW can be fairly coarse-grained – typically a small number of operations will be invoked on any given resource within an individual workflow, and each operation performs a fairly substantial task. But there are still two important problems to address:

- How can we make use of stand-alone applications that were intended for use via an integrated GUI on a particular platform, if we have no access to the source code or to some programmatic interface to the application? And
- If an extended sequence of interactions with a given resource is envisaged, where data will need to be transferred between the client and the resource rapidly in order to achieve good response, how can this be achieved?

We see a number of possibilities for dealing with these problems:

- A stand-alone application could be wrapped for use within BDW in the usual way, the wrapper in this case being fairly complex in that it would have to simulate user interactions with a running version of the application.
- The stand-alone application could either be installed on the user's machine, or controlled remotely using software such as VNC, and conventions for depositing data and retrieving data by BDW and the application software could be established. For example, BDW may gather all the data to be used by an analytic tool, store it in specified files on the user's machine, and then the user runs the analysis tool him/herself.
- Interaction with resources could be achieved using the above technique, or a commonly-needed set of interactive tasks could be implemented on the client side, within the BDW user interface itself.
- Another way to provide high throughput is to extend BDW with an alternative data transfer mechanism dependent directly on the underlying GRID software in use at the time. Effectively this would allow programmers to by-pass the layered architecture of BDW and, in so doing, to gain more efficiency. This would be at the expense of portability, but it may be considered worthwhile – for a small, selected number of resources – to accept this restriction.

⁹ http://biodi.sdsc.edu/ww_home.html

None of these approaches is perhaps ideal, but the general point that this section illustrates is that any given GRID architecture has its strengths and weaknesses, and that if (as in our case) performance is sacrificed for interoperability, on the grounds that performance is not usually the main problem in a given application domain, there are still work-rounds which can address the performance issue where necessary.

Interoperation with other GRIDs. An attractive aspect of our architecture is that it provides a means of using resources from other GRIDs: they can be wrapped as resources for our own GRID. To some extent it would also be possible for other GRIDs to interoperate with a given *deployment* of ours (e.g. an OGSA- or WSRF-based version), because the resources are published as GRID services. These kinds of interaction are important, because other projects – such as myGRID – are developing resources that are of interest to biodiversity research and, conversely, facilities available through BDW – such as the catalogue of life – hold information that is of benefit for effective retrieval of bioinformatics information.

5 Current Status and Conclusions

At the time of writing we have implemented a number of prototypes, mostly concentrating on the bioclimatic modelling task. We have two implementations of the BGI, namely a Java RMI-based version and an OGSA-based version. This plurality of BGI implementations gives us some confidence that the BGI is indeed ‘GRID’ infrastructure-independent. From the point of view of the BDW project either of these implementations is acceptable, although the OGSA-based version has the advantage for interaction with other GRID-based projects that it is based on a recognised standard.

A representative range of bioclimatic modelling-related resources have been incorporated into the system: some of these, such as the Species 2000 catalogue of life, are of general use within BDW, rather than being specific to bioclimatic modelling. We have been using a proprietary workflow enactment engine, but are currently in the process of adapting Triana¹⁰ to meet our needs. An initial prototype using Triana is now available. An embryonic metadata repository is currently in place. We are currently in the process of incorporating resources to support the other two scenarios listed in Section 2.1, so that scientific research using BDW in all three application areas can commence soon.

In this paper we have discussed the need for a biodiversity GRID, to support biological research that is generally outside the normally-recognised scope of bioinformatics. We have seen that the requirements for a biodiversity GRID have led to a distinctive design, intended to insulate large numbers of resources from infrastructure change, and that this contributes to the maintainability of our system. We have also seen that our architecture makes interoperability with

¹⁰ <http://www.triana.co.uk/>

other GRIDs feasible: BDW is therefore a step towards the realisation of the concept of a ubiquitous GRID.

Acknowledgements

The BiodiversityWorld project is funded by a research grant from the UK Biotechnology and Biological Sciences Research Council (BBSRC). The GRAB project was funded by a grant from the UK Department of Trade and Industry (DTI).

References

1. Gray, W., Thompson, C.: Bioinformatics and eScience. In Cox, S.J., ed.: Proc. UK e-Science All Hands Meeting, Nottingham, UK, EPSRC (2003) 66–71
2. Bakker, F., Culham, A., Hettiarachi, P., Touloumenidou, T., Gibby, M.: Phylogeny of *Pelargonium* (Geraniaceae) based on DNA sequences from three genomes. *Taxon* **53** (2004) 17–28
3. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco, CA (1999)
4. Jeffery, K.G.: GRIDs in ERCIM. ERCIM News (2001)
5. Cannataro, M., Talia, D.: The knowledge grid. *Communications of the ACM* **46** (2003) 89–93
6. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the Grid: An open Grid services architecture for distributed systems integration. (2002) <http://www.globus.org/research/papers/ogsa.pdf>.
7. Jones, A.C., Gray, W.A., Giddy, J.P., Fiddian, N.J.: Linking heterogeneous biodiversity information systems on the GRID: the GRAB demonstrator. *Computing and Informatics* **21** (2002) 383–398
8. Jones, A., White, R., Pittas, N., Gray, W., Sutton, T., Xu, X., Bromley, O., Caithness, N., Bisby, F., Fiddian, N., Scoble, M., Culham, A., Williams, P.: BiodiversityWorld: An architecture for an extensible virtual laboratory for analysing biodiversity patterns. In Cox, S.J., ed.: Proc. UK e-Science All Hands Meeting, Nottingham, UK, EPSRC (2003) 759–765
9. White, R., Bisby, F., Caithness, N., Sutton, T., Brewer, P., Williams, P., Culham, A., Scoble, M., Jones, A., Gray, W., Fiddian, N., Pittas, N., Xu, X., Bromley, O., Valdez, P.: The BiodiversityWorld environment as an extensible virtual laboratory for analysing biodiversity patterns. In Cox, S.J., ed.: Proc. UK e-Science All Hands Meeting, Nottingham, UK, EPSRC (2003) 341–344

Mega Process Genetic Algorithm Using Grid MP

Yoshiko Hanada¹, Tomoyuki Hiroyasu²,
Mitsunori Miki², and Yuko Okamoto³

¹ Graduate School, Department of Knowledge Engineering and
Computer Sciences, Doshisha University,
610-0321 Kyoto, Japan

`hanada@mikilab.doshisha.ac.jp`

² Department of Knowledge Engineering and Computer Sciences,
Doshisha University, 610-0321 Kyoto, Japan

`{tomo@is, mmiki@mail}.doshisha.ac.jp`

³ Department of Theoretical Studies Institute for Molecular Science,
Okazaki, Aichi 444-8585, Japan

`okamotoy@ims.ac.jp`

Abstract. In this study, a new Genetic Algorithm (GA) using the Tabu · Local Search mechanism is proposed. The GA described in this paper is considered a Mega Process GA, which has an effective mechanism to use massive processors, i.e., Mega Processors, in large-scale computing systems. Our proposed method has a GA-specific database that possesses information of searched space and performs a local search for the space that is not searched. Such mechanisms enable us to comprehend the quantitative rate of a searched region during the search. Using this information, the searched space can be expanded linearly as the number of computing resources increases and the exhaustive search is guaranteed under infinite computations. The proposed GA was applied to numerical test functions and the energy minimization problems of protein tertiary structures. The latter problem was performed under a heterogeneous distributed computing environment, which was built up with Grid MP produced by United Devices Inc.

1 Introduction

Genetic Algorithms (GAs) are among the most effective approximation algorithms for optimization problems[1]. Various mechanisms for improving GAs have been discussed. Minimal Generation Gap (MGG)[2] was proposed as a generation alternation model. Methods using Linkage Identification[3, 4], Real-coded GA[5], Probabilistic Model-Building GAs[6, 7], and Distributed GA[8] are other GAs that have strong search capabilities. The restart mechanisms have also been applied to enhance the performance of GAs[9, 10, 11, 12]. However, application of a GA to solve optimization problems has the drawback that GAs incur large computing costs. One solution to this problem is to perform GAs in parallel. In recent decades,

due to the remarkable improvements in computing capabilities, some parts of the drawback regarding computing costs of GAs are not really important. Furthermore, parallel processing is used to yield increases in performance of GAs. Recently, because of the emergence of super PC clusters and Grid computation environments, such as PC Grid comprised of desktop machines for home use or offices, the number of computational calculation resources is increasing. Thus, large computing projects in fields relating to evolutionary computing have become feasible. GAs are well suited to parallel processing environments due to the ability to search with multiple points, and consequently GAs have found application in large-scale computing[13, 14, 15, 16]. However, adapted methods are mostly simple parallelization of conventional GAs, which have been proposed for limited computing resources and an effective mechanism to make use of huge computing resources have yet to be designed. The easiest way that is commonly used for conventional GAs to use many resources is to increase the population size. However, enlarging the population also increases the diversity of the solutions, i.e., the convergence speed becomes slow. Subsequently, when GAs use a large amount of computer resources, the optimum solution is not derived quickly. At the same time, for conventional GAs, there is no guarantee of an exhaustive search in all search space although infinite computations are performed. As a result, although simple parallelization is applied to these methods, there is no assurance of improvement of their performance in accordance with the increase in available computing resources.

In this study, a new GA using the Tabu · Local Search mechanism for large-scale computer systems is proposed. We call such a GA using huge computing resources a Mega Process GA and our approach is to develop an effective mechanism to use massive processors, i.e., Mega Processors, in large-scale computing systems, such as super PC clusters and Grid computation environments. The proposed method uses a database that possesses information of space that has been searched already. At the same time, the proposed GA performs the local search for the space that is not searched to expand the searched space. These mechanisms enable us to comprehend the quantitative rate of the searched region during the search. Moreover, using this information, the searched space increases linearly as the number of computing resources increases and an exhaustive search is guaranteed under infinite computations. In addition, we examined the performance of the proposed method in a distributed computing environment, which is built up using the commercially available PC Grid middleware Grid MP that is currently used to develop several large-scale distributed computing projects produced by United Devices Inc¹.

2 Tabu · Local Search Mechanism for Mega Process GA

2.1 Database Structure

In this study, we introduce a GA-specific database that possesses information of the region that has already been searched. We used binary-coded individuals. In

¹ United Devices : <http://www.ud.com>

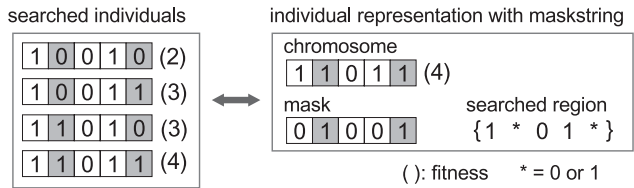


Fig. 1. An Example of an Individual stored in a Database

	Chromosome	Mask	Searched Region	Fitness	Hamming-Index
Individual x_1	1 0 1 1 1	1 0 0 0 1	* 0 1 1 *	4	2
Individual x_2	1 0 0 1 1	0 0 0 1 1	1 0 0 * *	3	2
Individual x_3	0 1 1 1 1	0 1 1 0 1	0 * * 1 *	4	3
Individual x_4	1 1 0 0 1	0 0 0 0 0	1 1 0 0 1	3	0

* = 0 or 1

Fig. 2. The Aspect of the Database of the proposed method

addition to chromosomes, individuals stored in the database possess bitstrings, which we call maskstrings. Maskstrings are also bitstrings, lengths of which are the same as those of chromosomes. A locus of a chromosome, which stands at '1' in a maskstring, represents that it has already searched all assignable genes to it. Fig. 1 shows that a set of searched individuals is compressed to one individual using a maskstring.

Our proposed database stores these individuals that hold maskstrings. Fig. 2 shows an instance of the database. The individual x_3 , stored in the database shown in Fig. 2 implies that "0 1 1 1 1" is the best solution under searching the set of individuals $X_3 = \{0 * * 1 *\} (* = 0 \text{ or } 1)$. We call the number of '1' in a maskstring the hamming-index, e.g. the hamming-index of the individual x_3 is 3.

When a database stores all the individual information, it takes a large time to check an individual that has been already searched due to its vast amounts of data. Our proposed notation of searched individuals is a highly compressed method using maskstrings and large searched regions are represented by several individuals. Moreover, checking individuals stored on the database is not time-consuming.

This notation of individuals enables us to provide a quantitative rate of a searched region during a search. In an individual x we denote its chromosome length by L , a gene of the locus l of a chromosome by c_{xl} , and a value of the locus l of a maskstring by m_{xl} . Quantitative sizes of searched regions are obtained as follows.

Case of one Individual. $|X|$ denotes a number of elements involved in a set X . The size of the searched region is indicated by an individual of which the hamming-index standing at h is 2^h , e.g., $|X_3|$, which is the size of the searched region indicated by the individual x_3 stored in the database shown in Fig. 2, as 2^3 .

Case of N Individuals. Given N individuals $x_i (1 \leq i \leq N)$ and their search regions X_i , the total searched region indicated by them is the union of sets $|X_1 \cup X_2 \cup \dots \cup X_N|$, i.e., $|\bigcup_{i \in I} X_i|$, where the set which consists of the suffix i is denoted $I = \{1, 2, \dots, N\}$. In most cases, it cannot be derived readily as a function of the number of elements in a union of sets. On the other hand, that of an intersection of sets is a closed form. The size of the searched region is such a case.

As a result, $|\bigcup_{i \in I} X_i|$ is derived from the sets of $|\bigcap_{j \in J} X_j|$, where J is a subset of I , which is shown in equation (1).²

$$|\bigcup_{i \in I} X_i| = \sum_{J \subseteq I, J \neq \phi} (-1)^{|J|-1} |\bigcap_{j \in J} X_j| \quad (1)$$

The distance between individuals x_i and x_j including their maskstrings, which is represented as $d(x_i, x_j)$, is defined in (2).

$$d(x_i, x_j) = \sum_{l=1}^L B |c_{x_i l} - c_{x_j l}| \quad (2)$$

$$B = \begin{cases} 1, & \text{if } m_{x_i l} = m_{x_j l} = 0 \\ 0, & \text{otherwise} \end{cases}$$

$|\bigcap_{i \in I} X_i|$ is a closed form and derived below as (3), where $[R]=z$ given real number R and integer z .

$$|X_1 \cap X_2 \cap \dots \cap X_N| = \begin{cases} 2^M, & \text{if } K = 0 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$$K = \sum_{i=1}^{N-1} \sum_{j=(i+1)}^N d(x_i, x_j)$$

$$M = \sum_{l=1}^L \left[\frac{1}{N} \sum_{i=1}^N m_{x_i l} \right]$$

2.2 Concept of the Proposed Method

The proposed method consists of a GA and a local search. The flow of our proposed method is shown in Fig. 3. To obtain optima earlier, our method of searches

² For instance, the number of elements of the union of three sets, X_1 , X_2 , and X_3 , is obtained as follows:

$$|X_1 \cup X_2 \cup X_3| = |X_1| + |X_2| + |X_3| - |X_1 \cap X_2| - |X_2 \cap X_3| - |X_3 \cap X_1| + |X_1 \cap X_2 \cap X_3|$$

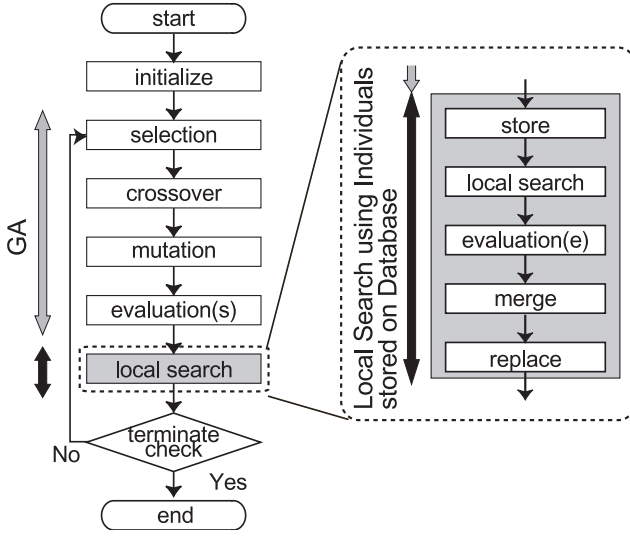


Fig. 3. The Flow of the Proposed Method

used mainly schemes of GA. Any method of operators, such as a crossover and mutation, or a generation alternation model can be applied. To use idle computing resources of enormous computing environments effectively, a local search is applied. Our proposed method is outlined as follows.

Step 1. Generate N_{pop} individuals randomly, where N_{pop} is the population size. In addition, no individuals are stored in the database.

Step 2. Apply operators, such as crossovers, mutations, and selections to individuals in a GA population.

Step 3. Store the individual y_{best} , which is the best individual of a GA population, in the database and set its maskstring to the bitstring of which the values of all loci are '0'. However, y_{best} is not stored when it is included in the searched regions, which are indicated by individuals that have already been stored in the database.

Step 4. /Local search/ Expand a searched region indicated by a certain individual stored in the database. When a better individual is found, use it to replace the worst individual of the population of GA. The details of the local search are described in the next section 2.3.

Step 5. Go back to step 2 until some termination conditions, e.g., computing cost reaches a limited amount or the exhaustive search is done, are satisfied.

At step 3, when there are N_{DB} individuals in the database, replace the individual x_{worst} that has the poorest fitness in the database with y_{best} if the fitness

of y_{best} is bigger than that of x_{worst} , otherwise y_{best} is not stored, where N_{DB} is the parameter of database capacity.

2.3 Local Search

We chose a certain individual from the database to apply the local search in every generation. This individual satisfies the condition that the hamming-index is the minimum value and its fitness is the maximum value. In our proposed local search, one of the loci, the values of which stand at '0' of the maskstring of the selected individual, is changed to a value of '1'. The locus that holds the largest variance of genes among all loci is selected. This operation is performed because it is suitable to keep essentially the same hamming-index among individuals stored in the database for the process of merging individuals, which is described later. Moreover, it is desirable to retain genes of loci, which have lower variances as they can be considered part of a better solution.

Given N individuals $x_i (1 \leq i \leq N)$ stored in the database, the process of the local search is outlined as follows. Fig. 4 shows an example of the local search.

Step 1. Find the results of a_l using equation (4), which is the absolute value of the difference between the average value of genes and 0.5 at each locus.

$$a_l = \left| \frac{1}{N} \sum_{i=1}^N (c_{x_i l}) - 0.5 \right| \quad (1 \leq l \leq L) \quad (4)$$

Step 2. Select the individual x , which has the minimum hamming-index and maximum fitness, from N individuals stored in the database.

Step 3. Select the locus l^* , which indicates $m_{xl^*} = 0$. At the same time, a_{l^*} is the minimum value, from $a_l (1 \leq l \leq L)$.

Step 4. Prepare the individual x' , which has the same maskstring as that of x . In its chromosome, each gene is exactly the same as that of x at loci standing at '1' except l^* in its maskstring. x' is the best individual under search X' , which should be searched to expand the search region.

Step 5. Update m_{xl^*} to '1' and h_x to $h_x + 1$. Furthermore, let x be x' and replace the worst individual of the population of GA by x if $x < x'$. The exhaustive search is finished when h_x approaches L .

At step 4, described above, X' is comprised of individuals produced by flipping the gene at l^* in the chromosome of each individual included in X . This requires searching of 2^{h_x} individuals where the hamming-index of x is h_x . Nevertheless, parallelization can be applied to easily search X' . Moreover, this mechanism uses computing resources effectively as the searched space increases linearly with increasing computing resources and an exhaustive search is guaranteed under infinite computations.

	Chromosome	Mask	Searched Region	Fitness	Hamming-Index																																				
Individual x_1	<table><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	1	1	1	1	<table><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table>	1	0	1	0	1	0	<table><tr><td>*</td><td>0</td><td>*</td><td>1</td><td>*</td><td>1</td></tr></table>	*	0	*	1	*	1	5	3																		
1	0	1	1	1	1																																				
1	0	1	0	1	0																																				
*	0	*	1	*	1																																				
Individual x_2	<table><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr></table> <div><div>↶</div><div>update</div><table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table></div>	1	1	0	1	1	1	1	1	1	1	1	1	<table><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table> <div><div>↶</div><div>update</div><table><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table></div>	1	0	0	0	1	0	1	0	1	0	1	0	<table><tr><td>*</td><td>1</td><td>0</td><td>1</td><td>*</td><td>1</td></tr></table> <div><div>↶</div><div>update</div><table><tr><td>*</td><td>1</td><td>*</td><td>1</td><td>*</td><td>1</td></tr></table></div>	*	1	0	1	*	1	*	1	*	1	*	1	5 <div><div>↶</div><div>update</div>6</div>	2 <div><div>↶</div><div>update</div>3</div>
1	1	0	1	1	1																																				
1	1	1	1	1	1																																				
1	0	0	0	1	0																																				
1	0	1	0	1	0																																				
*	1	0	1	*	1																																				
*	1	*	1	*	1																																				
Individual x_3	<table><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	0	1	1	0	<table><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	0	0	1	1	0	<table><tr><td>0</td><td>1</td><td>0</td><td>*</td><td>*</td><td>0</td></tr></table>	0	1	0	*	*	0	3	2																		
0	1	0	1	1	0																																				
0	0	0	1	1	0																																				
0	1	0	*	*	0																																				
Individual x_4	<table><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	1	1	0	<table><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	0	0	1	1	1	0	<table><tr><td>0</td><td>1</td><td>*</td><td>*</td><td>*</td><td>0</td></tr></table>	0	1	*	*	*	0	4	3																		
0	1	1	1	1	0																																				
0	0	1	1	1	0																																				
0	1	*	*	*	0																																				

*

 = 0 or 1

Fig. 4. An Example of Local Search in the one max problem: The individual of which the search region is expanded is individual x_2 . Expanding the region $X_2 = \{ * 1 \underline{0} 1 * 1 \}$ to the region $\{ * 1 * 1 * 1 \}$ requires searching the region $X'_2 = \{ * 1 \underline{1} 1 * 1 \}$. The searched region of x_2 , i.e., X_2 , becomes $\{ * 1 * 1 * 1 \}$, after applying this expansion

2.4 Merge Operation in Database

Following the local search, a merge of individuals stored in the database is executed in every generation to avoid overlapping searches in the process of the local search. Individuals can be merged when the following conditions are satisfied:

Condition 1. x_a and x_b are given individuals. When they satisfy the following conditions, they are in condition 1. They have the same maskstrings, $d(x_a, x_b) = 1$, and locus is l^* , which satisfies $m_{x_a l^*} = m_{x_b l^*} = 0$ and $c_{x_a l^*} \neq c_{x_b l^*}$. In this case, select one that has better fitness from x_a and x_b , and let its value of l^* in its maskstring be '1'. At the same time, the other is deleted from the database. For example, in Fig. 4, the individual x_1 and the updated individual x_2 can be merged with the condition 1 then let $m_{x_2 2}$ be '1' and h_{x_2} be '4'. x_1 is then deleted.

Condition 2. x_a and x_b are given individuals. When these individuals satisfy the following conditions, they are in condition 2: $d(x_a, x_b) = 0$ and no locus exists that satisfies $m_{x_a l} = 1, m_{x_b l} = 0 (1 \leq l \leq L)$. In this case, x_a is deleted from the database because of $X_a \subset X_b$. For example, in Fig. 4, the individual x_3 and the individual x_4 can be merged meeting condition 2. Therefore, x_3 is deleted.

Condition 3. x_a and x_b are given individuals. When $X_a \cap X_b \neq \phi$ and $|X_a| \geq |X_b|$, they are in condition 3 shown Fig. 5. In this case, X_a is expanded until it can include X_b , and then they are merged using condition 2. X'_a indicates the region that is required to expand until it can include X_b . $X'_a \cap \neg X_b$ must be searched to merge using condition 2. We introduce parameter A , which indicates the ratio of $|X_a \cap X_b|$ of $|X'_a \cap \neg X_b|$, i.e., $|X_a \cap X_b| / |X'_a \cap \neg X_b|$, because we obtain a better solution under the available limited computing resources. This merge is

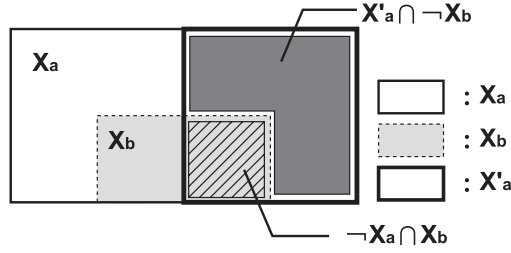


Fig. 5. Merge using Condition 3

not applied when A is larger than a certain value. To decide an appropriate A , the size of the search space and the number of available computing resources or cost must be considered. In the present study, we set A to 8.

3 Implementation of Mega Process GA on Distributed Computing Environments

3.1 Distributed Processing at Local Search

Parallelization is applicable to GA by using a master-slave model at evaluations or crossovers. However, when the population becomes larger, the diversity of the solutions also increases. As a consequence of this, huge computing resources cannot be used effectively. Our proposed local search is applied to use massive processors, i.e., Mega Processors.

At the local search phase in our proposed method, application of a local search to individual x corresponds to searching the individual x' of which the maskstring is the same as that of x . It also satisfies $d(x, x') = 1$, i.e., the set of individuals that have arbitrary values at loci of the chromosome standing at '1' in the maskstring of x' and the same values as those of x at other loci should be searched. In the example shown in Fig. 4, for application of the local search to $X_2 = \{ * 1 0 1 * 1 \}$ at locus 3 it is necessary to search $X'_2 = \{ * 1 1 1 * 1 \}$ ($*$ = 0 or 1).

By using the opposite operation against condition 1 of the merger, parallelization can be applied easily to evaluation of a set of individuals represented with a maskstring. Fig. 6 shows an example in which a set of individuals is split to some segmental sets of individuals.

To execute the local search in distributed computing environments, these parts of the set are allotted to computation nodes. Each node evaluates assigned individuals independently, which requires no communication among nodes.

3.2 Implementation on Grid MP

We examine the performance of the proposed method in a distributed computing environment, built using the commercially available middleware Grid MP from

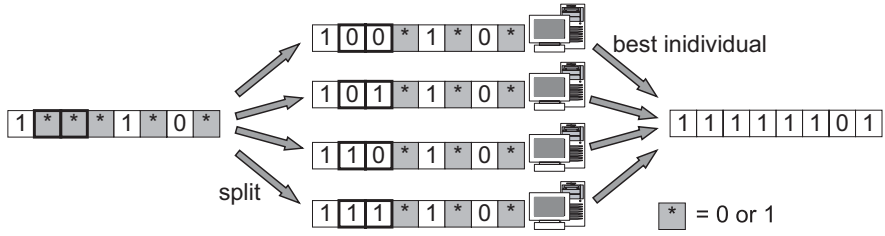


Fig. 6. Example of Splitting Individuals

United Devices Inc.[17]. Grid MP is one of the toolkits of Grid computing and is currently used to power several large-scale distributed computing projects. In the Grid MP platform, the underutilized resources of many computers are aggregated and used as a virtual computer system. The distributed computing environment constructed by Grid MP consists of an MP Server and Devices. The MP Server is a server that carries out user or Device authentication, monitoring and scheduling of jobs, dispatching jobs to Devices, etc. Devices are computation nodes that execute the jobs assigned by the MP Server.

In the Grid MP platform, application developers prepare their Program Module executables, which are components of applications and consist of precompiled executables and MDFs (Module Definition Files), and upload them to the File Service of the MP Server. Data Packages, which are sets of reference data during executions and PMFs (Package Manifest Files), have to also be registered as a Data Set. MDFs indicate names of the executable, arguments, and the file in which the results are written. PMFs show the compression format and the file name of the data. A Job object comprising of several Workunits is created once a user submits a Job. A Workunit is the minimum unit of a Job that one Device has to execute and defines the Program Module executable and the Data to be used. A Device during the polling state is assigned one Workunit. Fig. 7 shows the standard Job execution form of Grid MP.

To implement our proposed local search on distributed computing environments as shown in Fig. 6, the Program Module, which reads a data file in which individuals to be searched are written and searches these individuals, must be prepared.

3.3 Overhead of Grid MP

We performed our proposed method in the heterogeneous distributed computing environment composed of machines belonging to RIKEN Genomic Sciences Center (GSC)³ and Intelligent Design Systems Laboratory (ISDL) of Doshisha University⁴. We used Grid MP platform version 4.0-3106. The specification of machines used for the experiments is shown in Table 1. The User Machine in Table 1 indicates the machine of a user who submits a job to the MP Server.

³ RIKEN GSC : <http://big.gsc.riken.jp/>

⁴ ISDL of Doshisha Univ. : <http://mikilab.doshisha.ac.jp/>

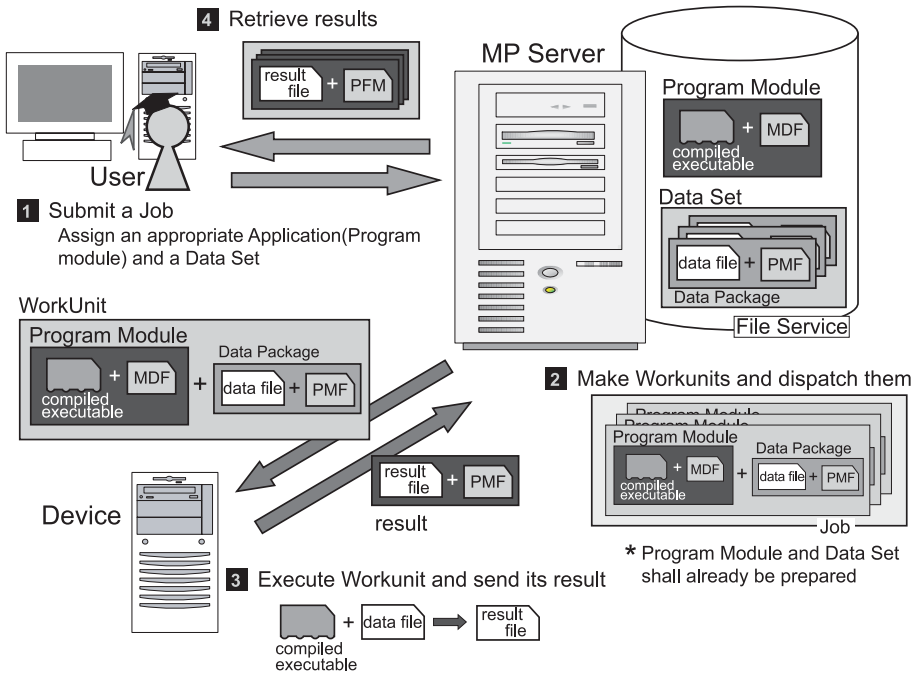


Fig. 7. Aspect of Performance of Job Execution form of Grid MP

Table 1. Specification of Machines Used for the Experiment

	Affiliation	Machine name	#Nodes	Processor	Memory
User Machine	- -	harrier	1	mobile Pentium 1.8GHz	1GB
MP Server		naiad	1	Xeon 2.8GHz x 2	2GB
Devices	Doshisha	forte01-15	15	PentiumIII 600MHz	128MB
	ISDL	libra/tiger	2	Xeon 2.8GHz x 2	1GB
	RIKEN GSC	le01-23	23	Celeron 1.3GHz	896MB

There is overhead due to the distinctive characteristics of the distributed computing environment, such as communication environment and latency of the middleware architecture dispatch mechanism. The latency is caused by some processes on the MP Server and the Devices, e.g., creation of a Job object and a schedule of Workunits and downloads of Workunits from the File Service of the MP Server.

We examined elapsed times of execution of the Job, the Workunit of which was only assigned and required no calculation in Devices. The number of Workunits included in the Job was set to 32, 64, 128, and 256. Polling interval was set to 30 seconds and 1 minute. 30 seconds is the minimum feasible interval. Fig. 8 shows the average value of Job creation time and its execution time. The results shown are from 10 trials.

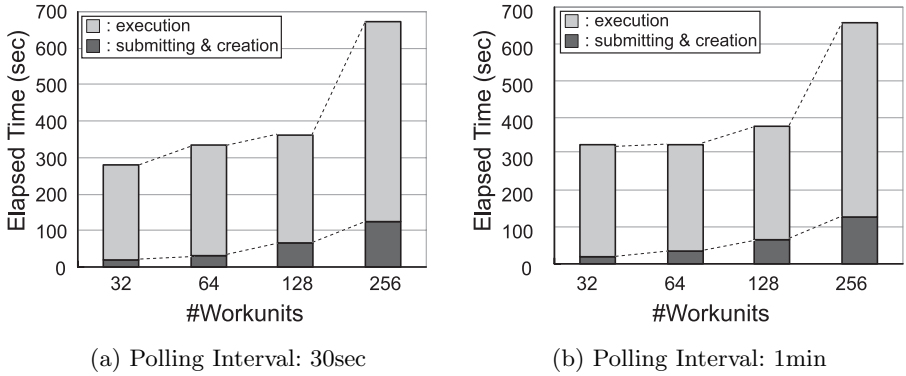


Fig. 8. Elapsed Times of Empty Job

A Workunit is allotted to a Device, which has sent idle state of CPU to the MP Server several times. As a consequence of this, execution time is expected to depend on polling interval. However Fig. 8 illustrates that this interval has no effect much on elapsed times. It indicates that the execution time is taken up by preliminary processes such as preparing Workunits for their assignments rather than by Devices queuing.

From Fig. 8 the period time of Job creation increases exponentially as increasing in Workunits, in contrast execution time will not increase. This is because once the Device completes its assigned Workunit and sends result to the MP Server, it obtains a Workunit again if incomplete Workunits stay still in the File Service. Therefore the latency of dispatching is hidden seemingly.

4 Numerical Experiments

To discuss the effectiveness of our proposed method in both infinite and limited computation, it was applied to the one max problem and 3-deceptive problem[18]. The former is the most primitive benchmark problem of bitstrings, and its fitness is a summation of the number of '1' included in a chromosome. The latter is one of trap functions described as equation (5):

$$F_{3\text{-deceptive}} = \sum_{i=1}^N f_i \quad (5)$$

$$f_i = \begin{cases} 0.9, & u_i = 0 \\ 0.8, & u_i = 1 \\ 0.7, & u_i = 2 \\ 1.0, & u_i = 3 \end{cases}$$

The effectiveness of our method is discussed by solving these problems with limited computation. The following equations (6), (7), (8), and (9) are the continuous test functions:

$$F_{Rastrigin} = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (6)$$

$$x_i \in [-5.12, 5.12]$$

$$F_{Schwefel} = \sum_{i=1}^n -x_i \sin(\sqrt{|x_i|}) \quad (7)$$

$$x_i \in [-5.12, 5.12]$$

$$F_{Ridge} = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2 \quad (8)$$

$$x_i \in [-64, 64]$$

$$F_{Griewank} = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \left(\cos\left(\frac{x_i}{\sqrt{i}}\right) \right) \quad (9)$$

$$x_i \in [-512, 512]$$

In addition, we applied our proposed method to prediction of protein tertiary structures. Proteins in nature have structures with the lowest potential energy. Therefore, their structures can be predicted by energy minimization. To treat prediction protein tertiary structures as optimization problems, energy functions that define the structures of proteins are used as objective functions, a design variable of which is the dihedral angle among the atoms that make up the proteins[19]. We compared our method to conventional GA for this problem and examined the performance of the proposed method in the distributed computing environment built using Grid MP.

4.1 Performance of the Proposed Method in Infinite Computation Cost

We applied the proposed method to the one max problem and 3-deceptive problem with a string length of $L=30$ without limiting computing resources. This search was terminated when all the combinations had been searched. An exhaustive search needs 2^{30} solutions. The ER model[20] was used as the alternation in each generation. We applied a GA with uniform crossover and each couple, or parents, generated 20 children by crossover. The mutation rate was $0.03(=1/L)$ and population size was 20. The capacity of the database, N_{DB} , was 5 in this experiment.

Fig. 9 shows the transition of the fitness and the ratio of the searched region on solving the one max problem. In Fig. 9(b), when the ratio of the searched region attained 1.0, all the area had been searched.

The performance of the proposed method was similar to that of the conventional GA. This was because the proposed method searched using mainly

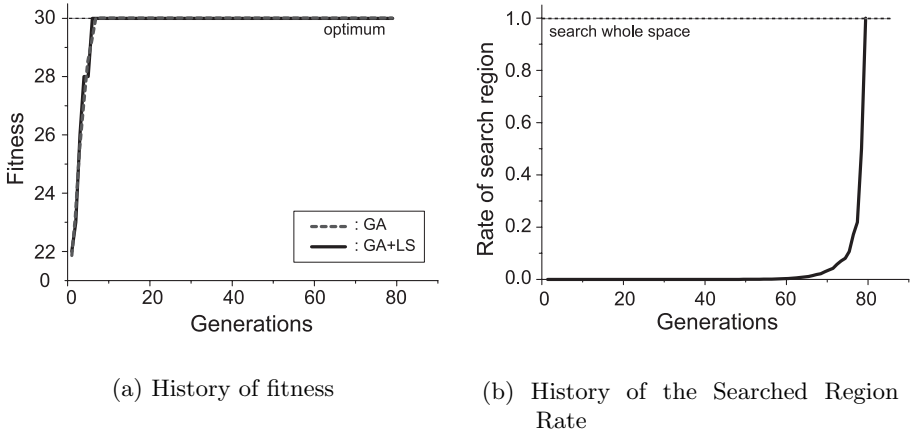


Fig. 9. Performance of GA using the Local Search Mechanism in the One Max Problem

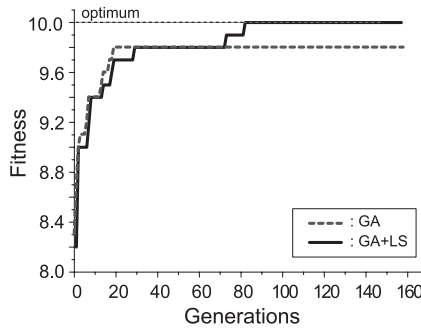


Fig. 10. Performance of GA using the Local Search Mechanism in the 3-deceptive Problem

schemes of GA. It is certain that the solution obtained by an exhaustive search is the optimum, although the optimal solution is obtained in the earliest part of the search. Moreover, these mechanisms enable us to show the quantitative ratio of the searched region during the search as in Fig. 9(b).

Fig. 10 shows the transition of the fitness in the 3-deceptive problem. Both the conventional GA and our proposed method fell into local optima in the early stages of the search. In the 3-deceptive problem, it is difficult for GAs to obtain the optimal solution because populations tend to be trapped by local optima. Similar to a conventional GA, our proposed method obtains convergence. Nevertheless, it can obtain the optimum as increases in computing costs yield increases in the searched regions. As a consequence of this, the optimal solution can be found by continuing the search.

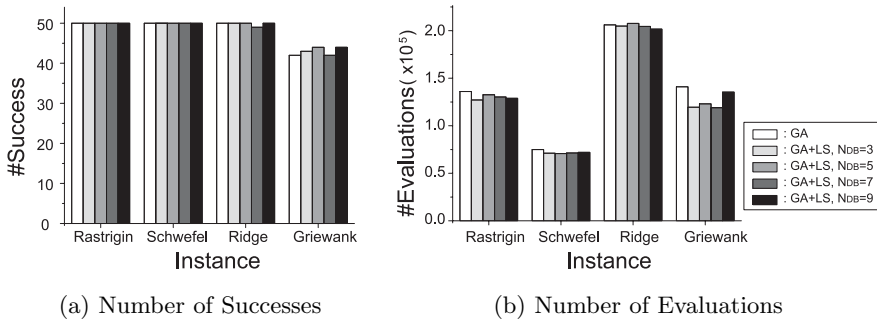


Fig. 11. Performance of GA using the Local Search Mechanism with Continuous Functions

4.2 Performance of the Proposed Method with Limited Computation Cost

To perform the exhaustive search, many evaluations of individuals are required. Our proposed method can perform an exhaustive search. It is expected that the proposed method has a high possibility of finding the optimum solution in the early stages of searches because it mainly uses schemes of GA. Therefore, we then examined whether our method can obtain the optimum under limiting evaluations.

We applied the proposed method to four test continuous functions to compare it with conventional GA. In each function, an optimum solution was attempted by the proposed method. Each function had 10 dimensions and the number of evaluations was limited to 2.5×10^5 . The ER model was used as the alternation in each generation. We applied a GA with uniform crossover and each couple, or parents, generated 20 children by crossover. We set the mutation rate to $0.01 (=1/L)$ and the population size to 200. The capacity of the database, N_{DB} , was set to 3, 5, 7, and 9 in this experiment.

Fig. 11 describes the number of trials that obtained the optimum and the average number of evaluations needed to acquire the optimum. The results shown are from 50 trials.

Fig. 11(a) illustrates that the proposed method and the conventional GA obtained the optimal solution in all trials at Rastrigin function, Schwefel function, and Ridge function. The proposed method derived the optimal solutions several times at Griewank function. In addition, Fig. 11(b) indicates that our method could obtain the optimum with fewer evaluations than a conventional GA although our proposed method required many evaluations at the local search phase. These results indicated that our method also retained superior performance with limited computing costs. We focused attention on the effects of parameter N_{DB} on the performance of our method. The numbers of successful trials of $N_{DB} = 3$ and 9 were greater than those of $N_{DB} = 5$ and 7, whereas the result of the number of evaluations was contrary in the Ridge function. In contrast, the number of successful trials of $N_{DB} = 9$ was better than those of

$N_{DB} = 3, 5$ and 7 . As a result, there is no setting that can acquire a more optimal solution with fewer evaluations but the proposed method can search free from the setting of parameter N_{DB} .

4.3 Performance of the Proposed Method in Protein Tertiary Structure Energy Minimization Problems

To discuss the performance of our method in real complex problems, we applied the method to protein tertiary structure energy minimization problems. In this experiment, we used OPLS-AA/L[21, 22], which is one of the potential energy functions within the framework of classical mechanics consisting of certain energy terms with force-field parameters, and examined our method on a small protein named Met-enkephalin composed of 5 amino residues and 23 dihedral angles.

Applying this minimization problem to GA, the range of value of each dihedral angle was $[-\pi, \pi)$. An angle is expressed strings of 6 bits, i.e., it is divided into 2^6 equal intervals. dMSXF[23] was used as the crossover and the alternation in each generation. In all crossovers, the number of transitions, k_{max} , was set to 10 and the number of generating neighbor individuals of the respective step, μ , was 10. Therefore, each couple, or parents, generated 100 children. We set the population size to 40, 60, and 80. The capacity of the database, N_{DB} , was set to 5 in this experiment. The number of evaluations was limited to 1.2×10^5 .

Table 2 shows the best, the average, the median and the worst value of obtained energies. Our method retains the performance of a conventional GA in

Table 2. The Performance of Proposed Method on Met-enkephalin

Population Size	GA+LS				GA			
	best	med*	avg**	worst	best	med	avg	worst
40	-282.6	-281.3	-281.4	-281.1	-283.0	-281.3	-281.5	-281.2
60	-282.2	-281.3	-281.4	-281.2	-282.0	-281.3	-281.4	-281.2
80	-283.2	-281.2	-281.2	-280.9	-282.1	-281.2	-281.2	-281.0

*,median, **:average

a complex problem, i.q., benchmark problems, such as the one max problem. The latest reported minimum energy of this protein is approximately -287.5 obtained by the method PSA/GAc[24]. This examination has not matured yet and needs sophistication of the parameters to obtain better solutions.

We performed this experiment in a heterogeneous distributed computing environment shown in Table 1. To implement our method in this environment, operations of the GA are executed on the User Machine. At the local search phase, expanding the searched region is executed in parallel using 40 Devices, forte01-15, libra, tiger, and le01-23. Fig. 12 illustrates the environment used for the experiments.

From Fig. 8, at least approximately 400 seconds exist as overhead. The local search is then executed on distributed environment when the size of the individ-

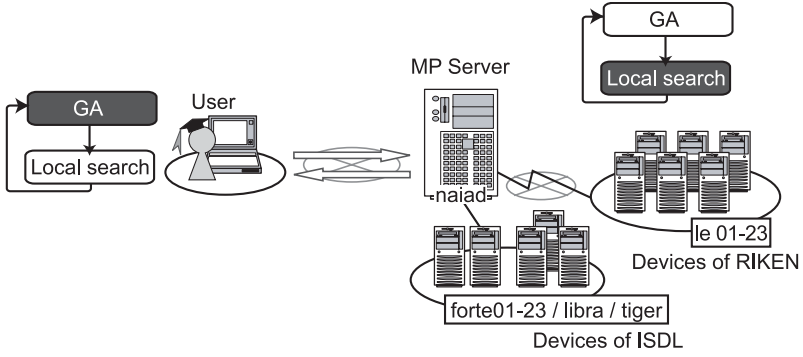


Fig. 12. Computing Environment for Computational Experiments

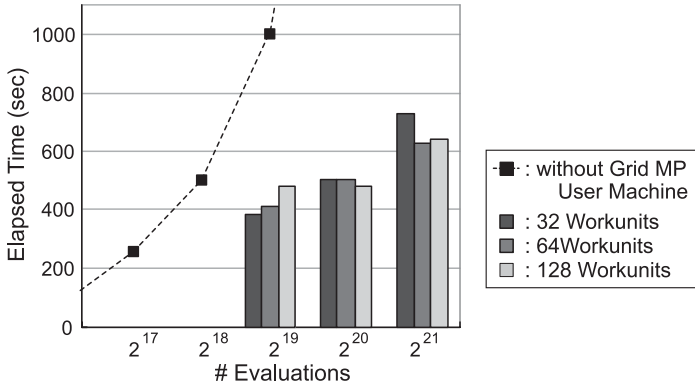


Fig. 13. Elapsed Times of Local Search on energy minimization of Met-enkephalin

uals that should be searched surpasses a certain value. In the energy minimization problem of Met-enkephalin, it is clear at pilot study that evaluations of 2^{18} individuals take approximately 500 seconds using the User Machine. Thereby evaluations in excess of 2^{19} should be executed in this distributed environment.

We examined the elapsed time of the Job, which evaluated a set of individuals represented with a maskstring. In this experiment, 2^{19} , 2^{20} , and 2^{21} evaluations were split to 32, 64, and 128 Workunits. In each setting of number of Workunits, the total of evaluations was divided equally to Workunits, e.g., the number of evaluations that each Workunit had was 2^{14} where the number of Workunits was set to 64 and total of evaluations was 2^{20} .

Fig. 13 shows the total execution times of energy minimization problem of Met-enkephalin using Grid MP. The results are the averages of 10 trials.

It illustrates that our proposed method can execute faster on Grid MP than only using the User Machine. In addition the execution time does not depend on setting of Workunits essentially. Focusing at the result of 32 Workunits, execution time is slightly much than 64 and 128 Workunits with 2^{19} calculations. Grid MP

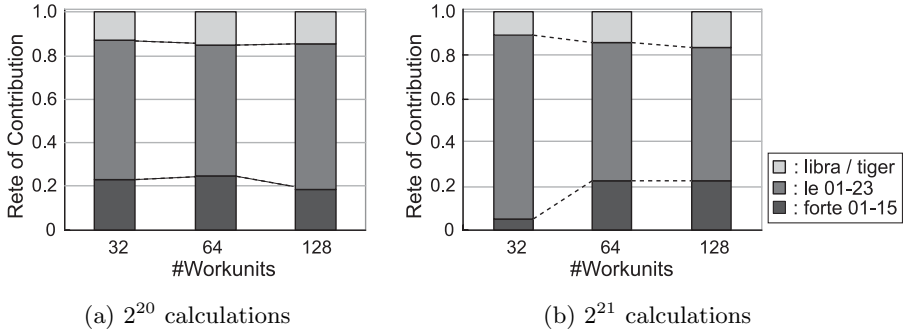


Fig. 14. Contribution of Devices

has the redundancy for its high performance and fault-tolerance. Some Devices execute the same Workunit at a time and the fastest finished result is adopted when total of Workunits is less than number of Devices. As a result, the waste of calculations arises if no fault exists. In addition high spec machines will have finished calculating and sending their results before low spec machines even if low spec machines obtain Workunits earlier than high spec machines. Thus low spec machines, which are assigned Workunits that high spec machines also execute, cannot contribute their computation resources in the environment which has excessive Devices against Workunits unless high spec machines lose connections to the MP Server by some problems during executing Workunits.

Fig. 14 shows that the rate of Devices that are adopted their results, i.e., the rate of contribution of Devices, with 2^{20} and 2^{21} calculations.

The contribution was much the same in all experiments except for the result 32 Workunits with 2^{21} calculations. Few trials exist that forte01-15 could contribute superficially in the experiment of 32 Workunits with 2^{21} calculations. In this implementation, one Workunit poses 2^{16} calculations to a Device where the number of Workunits is set to 32 and total of evaluations is 2^{21} . 2^{16} calculations are so heavy that forte01-15 cannot finish the executable even if delays of assignment of Workunits exist.

The number of Workunits has to be set bigger than the number of Devices. The overhead increases in response to increasing in Workunits. Thus appropriate number of Workunits should be set to yield high performance of proposed method in a heterogeneous distributed environment with considering contributions of Devices.

5 Conclusions and Future Work

GAs are suitable algorithms for parallel processing. However, increases in the number of individuals and/or computing resources do not yield improvements of performance in most methods because the diversity of the solution is also increased. Our proposed method, Tabu · Local Search mechanism for Mega Pro-

cess GA, can expand the searched region linearly as the available computing resources increases. Furthermore, the exhaustive search is guaranteed under infinite computations, while the exhaustive search is not guaranteed with conventional GAs. The proposed method was tested on the one max problem without limiting computing resources. The results confirmed that the solution obtained with this method is optimum by exhaustive search, although the optimal solution is obtained in the earliest part of the search. In addition, we applied the proposed method to four test continuous functions to derive the optimum solutions for comparison with conventional GA. These results indicated that the proposed method also retains superior performance with limited computing costs.

In addition we performed the proposed method in one of the instances of the energy minimization problems of protein tertiary structures in a heterogeneous distributed computing environment composed of 40 computation nodes belonging to RIKEN GSC and ISDL of Doshisha University, which was built up with Grid MP. We examined the execution time that included the overhead in this environment and discussed the appropriate setting of number of Workunits with considering overheads and contribution of computation nodes.

In future work, we will apply our proposed method to a large-scaled computing Grid and examine its effectiveness. Moreover, we will apply restarts in the non-searched region when the population of the GA obtains convergences as the proposed method can distinguish the non-searched region from the whole search space.

Acknowledgments

We are grateful to Prof. Dr. Akihiko Konagaya and Fumikazu Konishi of RIKEN Genomic Sciences Center and Hiroyuki Kobayashi of Sumisho Electronics Co., Ltd.⁵ for valuable discussion and contributions to the development of the distributed computing environment built using Grid MP.

References

1. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley (1989)
2. H. Satoh, M. Yamamura and S. Kobayashi: Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation. Proc. of IIZUKA. pp.494-497. 1996
3. H. Kargupta: SEARCH, polynomial complexity, and the fast messy genetic algorithm. University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Report No. 95008. 1995
4. G. R. Harik: Linkage learning in via probabilistic modeling in the ECGA. University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Technical Report No. 99010. 1999

⁵ Sumisho Electronics Co., Ltd. : <http://www.sse.co.jp/e/index.html>

5. I. Ono and S. Kobayashi: A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover. Proc. of 7th Int. Conf. on Genetic Algorithms. pp.246-253. 1997
6. Pelikan,M., Goldberg,D.E., and Lobo,F.: A Survey of Optimization by Building and Using Probabilistic Models. Technical Report 99018, IlliGAL (1999)
7. Larranaga,P., Lozano,J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers (2001)
8. Reiko Tanese: Distributed Genetic Algorithms. Proc. 3rd International Conference on Genetic Algorithms. pp.434-439. 1989
9. T. Jansen: On the Analysis of Dynamic Restart Strategies for Evolutionary Algorithms Proc. Parallel Problem Solving from Nature - PPSN VII, 7th International Conference. pp.33-43. 2002
10. Alex S. Fukunaga: Restart Scheduling for Genetic Algorithms, Lecture Notes in Computer Science, vol.1498, pp.357-369. 1998
11. Sean Luke: When Short Runs Beat Long Runs, Proceedings of the Genetic and Evolutionary Computation Conference, pp.74-80. 2001
12. J. Maresky et al.: Selectively Destructive Restart, Proc. of Sixth International Conference on Genetic Algorithms, pp.144-150. 1995
13. Yusuke Tanimura: Parallel and Distributed Genetic Algorithm on The Cluster System and The Computational Grid. University of Doshisha. 2003, in Japanese
14. Hiroaki Imade et al.: A Grid-Oriented Genetic Algorithm for Estimating Genetic Networks by S-Systems, Proc. SICE Annual Conf. pp3317-3322, 2003
15. Hiroaki Imade et al.: A framework of grid-oriented genetic algorithms for large-scale optimization in bioinformatics Proc. of The Congress on Evolutionary Computation in Canberra. vol.1, pp623- 630, 2003
16. H. Nakata et al.: Protein structure optimization using Genetic Algorithm on Jojo Journal of Information Processing Society of Japan. 2002-HPC-93, pp. 155-160, 2003. in Japanese
17. United Devices. Grid MP 4.0 Application Developer's Guide, 2003.
18. Martin Pelikan et al.: BOA:The Bayesian Optimization Algorithm. IlliGAL Report No. 99003 1999
19. Y. Sakae and Y. Okamoto. Optimization of protein force-field parameters with the Protein Data bank. <http://arxiv.org/abs/cond-mat/0309110>.
20. D. Thierens, D. E. Goldberg: Elitist Recombination: an integrated selection recombination GA Proceedings of the 1st IEEE Conference on Evolutionary Computation pp.508-512. 1994
21. W. L. Jorgensen, D. S. Maxwell and J. Tirado-Rives. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. J. Am. Chem. Soc., 117, 11225-11236. 1996
22. G. A. Kaminsky, R. A. Friesner, J. Tirado-Rives and W. L. Jorgensen. Evaluation and Reparametrization of the OPLS-AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides. J. Phys. Chem. B, 105, 6474-6487. 2001
23. K. Ikeda, S. Kobayashi: Deterministic Multi-step Crossover Fusion: A Handy Crossover for GAs. Proceedings of 7th International Conference on Parallel Problem Solving from Nature pp162-171. 2002
24. M. Ogura, T. Hiroyasu, M. Miki and Y. Okamoto. Implementation Models for Distributed Memory Architecture of Parallel Simulated Annealing using Genetic Crossover. Proceedings of Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. pp121-126. 2001

“Gridifying” an Evolutionary Algorithm for Inference of Genetic Networks Using the Improved GOGA Framework and Its Performance Evaluation on OBI Grid

Hiroaki Imade¹, Naoaki Mizuguchi¹, Isao Ono¹, Norihiko Ono¹,
and Masahiro Okamoto²

¹ The University of Tokushima, 2-1 Minamijosanjima, Tokushima, 770-8506, Japan
{hiro1121, nio, isao, ono}@is.tokushima-u.ac.jp

² Kyushu University, 6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
okahon@brs.kyushu-u.ac.jp

Abstract. This paper presents a genetic algorithm running on a grid computing environment for inference of genetic networks. In bioinformatics, inference of genetic networks is one of the most important problems, in which mutual interactions among genes are estimated by using gene-expression time-course data. Network-Structure-Search Evolutionary Algorithm (NSS-EA) is a promising inference method of genetic networks that employs S-system as a model of genetic network and a genetic algorithm (GA) as a search engine. In this paper, we propose an implementation of NSS-EA running on a multi-PC-cluster grid computing environment where multiple PC clusters are connected over the Internet. We “Gridify” NSS-EA by using a framework for the development of GAs running on a multi-PC-cluster grid environment, named Grid-Oriented Genetic Algorithm Framework (GOGA Framework). We examined whether the “Gridified” NSS-EA works correctly and evaluated its performance on Open Bioinformatics Grid (OBIGrid) in Japan.

1 Introduction

Inference of genetic networks is a problem in which mutual interactions among genes are estimated by using gene-expression time-course data observed in biological experiments. Recently, inference of genetic networks has attracted attention as one of the most important problems in bioinformatics. This is because large-scale accurate gene expression profiles have been available in virtue of the recent development of DNA microarray or DNA chip technologies.

S-system [23, 28] is known as one of the promising models of genetic networks. S-system has been expected to be able to approximate various interactions among genes because it is a simultaneous nonlinear differential equation system based on the power law formalism. Inference of genetic networks by S-system is formalized as a search problem in which appropriate system parameters of S-system must be found so that the difference between the time course data

calculated by the S-system model and that observed in biological experiments is less than a tolerance given by a user. This problem has the difficulty of high-dimensionality, multimodality and strong non-linearity. This problem also has the difficulty that there is no way to determine a network structure uniquely. This is because sufficient time-course data to determine a network structure uniquely cannot be obtained in experiments because of difficulty and high cost of experiments. For this reason, we have to find as many network structures that explain the experimentally-observed time-course data and are biologically-promising ones as possible.

The Genetic Algorithm (GA) is an optimization method inspired by the evolution process of living things [5]. Because the GA is not only a direct search method but also a population-based stochastic search method, the GA can efficiently search multiple good solutions on high-dimensional, non-linear and multimodal problems. Hence, the GA has been expected as a promising method for searching the system parameters of S-system [1, 2, 10, 12, 18, 16, 17, 21, 25, 26, 27]. Of the existing methods, Network-Structure-Search Evolutionary Algorithm (NSS-EA) [18] has shown good performance. In NSS-EA, the search for network structures and that for system parameters corresponding to each network structure are explicitly separated, which enables to efficiently find multiple different good network structures. This method reportedly succeeded to find multiple different good structures more efficiently than conventional methods on a five-substance benchmark problem [18].

For problems such as inference of genetic networks by S-system in which it takes a long time to calculate an evaluation value of a solution candidate, GAs running on parallel computers, called parallel GAs, are useful to find good solutions in a realistic time. There are many studies on parallel GAs for reducing search time by using multiple CPUs. They can be categorized into three approaches; the master-worker-type GA [15], the island-type GA [24, 6, 9, 8] and the cellular-type GA [7]. In these studies, they implemented their algorithms on SMP computers or PC clusters. However, in order to solve larger problems, we need more computation power than that provided by a single organization.

Recently, the grid computing environment has attracted attention as a new computing environment from the viewpoint of problem solving by virtual organizations. Some studies on grid computing have been made actively in order to use geographically-distributed resources securely and transparently over the Internet [4, 14]. There are some grids for studying bioinformatics applications such as North Carolina Bioinformatics Grid [19], BioGrid [3] and Open Bioinformatics Grid [13]. However, there are almost no studies on GAs that are supposed to run efficiently on a grid computing environment due to the following reasons: Generally, a grid differs from a PC cluster in that the users belong to multiple different organizations. All nodes cannot share a file system. Remote nodes cannot be directly accessed over the Internet due to NATs. The connections over the Internet are unstable and low-speed, and computation abilities of each node are not homogeneous. For these reasons, the existing implementations of parallel GAs developed for SMP computers or PC clusters can not

work on grid computing environments. In order to develop Grid-Oriented GAs (GOGAs), GA researchers must understand various domain knowledge on grid computing such as security, transferring files and invoking remote processes over the Internet, handling troubles with remote nodes and network, and how to hide the network latency, even if some middlewares such as Globus Tool Kit (GTK) [4] are adopted. In order to facilitate the development of GOGAs, we have developed the GOGA Framework [11] based on GTK and Java. By using the GOGA Framework, a GOGA developer can be liberated from writing complex source codes for utilizing grid middleware.

The rest of this paper consists of six sections as follows: In section two, we briefly introduce NSS-EA [18]. Then, we discuss the requirements of the grid implementation of NSS-EA in section three. Section four introduces GOGA Framework [11] including recent improvements. In section five, we discuss an implementation of NSS-EA running on a multi-PC-cluster grid environment, named the “Gridified” NSS-EA developed by using the GOGA Framework. We report the early evaluation of the proposed “Gridified” NSS-EA on the OBIGrid in section six. Section seven is conclusion and future work.

2 Network-Structure-Search Evolutionary Algorithm

2.1 Inference of Genetic Networks by S-System

S-system has attracted attention as one of the promising models of genetic networks. S-system is given by

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^n X_j^{g_{ij}} - \beta_i \prod_{j=1}^n X_j^{h_{ij}} \quad (1)$$

where X_i (> 0) is the concentration of a substance i or an expression level, α_i , β_i , g_{ij} and h_{ij} are the system parameters that determine the shape of time course. g_{ij} is the interaction coefficients of the effect of X_j on the synthesis of X_i . h_{ij} is the interaction coefficients of the effect of X_j on the degradation of X_i . α_i and β_i are the rate constants of synthesis and degradation, respectively, and are non-negative. As shown in Eq. 1, S-system is a full connection model which assumes that all the state variables, X_j , affects the synthesis process of X_i , represented by the first term in the right part of Eq. 1, and the degradation process of X_i , represented by the second term in the right part of Eq. 1. The signs of g_{ij} and h_{ij} determine a network structure. If g_{ij} (h_{ij}) is positive, substance j induces the synthesis (degradation) of substance i . If g_{ij} (h_{ij}) is negative, substance j suppresses the synthesis (degradation) of substance i . If g_{ij} (h_{ij}) is zero, substance j has no effects on the synthesis (degradation) process of substance i .

Inference of genetic networks by S-system is formalized as a search problem in which appropriate system parameters of S-system must be found so that the difference between the time course data calculated by the S-system model and

the time course data observed in experiments is less than a tolerance given by a user. In NSS-EA [18], the following two evaluation functions are used:

$$f_{\text{MeanSquaredError}} = \sum_{i=1}^n \sum_{t=1}^T \left(\frac{X_i^{\text{cal}}(t) - X_i^{\text{exp}}(t)}{X_i^{\text{exp}}(t)} \right)^2 \quad (2)$$

$$f_{\text{MaxError}} = \max_{\substack{1 \leq i \leq n \\ 1 \leq t \leq T}} \left| \frac{X_i^{\text{cal}}(t) - X_i^{\text{exp}}(t)}{X_i^{\text{exp}}(t)} \right| \quad (3)$$

where T is the number of sampling points, $X_i^{\text{exp}}(t)$ is the value of X_i observed in experiments at time t , $X_i^{\text{cal}}(t)$ is the value of X_i at time t obtained by solving S-system. This problem has the difficulty of high-dimensionality, multimodality and strong non-linearity. This problem also has the difficulty that there is no way to determine a network structure uniquely. This is because sufficient time-course data to determine a network structure uniquely cannot be obtained in experiments because of difficulty and high cost of experiments. For this reason, a search engine should find as many network structures that explain the experimentally-observed time-course and satisfy biological knowledge as possible. For example, it is known that the number of substances having effects on a single substance is relatively small, which means that many of g_{ij} and h_{ij} are zero, in actual genetic networks. Two network structures are judged to be the same when the interaction among the all substances are the same between the two networks, *i.e.* the signs of all the interaction coefficients of S-system are the same between the two networks.

2.2 NSS-EA

Overview of NSS-EA. Morishita *et al.* have proposed a search method for efficiently finding multiple biologically-promising network structures that explain experimentally-observed time-course data well, named Network-Structure-Search Evolutionary Algorithm (NSS-EA) [18]. As shown in Fig. 1, NSS-EA consists of two parts: the search for network structures, named the *structure search*, and the search for the system parameters of each network structure, named the *parameter search*. By separating the structure search and the parameter search explicitly, NSS-EA can sample only biologically-promising network structures intensively by controlling the search process of network structures.

Morishita *et al.* compared the performance of NSS-EA and UNDX+MGG-SS [27]. UNDX+MGG-SS is a conventional method based on a GA and has been reported that it showed good performance. Morishita *et al.* applied NSS-EA and UNDX+MGG-SS to a five-substance genetic-network estimation problem. In the case of no limited number of substance-substance interactions, NSS-EA succeeded in finding 207 different satisfactory structures while UNDX+MGG-SS found only 19 ones. In the case where the number of substances interacting with a substance is limited to three, NSS-EA found 100 different satisfactory structures while UNDX+MGG-SS did not [18].

The algorithms of the structure search and the parameter search in NSS-EA are detailed in the followings.

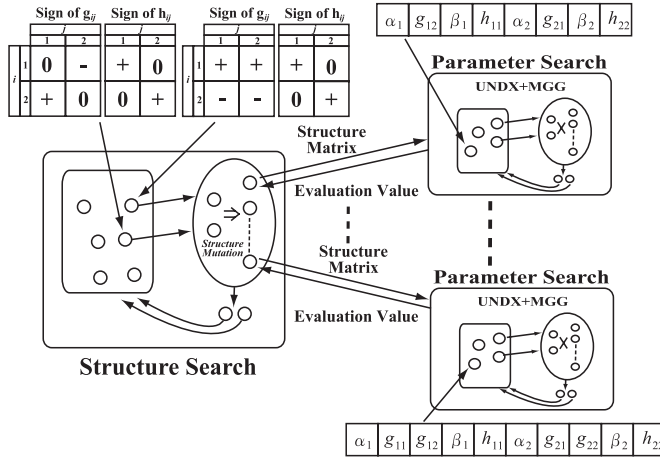


Fig. 1. Network-Structure-Search Evolutionary Algorithm (NSS-EA) [18]

Structure Search. As shown in Fig. 1, the structure search employs the structure matrix whose elements are $s_{g_{ij}}$ and $s_{h_{ij}}$ as an individual representation. $s_{g_{ij}}$ and $s_{h_{ij}}$ are the signs of interaction coefficients between substances, g_{ij} and h_{ij} , respectively, where $s_{g_{ij}} = '+'$ ($s_{h_{ij}} = '+'$) if g_{ij} (h_{ij}) is positive, $s_{g_{ij}} = '-'$ ($s_{h_{ij}} = '-'$) if g_{ij} (h_{ij}) is negative, and $s_{g_{ij}} = '0'$ ($s_{h_{ij}} = '0'$) if g_{ij} (h_{ij}) is zero. The structure search uses the structure mutation as a structure-search operator. The structure mutation changes the value of a randomly-chosen element to another allele randomly. The generation-alternation model is a model based on Minimal Generation Gap (MGG) [22]. The algorithm of the structure search is as follows:

1. *Generation of initial population*

Generate $n_{\text{pop}}^{\text{SS}}$ individuals with structure matrices randomly and let the individuals be an initial population. Here, the number of substances interacting with each substance is limited to one, which means that the number of elements whose values are other than '0' in a row of a structure matrix is one. More than two individuals with the same matrix are not allowed to exist in the population. Then, evaluate each individual in the population by the parameter search described below.

2. *Selection for reproduction*

Choose two parents randomly and remove them from the population.

3. *Generation of kids*

Apply the structure mutation to each parent $n_{\text{kid}}^{\text{SS}}$ times to generate $2n_{\text{kid}}^{\text{SS}}$ kids.

4. *Evaluation of kids*

For each kid generated in step 3, perform the parameter search to obtain the evaluation value of the kid, f_{MaxError} . In order to reduce the search time, individuals and their evaluation values are cached if the evaluation value is less than f_{MaxError}^* or is larger than λf_{best} , where λ is a constant given by

a user and f_{best} is the evaluation value of the best individual in the current population, and reuse them when the same individuals are generated in step 3. In order to obtain various network structures, individuals with the evaluation values of $f_{\text{MaxError}} \leq f_{\text{MaxError}}^*$ are saved in a file, where f_{MaxError}^* is given by a user in advance.

5. *Selection for survival*

Remove the same individuals as ones included in the population from the family consisting of the parents and their kids. Select the best and second best individuals from the family and add them to the population.

6. Repeat the above steps from two to five $n_{\text{itr}}^{\text{SS}}$ times.

Parameter Search. In the parameter search, the system parameters of S-sysmtem, α_i , β_i , g_{ij} and h_{ij} , are searched according to the structure matrix passed by the structure search as shown in Fig. 1. In the case of $s_{g_{ij}} = '+'$ ($s_{h_{ij}} = '+'$) in the structure matrix, the search range of g_{ij} (h_{ij}) is limited to positive. In the case of $s_{g_{ij}} = '-'$ ($s_{h_{ij}} = '-'$), the search range of g_{ij} (h_{ij}) is limited to negative. In the case of $s_{g_{ij}} = '0'$ ($s_{h_{ij}} = '0'$), the search range of g_{ij} (h_{ij}) is fixed to zero. As described above, the value f_{MaxError} of the best solution found in the parameter search is returned as the evaluation value of the structure matrix to the structure search. The parameter search employs Unimodal Normal Distribution Crossover (UNDX) [20] as a crossover operator and MGG [22] as a generation-alternation model. The number of parameters to be searched changes depending on the number of $s_{g_{ij}}$ and $s_{h_{ij}}$ whose values are '0'. From a viewpoint of search efficiency, the population size is set to $n_{\text{pop}}^{\text{PS}} = \gamma_{\text{pop}} n_{\text{param}}$, the number of applying UNDX to a pair of parents in a generation is set to $n_{\text{kid}}^{\text{PS}} = \gamma_{\text{kid}} n_{\text{param}}$ and the number of iterations for a trial of the parameter search is set to $n_{\text{itr}}^{\text{PS}} = \gamma_{\text{itr}} n_{\text{pop}}^{\text{PS}}$, where n_{param} is the number of $s_{g_{ij}}$ and $s_{h_{ij}}$ whose values are '+' or '-' and γ_{pop} , γ_{kid} , γ_{itr} are constants given by a user in advance. The evaluation value of an individual in the parameter search is the value of $f_{\text{MeanSquaredError}}$.

3 Requirements for Implementation of NSS-EA

In NSS-EA, it takes a long time to calculate the evaluation value of a structure matrix by performing the parameter search because a complex nonlinear differential equation system must be solved enormous times. In order to remedy this problem, NSS-EA adopted a master-worker model in which the process of the structure search is performed on a master node and the processes of the parameter search are done on multiple worker nodes in parallel. In order to enable NSS-EA to run on a multi-PC-cluster grid computing environment where multiple PC clusters are connected over the Internet, the "Gridified" NSS-EA should satisfy the following requirements:

– *Security*

1. Secure user authentication when a worker process is invoked on a remote node and a connection is made over the Internet
2. Data encryption and decryption when exchanged over the Internet

- *Flexibility*
 1. Dynamic addition and deletion of worker nodes without stopping computation
- *User-friendliness*
 1. Single-sign-on authentication over the Internet
 2. Automatic file transfer mechanism to the remote nodes necessary to run programs
 3. Temporal resource clean up mechanism after program execution
 4. Remote error message report mechanism
 5. Interactive process invocation and termination on remote nodes
 6. Minimum overhead for the start-up time
- *Robustness*
 1. Dynamic node-separation mechanism when a master cannot communicate with a certain worker due to some trouble in the worker node and/or network
 2. Checkpoint and restart mechanism when a master node fails due to some troubles
- *Scalability*
 1. Performance improvement as the number of workers increases

4 Grid-Oriented Genetic Algorithm Framework (GOGA Framework)

In order to facilitate the development of Grid-Oriented Genetic Algorithm (GOGA), we have developed a framework named the GOGA Framework. A developer can realize the requirements of *security*, *flexibility* and *user-friendliness* only by using the GOGA Framework. The requirements of *robustness* and *scalability* can be also easily realized by using the GOGA Framework.

4.1 Genetic Algorithm (GA)

Generally, GA models can be categorized into the *single population model* and the *multiple population model*. A developer can easily “Gridify” both of the single population model and the multiple population model by using the GOGA Framework.

The single population model is a model in which one population are evolved by genetic operators. Simple GA [5] and MGG [22] are its examples. Generally, its algorithm can be described as follows: 1) generate an initial population randomly (generation of initial population), 2) choose parents from the population (selection for reproduction), 3) generate kids by crossover and mutation (generation of kids), 4) calculate the evaluation values of the kids (calculation of evaluation values), 5) select the individuals surviving in the next generation (selection for survival), and 6) repeat the above steps from two to five until a terminating condition is satisfied. This model can be parallelized by assigning

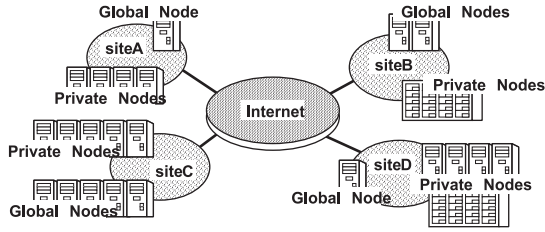


Fig. 2. Grid Computing Environment

the population data and the processes other than *calculation of evaluation values* to one node, named *master*, and assigning the processes of *calculation of evaluation values* to multiple nodes, named *workers*.

The multiple population model is a model in which multiple sub-populations are evolved independently and interact with each other periodically. Island model [24] and DuDGA [9] are its examples. Generally, its algorithm can be described as follows: 1) generate multiple initial sub-populations (generation of initial population), 2) apply genetic operators to each sub-population (independent evolution), and 3) exchange information, such as the best individuals in each sub-population, among sub-populations periodically (interactions among sub-populations). This model can be parallelized by assigning each sub-population and the processes of *independent evolution* to each node, named *peer*.

4.2 Grid Environment

The GOGA Framework assumes a grid computing environment in which multiple sites connect to each other via the Internet and each site includes *global nodes* and *private nodes*, as shown in Fig. 2. A global node has a global IP and can be directly accessed from other sites. A private node has a global or private IP and can not be directly accessed from other sites.

4.3 Programming Language and Middleware

The GOGA Framework employs Java as a programming language from the viewpoints of robustness, flexibility, portability and hiding network latency. Java is an object-oriented programming language that provides powerful multithread management, memory management, exception handling functions and good portability. The GOGA Framework uses Globus Tool Kit (GTK) [4], which is a defacto standard middleware for grid programming. We also use CoG Tool Kit [14] to access GTK with Java. GTK provides functions for authenticating users based on X.509 certificates, enabling single-sign-on, invoking processes on remote global nodes, transferring files and standard input, output, or error messages to remote global nodes, and encrypting data transferred between global nodes. Note that GTK cannot handle private nodes behind NAT.

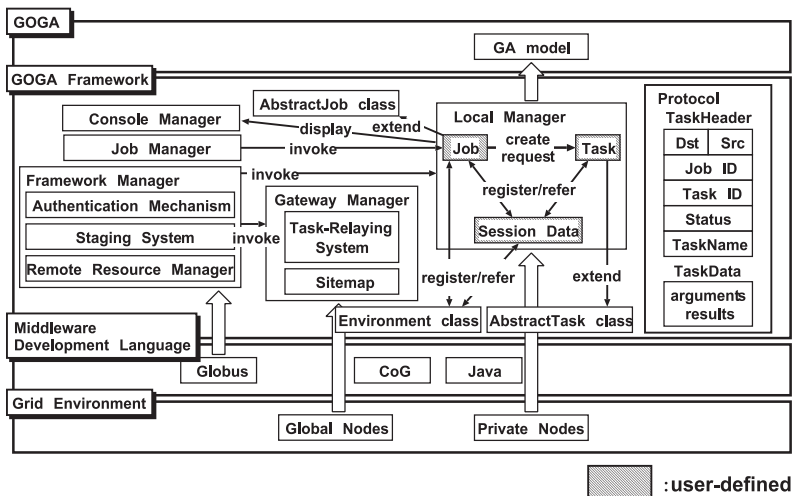


Fig. 3. Architecture of GOGA Framework

4.4 Architecture

Figure 3 shows the architecture of the GOGA Framework. In the GOGA Framework, a persistent process on a node such as the process of generation alternation on a master and the process of *independent evolution* on each peer is called a *job*. A transient process invoked by other nodes such as the request of *calculation of evaluation values* and the request of *interactions among sub-populations* is called a *task*. A persistent data on a node such as a population on a master or a peer is called *session data*. The GOGA Framework consists of the following four managers.

- *Local manager*

The local manager program manages jobs and session data and handles tasks received from other local managers on each private node. The processes of sending, receiving and performing tasks and that of executing jobs are assigned to independent threads, respectively, so that GOGA developers can easily realize to hide network latency and to handle troubles by terminating the corresponding job threads. This program requires Java and the rsh server.

- *Gateway manager*

The gateway manager program runs on more than one global node in each site and relays a task between sites over the Internet according to a grid node database named the *sitemap*. In the site map, a node is specified by a host-name and a port number, called the *contact string*. All the tasks exchanged among gateway managers are encrypted by SSL. This program requires Java, CoG, GTK and rsh/rcp clients.

- *Framework manager*

The framework manager program is used for starting up the GOGA Framework. This program runs on a certain private node that the user uses as a

user-terminal node. The framework manager, first, performs authentication process and, then, transfers necessary files to other nodes. After that, the framework manager invokes gateway managers on global nodes and local managers on private nodes. This program requires Java and CoG.

- *Job manager*

The job manager program is used for invoking jobs on local managers. This program runs on a user-terminal node. This program requires Java and CoG.

- *Console manager*

The console manager program is used for displaying messages received from tasks and jobs on remote nodes. This program requires Java and CoG.

4.5 Programming Interfaces

The GOGA Framework provides the following classes to enable a GOGA developer to easily “Gridify” parallel GAs:

- *AbstractJob class*

This class provides methods for sending and receiving user tasks, initialization and job processing. Errors during requesting tasks are notified to the user-defined codes of job processes.

- *AbstractTask class*

This class provides methods for user tasks and the serialization/deserialization of necessary data to perform the task.

- *Environment class*

By using Environment class, jobs and tasks can register any objects as session data and any jobs to the local manager, to refer them and to delete them. This class also provides two methods, named *printStringsToStandardOutput* and *printStringsToStanardError*, for displaying any messages on the console manager.

4.6 Behavior

The framework manager, first, generates a certificate by receiving user’s pass phrase and, then, performs authentication processes on global nodes on each site. Next, the framework manager sends all necessary files to all the nodes on the grid and, after that, invokes the gateway managers on global nodes and local managers on private nodes.

The job manager, first, reads *job files*. A job file includes the class name of a job, the node name where the job is to be invoked and some arguments for the initialization. Then, the job manager invokes the jobs on the specified local managers. Figure 4 shows a typical behavior of the framework when a job requests a task.

4.7 Improvements from the Original Version

The current version of the GOGA Framework has been improved from the original version [11] in the following points:

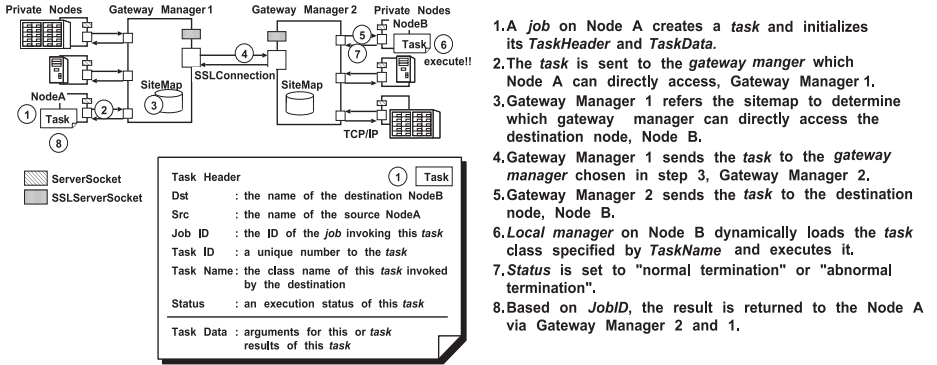


Fig. 4. Behavior of GOGA Framework

- Even if the framework manager terminates abnormally, all the processes and files are surely deleted on remote nodes.
- When a task made by a job is not normally executed on a specified destination node due to some troubles with the communication path, the remote node or the task, a task with a header including detailed information on *what kind of error has occurred* and *where the error has occurred*, named an *error task*, is surely returned to the source job.
- User interfaces for interactively accepting commands from a user are added to the framework manager and the job manager.
- The gateway manager is modified so that remote gateway managers and local managers are activated in parallel by using threads.
- The *console manager* program is added.
- Two methods for outputting messages to the standard output and the standard error of the user terminal are added to the Environment class.
- The local manager is improved so that a local manager can directly send a task to another local manager in the same site.

5 The “Gridified” NSS-EA Using the GOGA Framework

We “Gridify” NSS-EA by using the GOGA Framework described in the previous section. Figure 5 shows the architecture of the “Gridified” NSS-EA. The “Gridified” NSS-EA employs a master-worker model. From the viewpoint of *scalability*, the process of the structure search is assigned to a master and the processes of the parameter search to multiple workers. This is because the data size of structure passed from the structure search to the parameter search is small and the time required for performing the parameter search is long. The “Gridified” NSS-EA consists of four kinds of jobs that are the subclasses of the *AbstractJob* class, three kinds of tasks that are the subclasses of the *AbstractTask* class and two kinds of session data.

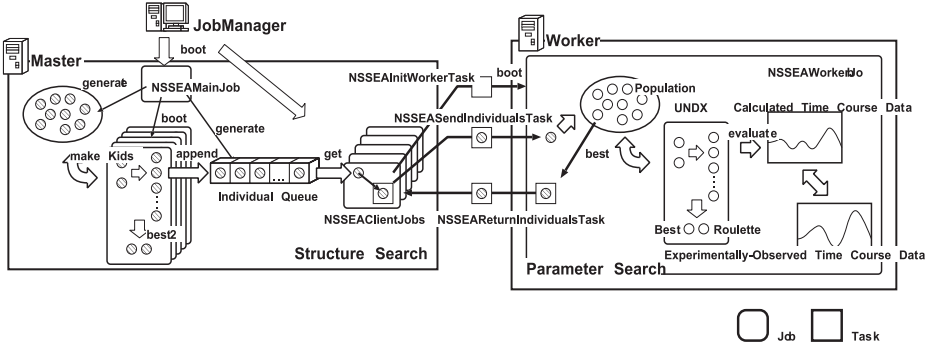


Fig. 5. Architecture of the "Gridified" NSS-EA

The NSSEAGenerationAlternationJob is a job to handle the process of generation alternation for the structure search. There are multiple NSSEAGenerationAlternationJobs running on the master node so that individuals enough to keep all the workers operating are provided to the individual queue. From the viewpoint of efficiency, while an NSSEAGenerationAlternationJob chooses the pair of parents and makes multiple kids to add them to the individual queue, the others are not allowed to operate in parallel. This is realized by using the lock mechanism provided by the thread library of Java.

The NSSEAClientJob is a job to communicate with a worker job named the NSSEAWorkerJob. There are multiple NSSEAClientJobs running on the master node. An NSSEAClientJob invokes an NSSEAWorkerJob on a local manager running on a remote worker node by sending an NSSEASendIndividualsTask. Then, the NSSEAClientJob keeps requesting the NSSEAWorkerJob to evaluate individuals by sending an NSSEASendIndividualsTask. If the NSSEAClientJobs cannot communicate with the NSSEAWorkerJob during calculation, due to some troubles, the NSSEAClientJobs terminates after notifying to NSSEAGenerationAlternationJobs appropriately.

The NSSEAWorkerJob is a job to handle the process of the parameter search. An NSSEAWorkerJob runs on a local manager on a worker node. It receives a structure matrix from an NSSEASendIndividualsTask, performs the parameter search based on the structure matrix and returns the results by sending an NSSEAReturnIndividualsTask to the corresponding NSSEAClientJob.

6 Experiments

By using PC clusters of the University of Tokushima and Kyushu University on Open Bioinformatics Grid (OBIGrid) [13] in Japan, we confirmed that the proposed "Gridified" NSS-EA satisfied all the requirements discussed in section three well. In OBIGrid, sites are connected by using VPN over the Internet. As shown in Fig. 6, in this experiment, we used one single-CPU PC and 58 dual-CPU PCs in the site of the University of Tokushima and one single-CPU

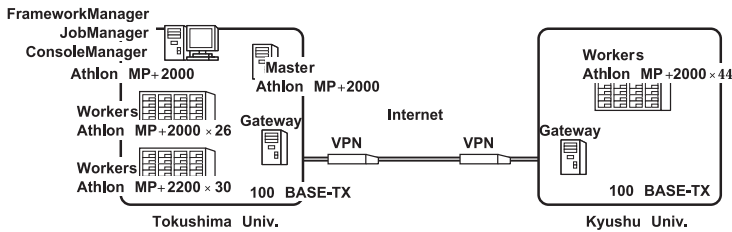


Fig. 6. Grid testbed on OBIGrid

PC and 44 dual-CPU PCs in the site of Kyushu University. The single-CPU PCs are the gateway nodes of each site. We used a dual-CPU PC as a user terminal node and another dual-CPU PC as a master node in the site of the University of Tokushima and the rest of 100 dual-CPU PCs as worker nodes. We executed the framework manager, the job manager and the console manager on the user terminal node, the gateway managers on the gateway nodes of each site and local managers on the master node and the worker nodes. We invoked an NSSEAMainJob, NSSEAGenerationAlternationJobs and NSSEAClient-Jobs on the master node and NSSEAWorkerJobs on the worker nodes. We applied the proposed “Gridified” NSS-EA to a five-gene genetic network estimation problem.

In the followings, we detail the experiments done for confirming that the proposed “Gridified” NSS-EA satisfies the requirements of *robustness* and *scalability*.

[Robustness]: Recovery from Artificial Troubles on Worker Nodes. We confirmed that the proposed “Gridified” NSS-EA detected troubles with worker nodes and safely separated the troubles without stopping the calculation in the following cases:

- Power shut down on a worker node in execution
- Pulling out a network cable from a worker node
- Killing NSSEASendIndividualsTask, NSSEASendIndividualsTask and NSSEAReturnIndividualsTask in execution

[Scalability]: Performance Improvement as the Number of Workers Increases. We increased the number of worker nodes from one to 200. Figure 7 (left) shows the number of workers versus the time required for 20 generations. Figure 7 (right) shows the number of workers versus the speed up rate, where the speed of one worker is supposed to be one. The average time required for a single evaluation to be accomplished on a worker is distributed between six and eight seconds. The average size of NSSEASendIndividualsTask and that of NSSEAReturnIndividualsTask are about 220 and 990 bytes, respectively. As shown in Fig. 7, the proposed “Gridified” NSS-EA shows good scalability.

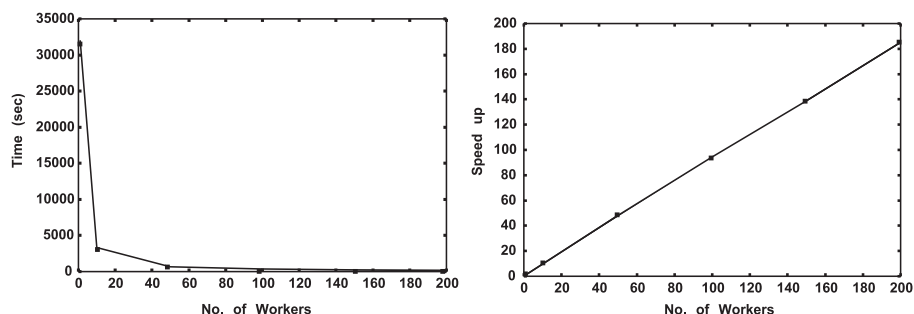


Fig. 7. Time required for 20 generations (left) and speed-up rate (right)

7 Conclusion

In this paper, we proposed the “Gridifid” NSS-EA by using the Grid-Oriented GA (GOGA) Framework. We confirmed that the proposed “Gridified” NSS-EA works correctly and evaluated its performance on the Open Bioinformatics Grid (OBIGrid) in Japan.

We are now developing a mechanism for automatically rebooting workers when worker nodes are shut down due to some troubles with the nodes or network. A resource management mechanism for automatically balancing computational resources among users is also under development.

Acknowledgements. This work was partially supported by the Grants-in-Aid for Scientific Research on Priority Areas, “Genome Information Sciences” (No.12208008) from the Ministry of Education, Culture, Sports, Science and Technology in Japan. We thank all the participants of OBI Grid and Mr. Morishita and Mr. Seike of The University of Tokushima.

References

1. Ando, S., Iba, H.: Inference of Gene Regulatory Model by Genetic Algorithms. Proc. Con-gress on Evolutionary Computation 2001, (2001) 712–719
2. Ando, S., Sakamoto, E., Iba, H.: Modeling Genetic Network by Hybrid GP, Proc. Congress on Evolutionary Computation 2002 (CEC2002), (2002) 291–296
3. BioGrid. <http://www.biogrid.jp>
4. Foster, I., Kasseleman C.: Globus: A metacomputing infrastructure toolkit. Int’l Journal of Supercomputing Applications. 11(2) (1997) 115–128
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addi-son-Wesley Publishing Company Inc. (1989)
6. Gordon, V.S., Whitley, D.W.: Serial and Parallel Genetic Algorithms as Function Optimiz-ers. Proc. of the Fifth Int’l Conf. on Genetic Algorithms (1993) 434–439
7. Gorges-Scheluter, M.: ASPARAGOS An Asynchronous Parallel Genetic Optimiza-tion Strategy. Proc. of the Third Int’l Conf. on Genetic Algorithms, (1989) 422–427

8. Hamilton-Wright, A., Stacey, D.: Fault-Tolerant Network Computation of Individuals in Genetic Algorithms. Congress on Evolutionary Computation 2002 (2002) 1721–1726
9. Hiroyasu, T., Miki, M., Hamasaki, M., Tanimura, Y.: A New Model of Distributed Genetic Algorithm for Cluster Systems: Dual Individual DGA. Proc. of the Int’l Conf. on Parallel and Distributed Processing Techniques and Applications, Vol.1 (2000) 477–483
10. Iba, H., Mimura, A.: Inference of Gene Regulatory Network by means of Interactive Evolutionary Computing, Information Sciences, Vol. 145, No. 3-4, (2002) 225–236
11. Imade, H., Morishita, R., Ono, I., Ono, N., Okamoto, M.: A Grid-Oriented Genetic Algorithm Framework for Bioinformatics. New Generation Computing, Vol.22 (2004) 177–186
12. Kimura, S., Hatakeyama, H., Konagaya, A.: Inference of S-system Models of Genetic Networks using a Genetic Local Search. Proc. 2003 Congress on Evolutionary Computation (CEC2003), (2003) 631–638
13. Konagaya, A., Konishi, F., Hatakeyama, M. And Satou, K.: The Superstructure towards Open Bioinformatics Grid, New Generation Computing, 22 (2004) 167–176
14. Laszewski, G., Foster, I., Gawor, J., Smith, W., Tuecke, S.: CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids. In ACM 2000 Java Grande Conf. (2000) 97–106
15. Lee, C.H., Park, K.H., Kim, J.H.: Hybrid Parallel Evolutionary Algorithms for constrained optimization utilizing PC Clustering. Congress on Evolutionary Computation 2001 (2001) 1436–1441
16. Maki, Y., Tominaga, D., Okamoto, M., Watanabe, S., Eguchi, Y.: Development of a System for the Inference of Large Scale Genetic Networks, Proc. Pacific Symp. on Biocomputing, (2001), 446–458
17. Maki, Y., Ueda, T., Okamoto, M., Uematsu, N., Inamura, Y., Eguchi, Y.: Inference of Genetic Network Using the Expression Profile Time Course Data of Mouse P19 Cells, Genome Informatics, Vol. 13, (2002), 382–383
18. Morishita, R., Imade, H., Ono, I., Ono, N., Okamoto, M.: Finding Multiple Solutions Based on An Evolutionary Algorithm for Inference of Genetic Networks by S-system. Proc. 2003 Congress on Evolutionary Computation (CEC2003) (2003) 615–622
19. North Carolina Bioinformatics Grid. <http://www.ncbiogrid.org>
20. Ono, I., Kobayashi, S.: A real-coded genetic algorithm for function optimization using Unimodal Normal Distribution Crossover. Proc. of the Seventh Int’l Conf. on Genetic Algorithms (1997) 246–253
21. Sakamoto, E., Iba, H.: Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming. Proc. 2001 Congress on Evolutionary Computation (CEC2001) (2001) 720–726
22. Sato, H., Yamamura, M., Kobayashi, S.: Minimal generation gap model for GAs considering both exploration and exploitation. Proc. of 4th Int. Conf. on Soft Computing (1996) 494–497
23. Savageau, M.A.: Biochemical Systems Analysis: A Study of Function and Design in Molecular Biology. Addison-Wesley, Massachusetts (1976)
24. Tanese, R.: Distributed Genetic Algorithms. Proc. of the Third Int’l Conf. on Genetic Algorithms (1989) 434–439
25. Tominaga, D., Okamoto, M.: Design of canonical model describing complex nonlinear dynamics, Computer Applications in Biotechnology 1998 (CAB7), Pergamon Press, Oxford, (1998) 85.

26. Tominaga, D., Koga, N., Okamoto, M.: Efficient Numerical Optimization Algorithm Based on Genetic Algorithm for Inverse Problem. Proc. of the Genetic and Evolutionary Computation Conf. (2000) 251–258
27. Ueda, T., Koga, N., Ono, I., Okamoto, M.: Efficient Numerical Optimization Technique Based on Read-Coded Genetic Algorithm for Inverse Problem. Proc. 7th Int'l Symp. On Artificial Life and Robotics (AROB'02) (2002) 290–293
28. Voit, E.O.: Computational Analysis of Biochemical Systems. A Practical Guide for Bio-chemists and Molecular Biologists, Cambridge University Press, Cambridge 2000 (2000)

Author Index

- Abdul Salam, Rosalina 92
Alex Grey, W. 140
Ang, Larry 53
Arzberger, Peter W. 53, 68
- Bayer, Micha 125
Berry, David 125
Bhagwat, Shonil 140
Birnbaum, Adam 68
Bisby, Frank A. 140
Bourne, Philip E. 53, 68
Brewer, Peter 140
Bromley, Oliver 140
Byrnes, Robert W. 53
- Caithness, Neil 140
Casanova, Henri 68
Chuon, Danny 53
Culham, Alastair 140
- Date, Susumu 43
Defago, Xavier 8
- Ferrier, Magnus 125
Fiddian, Nick J. 140
Fukuzaki, Akinobu 82
- Hanada, Yoshiko 152
Hatakeyama, Mariko 82
Hayes, James 68
Hiroyasu, Tomoyuki 152
Houghton, Derek 125
- Ide, Kaori 82
Imade, Hiroaki 171
Ito, Daiki 20
- Jones, Andrew C. 140
- Kitayama, Tomoya 20
Kian, Ng Lip 117
Konagaya, Akihiko 8, 32, 82
Konishi, Fumikazu 32, 82
- Kosaka, Takahiro 43
Kuramitsu, Seiki 82
- Li, Wilfred W. 53, 68
- Mahadi, Nor Muhammad 117
Mat Isa, Mohd Noor 117
Matsuda, Hideo 43
Matsuoka, Satoshi 53, 103
Miki, Mitsunori 152
Miller, Mark A. 53, 68
Mizuguchi, Naoaki 171
Mohamed, Rahmah 117
- Nagashima, Takeshi 82
Nakada, Hidemoto 103
Nakashima, Yasuhiko 8
- Okamoto, Masahiro 171
Okamoto, Yuko 152
Ono, Isao 171
Ono, Norihiko 171
- Pittas, Nick 140
- Raih, Mohd Firdaus 117
Raja Moktar, Raja Murzaferi 117
Rosni, Abdullah 92
- Satou, Kenji 8
Scoble, Malcolm 140
Shahab, Atif 53
Sharum, Mohd Yunus 117
Shimodaira, Hidetoshi 103
Shimojo, Shinji 43
Sinnott, Richard 125
Sugawara, Hideaki 1
Sugimoto, Masahiro 20
Sutton, Tim 140
Suzumua, Toyotaro 53
- Takahashi, Kouichi 20
Tanaka, Kouji 53
Tohsato, Yukako 43

Tomita, Masaru 20

Tsuji, Shin'ichi 8

White, Richard J. 140

Williams, Paul 140

Wooi Keat, Martin Chew 92

Xu, Xuebiao 140

Yamamoto, Yo 103

Yesson, Chris 140

Yokoyama, Shigeyuki 82