G-BLAST: a Grid-Based Solution of mpiBLAST on Computational Grids

Chao-Tung Yang Tsu-Fen Han
High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University
Taichung City, 40704, Taiwan R.O.C.
ctyang@thu.edu.tw g942814@thu.edu.tw

ABSTRACT

Research in the area of bioinformatics has grown with each passing day in recent years as demands for more computing power increased. The solutions to these demands usually involve using parallelism techniques. For example, mpiBLAST is a parallel program that executes jobs in parallel in clusters with MPI. Although Cluster environments can reduce the execution time and increase alignment efficiency, they may not be a good solution when aligning very large genomic databases. Grid Computing can coordinate the resources of distributed virtual organizations and satisfy a great many computational demands. In addition to integrating distributed resources, Grid Computing can reduce server idle times via management of integrated computing resources. This study proposes an approach that integrates mpiBLAST in a Grid Computing environment called BioGrid, and implementation of a Grid service called G-BLAST. In here, the G-BLAST is compliant with the Web Service Resource Framework (WSRF) standard.

1: INTRODUCTION

Bioinformatics combines biology and information technology and includes computational tools and methods for managing, analyzing, and manipulating large biology datasets. Computing technologies are vital for bioinformatics applications [1][2][3]. For example, biology problems often require repeating a task millions of times, such as when searching for sequence similarities in existing databases or comparing groups of sequences to determine evolutionary relationships. In such cases, high-performance computers to process this information are indispensable. Biological information is stored on many computers around the world. The easiest way to process this data is to join these computers together through network. Such activities require high-performance computing infrastructures [4][5] with access to huge information databases [6]. The major advances in computer technology and computer science over the past 30 years have dramatically changed much of our society. Currently, many parallel versions of bioinformatics applications can be used to conduct computing tasks on Linux PC cluster or Grid systems.

NCBI BLAST [7] is one of the best tools available to bioinformatics for using heuristic similarity-search algorithms to find sequences in genome databases similar to given query sequences. mpiBLAST [8] is a parallelization of BLAST that executes BLAST jobs in parallel using Message Passing Interface (MPI) [9]. Even with this parallel version, mpiBLAST, BLAST still needs sufficient time to analyze and align genomic databases.

Grid [10][11] enable virtual organizations to share geographically distributed resources in their pursuit of common goals, assuming the existence of a central location, central control, omniscience, and an existing trust relationship.

The BioGrid [2][12][13] is a Grid environment aimed at improving the performance of bioinformatics applications. In this work, we propose BioGrid Framework, a Grid-based solution for mpiBLAST built on the Globus Toolkit 4.0 [14] that integrates mpiBLAST into a Grid environment. Moreover, we report on implementing a G-BLAST Service that allows use of mpiBLAST, which is integrated into BioGrid. The purpose of this study is to get higher performance than traditional Cluster architectures and to have an easier user interface than previous mpiBLAST versions. Therefore, this present approach focuses on how to integrate mpiBLAST into a Grid environment, and attempts to build a hierarchical architecture that integrates a Grid environment and a cluster environment. The GUI implemented is used to perform users' invocations to BioGrid. The G-BLAST Service provides functions for managing submissions, segmenting genomic databases, dispatching and monitoring Grid jobs, combining result files, and returning results via WSRF [15].

The rest of this paper is organized as follows. In Section 2 we describe what Grid Computing and Gird Middleware are, and in Section 3 we describe the application, framework, and technology that G-BLAST uses. Development and details of G-BLAST are presented in Section 4. Conclusions are given in Section 5, with emphasis on potential areas for future work.

2: Background Review

2.1: Grid Computing

The main concept of Grid Computing [11][16] is to extend the original ideas of the Internet to sharing widespread computing power, storage capacities, and other resources. Grid Computing can coordinate the resources of distributed virtual organizations and satisfy a great quality of computational demands. Besides integrating distributed resources, Grid Computing can also reduce idle time of servers via management of integrated computing resources. Owing to a deluge of data and information, fields such as high-energy physics, bioinformatics, and digital archive, demand a great deal of computational and storage capacity.

2.2: Grid Middleware

The Globus Project provides software tools that make it easier to build computational Grid systems and applications. These tools are collectively called the Globus Toolkit. The toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. We used it as the infrastructure of our BioGrid.

The composition of the Globus Toolkit can be pictured as three pillars: Resource Management, Information Services, and Data Management. Each pillar represents a primary component of the Globus Toolkit and makes use of a common foundation of security. The Globus alliance proposed a common data transfer and access protocol called GridFTP. It is a high-performance, secure, efficient data movement, and reliable data transfer protocol optimized for Grid environments. The GridFTP protocol extends the standard FTP protocol, the highly-popular Internet file transfer protocol, and provides a superset of the features offered by the various Grid storage systems currently in use.

MPICH-G2 [17] is a grid-enabled implementation of the MPI v1.1 standard. In addition, MPICH-G2 allows coupling of multiple machines, with different architectures, to run MPI applications. MPICH-G2 automatically converts data in messages sent between machines of different architectures and supports multiprotocol communication by automatically selecting TCP for intermachine messaging and (where available) vendor-supplied MPI for intramachine messaging.

3: Parallel Bioinformatics Applications

Comparing a sequence against a database is one of the most common bioinformatics applications. A sequence alignment is needed before making comparative statements about nucleic acid or protein sequences. The concept of selecting an optimal sequence alignment is simple, but is not at all simple in practice. Choosing a good alignment artificially is possible, but it must do more than once or twice. An automatic method for finding the optimal alignment out of the thousands of alternatives is the right approach.

A common application of sequence alignment is searching a database for sequences similar to a query sequence. Hence, there are many sequence alignment tools, such as BLAST [5] [18] and FASTA, based on various algorithms. There are vast volumes of DNA sequence data, and we need to figure out which parts of that DNA control the various chemical processes of life and determine the functions of new proteins from the known functions and structures of some proteins. Availability of computer resources is the key factor limiting use of bioinformatics analyses as a result of the growing computational demands. Various databases of gene/protein sequences, gene expression, and related analysis tools help scientists determine whether and how a particular molecule is directly involved in a disease process. That, in turn, aids in the discovery of new and better drug targets [11] [19].

3.1: mpiBLAST

The Basic Local Alignment Search Tool (BLAST) is a sequence database search tool that seeks similarities between two substrings in molecular biology by using score matrices to improve filtration efficiency and to introduce more accurate rules for locating potential matches. BLAST attempts to find all locally maximal segment pairs in query sequences and database sequences with scores above some set threshold. mpiBLAST is a freely available, open-source parallelization of NCBI BLAST. It contains a pair of programs that replace formatdb and blastall with versions that execute BLAST jobs in parallel on clusters of computers with MPI installed.

There are two primary advantages to using mpiBLAST rather than conventional BLAST. First, mpiBLAST segments a target database, then dispatches the segments to nodes in clusters. Because the database segment in each node is small, it can usually reside in the buffer-cache, yielding a significant speedup due to the elimination of disk I/O. Second, it allows BLAST users to take advantage of efficient, low-cost Beowulf clusters because interprocessor communication demands are low. Table 1 shows Appropriate Query/Program Combinations of BLAST.

Table 1. Appropriate Query/Program Combinations for "BLAST 2 Sequences"

First Query	Second Query	Program to Use
Nucleotide	Nucleotide	*blastn, megablast, or tblastx*
Nucleotide	Protein	*blastx*
Protein	Nucleotide	*tblastn*
Protein	Protein	*blastp*

The mpiBLAST Partitioning schema is shown in Figure 1. It uses multithreading to segment databases, assigning distinct portions of the database to each processor. It wraps the standard NCBI formatdb called mpiformatdb to format the Database. Command line arguments specify the number of fragments. mpiformatdb formulates command line arguments that force NCBI formatdb to format and divide the database into many fragments of approximately equal size.

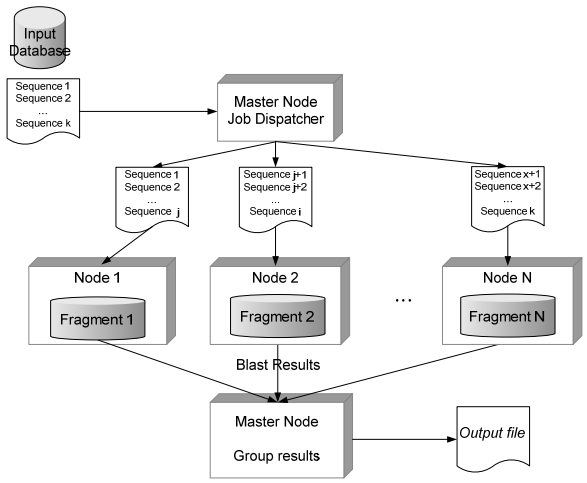

Figure 1. The mpiBLAST partitioning schema

When mpiformatdb execution is complete, the formatted fragments are placed in shared storage. Alignment of the database is accomplished by the local sequence alignment algorithm implemented in the NCBI [20] development library. If a node does not have fragments needed by a search, the fragments are copied from shared storage. Fragments are assigned to nodes using an algorithm that minimizes the number of fragments copied during each search.

3.2: BioGrid Framework

This project constructed the BioGrid framework shown in Figure 2. It is a Virtual Grid Organization that includes many sites. The framework can contain many bioinformatics applications, such as mpiBLAST and FASTA, among others. This present approach considers how to integrate mpiBLAST into a Grid environment.

In BioGrid, sites can be PC Clusters, PCs, or super computers. This project focuses on the Beowulf Cluster. Each site must have Fedora Core 4 Linux OS, Globus Toolkit 4.0 (GT4), and MPICH-G2 installed. Because mpiBLAST performs on clusters, we let a cluster be a Grid node. Therefore, this project built a hierarchical architecture that integrates a Grid environment and a cluster environment. In order to use mpiBLAST in BioGrid, we implemented a Grid service called G-BLAST. Users invocations are performed on G-BLAST, which is built on GT4 and manages the submission, dispatch and monitoring of Grid jobs, and returns results to users via WSRF. We describe G-BLAST in more detail in the next section.

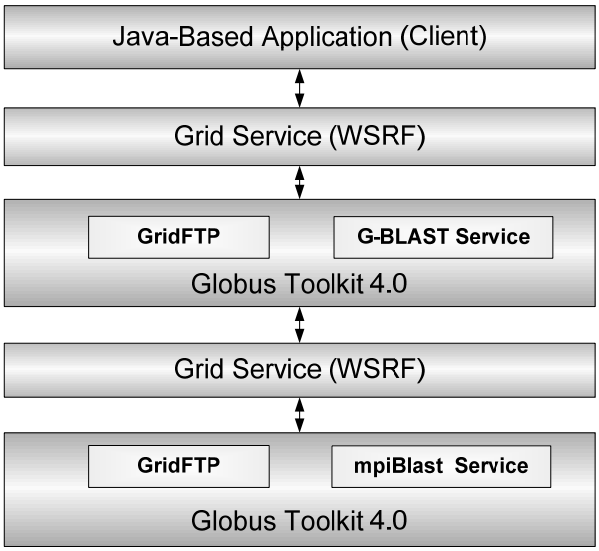

Figure 2. The BioGrid software architecture

4: G-BLAST Grid Service Design

We propose the G-BLAST Service built on the BioGrid framework as shown in Figure 3. This service uses the Java and C++ program languages to implement all components. The User Portal and mpiBlast Service are implemented in Java because it has better support for the Grid services in GT4 than others. The Workflow System, Job Dispatch System, Information System, Segmentation Databases System, Job Monitor System, and Combination Results are implemented in C++. Each mpiBlast Service is built on a master server in a standalone Cluster system, and executes mpiBLAST applications through MPICH-G2.

4.1: User Portal

The User Portal enables interactions with G-BLAST Service. It is divided into two parts that are the user application and the G-BLAST Service portal. They are wrapped in the SOAP format, which then transfers whatever it receives or returns via WSRF.

The first part is a desk client application that enables use of the G-BLAST Service. This portal was developed in Java and provides the same functions as mpiBLAST, but ours is a GUI that allows easy use, just like the conventional mpiBLAST. The portal always returns messages from G-BLAST Service, whether the user invocation was successful or not. If the user's invocation is successful, the portal shows the alignment results, or an appropriate message telling the user what has happened. Moreover, our portal provides security control in that the user must show authorization before using G-BLAST Service.

The second part is a part of G-BLAST, and it receives user invocations and returns results, such as job running time, job running status, and how many jobs running are, from G-BLAST.

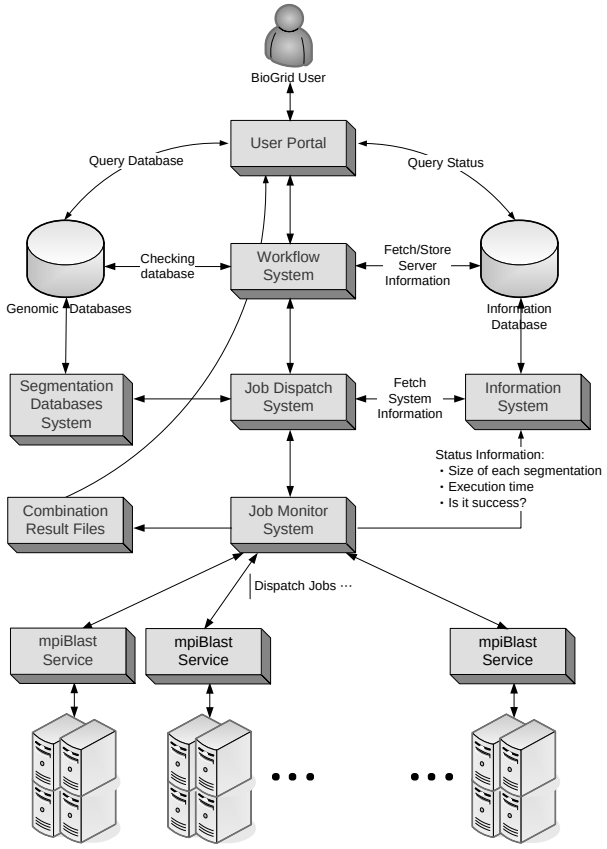

Figure. 3. The G-BLAST Service System architecture

4.2: Workflow System

The Workflow System checks Genomic Databases in the server and monitors the job loading of the server on which the G-BLAST Service is built.

Database checking is necessary because even though the probability is extremely low, the genomic database the client needs may be lost from the server. Thus, we must be sure this database, which is needed by the user, existing. If the checking result is true, then the user's invocation is accepted and the job is delivered to the Job Dispatch System. If the result is false, a message is displayed to notify the user that the invocation was rejected.

Server loading is monitored because there are many user invocations, and we need to know jobs statuses in order to avoid overloading the server. This component records the date, time, and size of the genomic database when G-BLAST is invoked, and records the date and time again after the invocation is accomplished. PostgreSQL stores these records in the Information Database. Since these records tell us the run time of each job and how many jobs are running, we can utilize them to estimate the run time of a new job in advance. Thus, we can also use them to control server loading with this component by deciding which job runs immediately and which job must wait for a while.

4.3: Information System

This component collects the running statuses of each site's jobs from the Job Monitor System. These statuses enable the Job Dispatch System to decide how to dispatch jobs and how to segment the database. When a site's job is accomplished, this component records the size of each segment, job execution time, and whether or not the job was successful.

PostgreSQL is used to store these records in the Information Database. It must be noted that the Workflow System and the Information System use different tables.

4.4: Database Segmentation System

This component is used to segment genomic sequence databases. You perhaps have a question that is why you do not use mpiformatdb command provided by mpiBLAST, and there are two reasons.

The first reason is that the mpiformatdb command uses simple division, putting the sequences in each fragment of the original sequence database in rotation. Since the sequence databases in this project are downloaded from NCBI and their sequences in the database are not stored in order of size, the fragments mpiformatdb yields are also not stored in order of size. "Size" refers to the total letters/length in the sequences. Sequence database lengths and search run times appear to be strongly related in that mpiBLAST uses the Local Alignment algorithm. Consequently, it orders the sequences in each fragment by rotation, but not in sequence length order, so it is difficult to predict execution time and tends to reduce performance. Therefore, this project sorts original databases in sequence length order in advance and then divides them. Fragments divided by this component are dispatched to each site by the Job Dispatch System, and a significant improvement in performance is obtained.

The second reason is that the mpiformatdb command provided by mpiBLAST is not used is that it divides sequence databases into approximately equal size fragments. Since our system has a hierarchical architecture in which each site may have its own computation capacity and network speed, dividing sequence databases into approximately equal size fragments is not a good method for us. It leads to faster sites needing to wait for slower sites to complete their sub jobs, and strongly affects performance. Therefore, we divide sequence databases into different size fragments according to site speed and capacity, thus allowing database sequences to be of appropriate size for each site. This information is sent to the Job Dispatch System. For example, if a job involves two sites, A and B, and site A has twice the computation capacity of site B, but they have the same network speed, we divide the

sequence database into a two-thirds fragment and a one-third fragment, and assign the two-thirds fragment to site A and the other to site B.

4.5: Job Dispatch System

This component fetches and analyzes system information and then utilizes it to decide how to segment databases and how to dispatch jobs from the Workflow System.

The process is divided into the five steps shown below.

- Step 1: Fetch a job from the Workflow System.
- Step 2: Fetch system information from the Information System.
- Step 3: Analyze system information such as the run status of each job from each site. This information is used to decide how many sites are needed by the current job and which sites can run the job.
- Step 4: Tell the Database Segmentation System how to divide the sequence database needed by the job.
- Step 5: Tell the Job Monitoring System which sites are needed to work on the current job.

4.6: Job Monitoring System

This component has both server and client roles, i.e., it belongs to G-BLAST Service and is a Job Monitoring System when it is acting as a server, but it needs to connect to mpiBlast Service at each site to dispatch sub-jobs when it is acting as a client.

It dispatches sub-jobs assigned by the Job Dispatch System and delivers the fragments divided by the Database Segmentation System to each site via GridFTP. Moreover, it monitors the status of each site and tells the Information System to store this information in the Information Database.

4.7: mpiBlast Service

This service is a simple Grid service. It first receives user invocations, which include commands, query sequences, and sequence databases. Second, it executes grid-proxy-init commands provided by GT4. Third, it executes mpiformatdb and mpiblast programs provided by mpiBLAST. Finally, it returns result files to users if the second and third steps are successful.

Because it is a simple Grid service, it connects to any client developed in accordance with WSDL documentation that is the same as the WSDL documentation of the mpiBlast Service. In this project, we put the client application in the Job Monitor System, which dispatches jobs to sites.

4.8: Result File Combination

Because sequence databases are segmented by the Database Segmentation System in advance, many unwanted result files are generated, rather than one complete result file. This component combines all result files belonging to a job into a complete result file and shows it on the user's display via the User Portal.

5: Discussion

Four findings from G-BLAST Service are worth summarizing.

First, G-BLAST segments sequence databases in advance in accordance with the Job Dispatch System and then decides which sites need to run the jobs. Conventional mpiBLAST divides sequence databases into approximately equal fragments using the mpiformatdb command and then assigns the fragments to nodes. However, the sites may have different computation capacities and network speeds, which leads to faster sites needing to wait for slower sites to complete their sub jobs. Overall performance is strongly affected by this situation.

Second, G-BLAST does not use the mpiformatdb command. It just segments the sequence databases, which is faster than using mpiformatdb. G-BLAST lets mpiBlast Service executes mpiformatdb and formats its own fragments. This is just like performing mpiformatdb in parallel, and is faster than conventional mpiBLAST. In addition, execution mpiformatdb by mpiBlast Service can avoid G-BLAST server overloading with segmenting sequence databases.

Third, BioGrid is flexible compared to conventional mpiBLAST. Because each BioGrid site is independent, each can easily adjust its environment or performance. Conventional mpiBLAST needs more steps to make such adjustments.

Fourth, BioGrid mpiBLAST is GUI-operated, rather than command-line-operated, which makes it easier to use. Results are also visible on users' screens allowing one–touch storage.

6: Related Work

In [21] Kandaswamy provided a web service portal in a grid environment and implemented a user interface that enabled BLAST service use. This work discussed web service for scientific Grid applications, but did not propose a solution for very large sequence databases.

In [22] Andrade provided two Perl scripts. The first script manages submission, monitoring, and collection of Grid Jobs, and also splits large files into selected numbers of smaller files. These functions are similar to the G-BLAST Service Job Monitor System and Database Segmentation System. The second script is for parallel execution on Grid nodes. Their research greatly resembles ours, but they split files into selected numbers. In G-BLAST Service, files are split according to the performance of each site. G-BLAST Service splits files based on the ability and speed of each site as determined by the Job Dispatch System.

Moreover, since BLAST is not a parallel system, each Grid node is standalone. Thus, [20] and [21] can use

BLAST but not mpiBLAST, so they cannot flexibly extend the performance of each node as we can.

7: Conclusions

This paper proposes the Grid-based solution BioGrid. It integrates mpiBLAST into a Grid environment. We also report on implementing a G-BLAST Service that allows use of mpiBLAST which is integrated into BioGrid. G-BLAST is faster than conventional Cluster architectures, and it is easier to use than previous mpiBLAST offerings.

Future work will keep improving the Workflow System and Job Dispatch System strategies to enable better G-BLAST performance. We hope future research will provide more detailed results that further differentiate these views.

REFERENCES

[1] Sturn A, Mlecnik B, Pieler R, Rainer J, Truskaller T, Trajanoski Z., "Client-Server Environment for High-Performance Gene Expression Data Analysis," *Bioinformatics*, 2003, 19(6):772-773.

[2] K. Fumikazu, Y. Tomoyuki, F. Akinobu, D. Xavier, S. Kenji, and K. Akihiko, "OBIGrid: A New Computing Platform for Bioinformatics," *Genome Informatics*, 13:484-485, 2002.

[3] S. Gernot, R. Dietmar, and T. Zlatko, "ClusterControl: A Web Interface for Distributing and Monitoring Bioinformatics Applications on a Linux Cluster," *Bioinformatics*, 20(5):805-807, 2004.

[4] R. Buyya, "High Performance cluster Computing: System and Architectures," *Vol. 1, Prentice Hall PTR, NJ*, 1999.

[5] R. Prodan and T. Fahringer, "ZENTURIO: An Experiment Management System for cluster and Grid Computing," *Proceedings of IEEE International Conference on cluster Computing (CLUSTER'02)*, pp. 9-18, Chicago, Illinois, USA, 2002.

[6] F. Achard, G. Vaysseis, and E. Barillot, "XML, bioinformatics and data integration," *Bioinformatics*, vol. 17 no.2 2001 Pages 115-125.

[7] NCBI BLAST, http://130.14.29.110/BLAST/.

[8] mpiBLAST, http://mpiblast.lanl.gov/.

[9] MPI, http://www.lam-mpi.org/.

[10] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *Physics Today*, 55(2):42-47, 2002.

[11] I. Foster and C. Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure (Elsevier Series in Grid Computing), Morgan Kaufmann, 2nd edition," 1999.

[12] C. T. Yang, Y. L. Kuo, K. C. Li, and J. L. Gaudiot, "On Design of Cluster and Grid Computing Environments for Bioinformatics Applications," *Distributed Computing - IWDC 2004: 6th International Workshop, Lecture Notes in Computer Science*, Springer-Verlag, Arunabha Sen, Nabanita Das, Sajal K. Das, *et al.* (Eds.), Kolkata, India, vol. 3326, pp. 82-87, Dec. 27-30, 2004.

[13] R. Nobrega, J. Barbosa, and P. Monteiro, "BioGrid Application Toolkit: a Grid-based Problem Solving Environment Tool for Biomedical Data Analysis," *VECPAR*, 2006

[14] GT4, http://www.globus.org/.

[15] WSRF, http://www.globus.org/wsrf/.

[16] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance* Computing, 11(2):115–128, summer 1997.

[17] N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing (JPDC)*, Vol. 63, No. 5, pp. 551-563, May 2003.

[18] C. T. Yang, Y. L. Kuo and C. L. Lai, "Design and Implementation of a Computational Grid for Bioinformatics," *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE 04)*, pp. 448-451, Grand Hotel, Taipei, Taiwan, March 28-31, 2004.

[19] P. Bala, J. Pytlinski, and M. Nazaruk, "BioGRID – An European Grid for Molecular Biology," *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, 2002, pp. 412.

[20] NCBI, http://www.ncbi.nlm.nih.gov/.

[21] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon, "Building web service for scientific grid applications," *International Business Machines Corporation*, 2006, VOL. 50

[22] J. Andrade, L. Berglund, M. Uhlen, and J. Odebrg, "The use of the Grid technology to solve computationally and data intensive bioinformatics tasks," *Workshop on state-of-the-art in scientific and parallel computing*, 2006