Neural, Parallel, and Scientific Computations 16 (2008) 517-530

Integrated Resource Management Framework for Bio-grid Computing

Lizhe Wang[1] and Wei Jie[2]

[1] Institute for Scientific Computing (IWR), Research Center Karlsruhe (FZK)
Hermann-von-Helmholtz-Platz 1, D-76344 Eggenstein-Leopoldshafen, Germany
[2] The National Centre for e-Social Science,
Arthur Lewis Building, Oxford Road, Manchester, U.K.

Abstract
Computing grid is a promising platform that provides plenty of resource for large scientific computing applications. To achieve the performance of applications in the grid environment, careful resource management is needed. Protein alignment analysis, a typical bio-grid computing application, is a computing intensive, data parallel application, which needs data storage resources and large computing resources. This paper proposes an integrated resource management framework for bio-grid computing. We demonstrate that this kind of applications can benefit from resource management framework.

Keywords - Grid computing, protein alignment analysis, bio-grid computing

1. INTRODUCTION

Now, computing grid has been the most promising computing platform for large scientific applications. Computing grid infrastructure middleware, such as Globus, have been developed to support application users with grid services, e.g. resource management, grid information services. However, on the top of these services, some higher-level supports are still needed to achieve better performance.

Bio-grid computing can benefit greatly from the computing grid [1,2,3]. The application of protein alignment analysis [4,5], a typical bio-grid computing application, is a common and often repeated work in the field of molecular biology. This application consists of finding similarities between a particular query sequence and all sequences in Protein Banks, which maybe large scale geographically distributed. Protein alignment analysis needs huge computing resources to process large amount of data. Thus, to get the best performance, integrated resource management for large computing resources and data storage resources in the Grid environment need to be developed.

Some previous work has focused on this field. SAM (Sequence Alignment and Modeling System) [6, 7] is developed for general sequence analysis. PAPIA (Parallel Protein Information Analysis system)[9] is a system for protein sequence analysis. These systems focus on parallel algorithms of sequence analysis and modeling; give little consideration of underlying parallel architecture support. An application level scheduling framework [9], which is based on AppLeS [10,11,12], is developed for Gene sequence analysis in the metacomputing systems. In this work, a time-balancing heuristic is used [13] and only computing resources are considered. Some other general resource manage systems, e.g., Condor [16,17], PBS [18], Legion [19] do not give convenient support for integrated resource management for this specific type of bio-grid applications.

Received September 8, 2007

In this paper, an integrated resource management framework is discussed. This resource management framework includes two parts: resource selection framework and replica selection framework. Computing resources and data storage resources are managed in the framework to improve the application performance. We improve self-scheduled work queue algorithm [14,15] for task scheduling in the resource selection framework. The replica selection framework is developed for data storage resources management. Since Globus [20] has been the de Facto standard of Grid computing, our work is developed on the platform of Globus Toolkit™ [20] and several popular resource management systems. Results show that bio-grid applications can reach high performance from the framework.

This paper is organized as follows: in section 2, we introduce the background of protein alignment analysis application. Section 3 describes the resource selection framework and section 4 describes the replica selection framework. In section 5, we present the performance of the resource management framework. We conclude our work in section 6.

2. PROTEIN ALIGNMENT ANALYSIS

2.1. Background

Protein alignment analysis is a common and often repeated task in molecular biology. The analysis process consists of finding similarities between a particular query sequence and all the sequences of a Protein Bank. This operation allows biologists to point out sequences sharing common subsequences. From a biological point of view, it leads to the identification of similar functionalities. The need for speeding up this application comes from the exponential growth of the bio-sequence Protein Banks: every year their size scaled by a factor 1.5 to 2 [4].

Surprising relationships have been discovered between protein sequences that have little overall similarity but in which similar subsequences can be found. In that sense, the identification of similar subsequences is probably the most useful and practical method for comparing two sequences. The Smith-Waterman algorithm [4] finds the most similar subsequences of two sequences (the local alignment) by dynamic programming. The algorithm compares two sequences by computing a distance that represents the minimal cost of transforming one segment into another. Two elementary operations are used: substitution and insertion/deletion (also called a gap operation). Through series of such elementary operations, any segments can be transformed into any other segment. The smallest number of operations required to change one segment into another can be used to measure the distance between the segments.

Consider two strings $S1$ and $S2$ of length $l1$ and $l2$. To identify common subsequences, the Smith-Waterman algorithm computes the similarity $H(i,j)$ of two sequences ending at position i and j of the two sequences $S1$ and $S2$. The computation of $H(i,j)$ is given by the following recurrences:

$$H(i,j) = \max \begin{cases} 0 \\ E(i,j) \\ F(i,j) \\ H(i-1,j-1) + Sbt(S1_i, S2_j) \end{cases}$$

$$1 \le i \le l_1, 1 \le j \le l2$$

$$E(i,j) = \max \begin{cases} H(i,j-1) - \alpha \\ E(i,j-1) - \beta \end{cases}$$

$$0 \le i \le l1\ ,\ 1 \le j \le l2$$
$$F(i,j) = \max \begin{cases} H(i-1,j) - \alpha \\ E(i-1,j) - \beta \end{cases}$$
$$1 \le i \le l1\ , 0 \le j \le l2$$

where *Sbt* is a character substitution cost table. Initializations of these values are given by:

$$H(i,0) = E(i,0) = 0 \qquad 0 \le i \le l1$$
$$H(0,j) = F(0,j) = 0 \qquad 0 \le j \le l2$$

Multiple gap costs are taken into account as follows: α is the cost of the first gap; β is the cost of the following gaps. Fig. 1 illustrates an example with gap costs $\alpha = 1$ and $\beta = 1$ and *Sbt* defined as:

$$Sbt(x,y) = \begin{cases} 2 & x = y \\ -1 & x \ne y \end{cases}$$

Each position of the matrix *H* is a similarity value. The two segments of *S*1 and *S*2 producing this value can be determined by a backtracking procedure (see Fig. 1).

	Ø	A	T	C	T	C	G	T	A	T	G	A	T	G
Ø	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	1	0	0	2	1	0	2
T	0	0	2	1	2	1	1	4	3	2	1	1	3	2
C	0	0	1	4	3	4	3	3	3	2	1	0	2	2
T	0	0	2	3	6	5	4	5	4	5	4	3	2	1
A	0	2	2	2	5	5	4	4	7	6	5	6	5	4
T	0	1	4	3	4	4	4	6	5	9	8	7	8	7
C	0	0	3	6	5	6	5	5	5	8	8	7	7	7
A	0	2	2	5	5	5	5	4	7	7	7	10	9	8
C	0	1	1	4	4	7	6	5	6	6	6	9	9	8

Fig. 1: Example of the Smith-Waterman algorithm to compute the local alignment between two DNA sequences ATCTCGTATGATG and GTCTATCAC. The matrix $H(i,j)$ is shown for the computation with gap costs $\alpha = 1$ and $\beta = 1$, and a substitution cost of +2 if the characters are identical and -1 otherwise. From the highest score (+10 in the example), a trace back procedure delivers the corresponding alignment (shaded cells), the two subsequences TCGTATGA and TCTATCA.

2.2 Master-Slave Paradigm

This application is based on SPMD concept. Thus, the application is modeled with the master-slave paradigm. The master is responsible for dividing the tasks and scheduling the tasks and slaves execute the tasks (see figure 2).

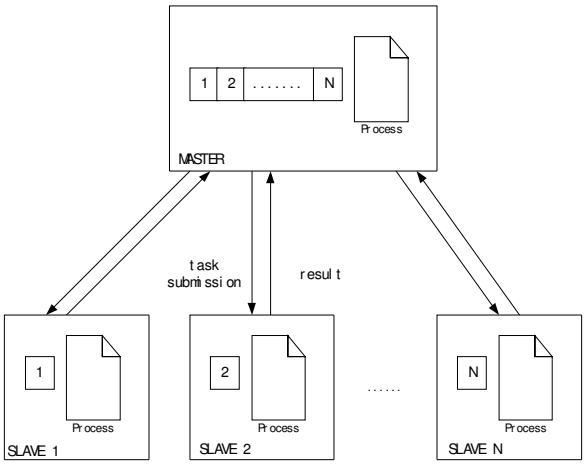

Fig. 2: Master-slave paradigm

Considering the tasks submitted to the slave, slave selects the nearest proper Protein Banks and make a copy of part or total protein alignment paradigms of the Protein Banks (see figure 3). These copies of protein alignment paradigms are used for protein alignment comparison. When the task is finished, the results are sent back to the master.

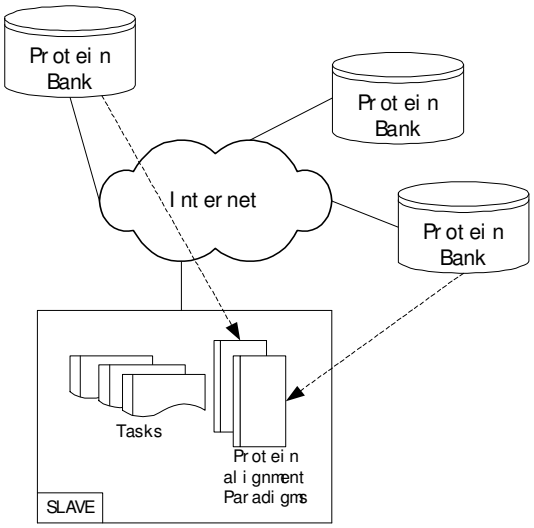

Figure 3 Replication Selection of Slave

3. RESOURCE SELECTION FRAMWORK

3. 1 Overview

The resource selection framework is designed with its server on the master node and clients on the slave nodes. Resources are managed and jobs are submitted from the server. The system configuration is shown in Figure 4.

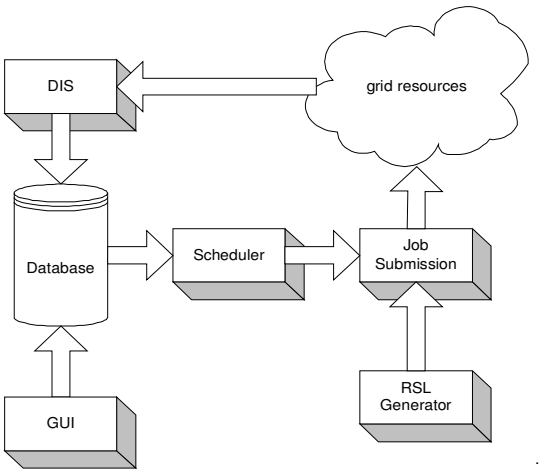

Figure 4 Resource Selection Framework

Distributed Information System (DIS) maintains the static and dynamic information of computing grid resources. Scheduler manages and schedules the resources. After RSL generator generates the RSL files, jobs can be submitted to the computing grid. Master-slave paradigm is used to schedule resources and submit jobs (Figure 5). With GUI, users can submit jobs from one host to the computing grid. The submission host is master node and other nodes in the computing grid are slave nodes.

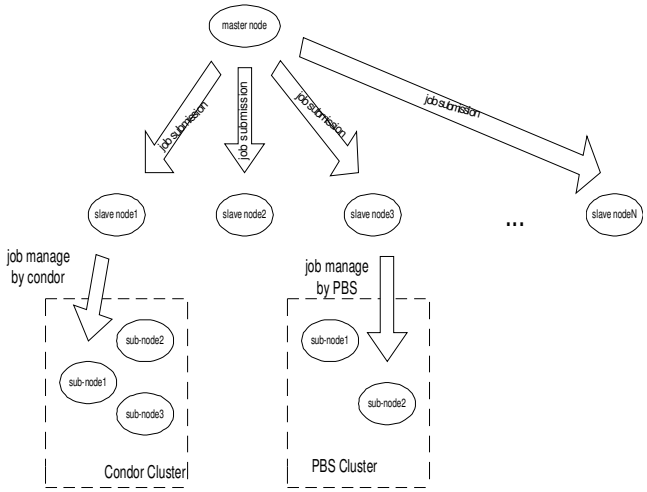

Figure 5 Job Submission

3.2 Distributed Information System

In the system, there is a daemon running on each slave node. The daemon will maintain the static information (e.g., CPU speed, CPU count, memory) and the dynamic information (e.g., available CPU percent and available memory) of slave nodes. The maser node will send request to slave nodes and get system information periodically. So, the master node can maintain a global system information table (Figure 6).

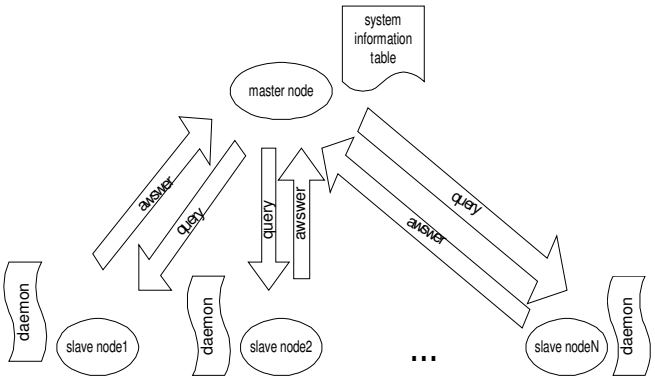

Figure 6 Distributed information system

With Globus API [22], we can get the information of slave nodes. Fig6 is an example of our system output (see Figure 7).

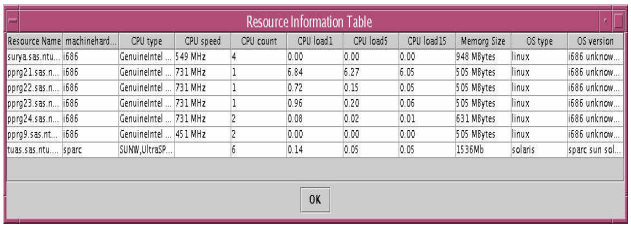

Figure 7 Output of system information

3.3 Scheduler

Computing grid is a hierarchical, distributed and shared system. The static information and run-time system information are both important for scheduling the resources [15]. We firstly quantify the static computing capacity by selecting a protein alignment whose size is $Length$ as the benchmark. On node j, the static computing capacity is:

$$C_j = \frac{Length}{T_j} \tag{1}$$

where T_j is the time to finish the job of processing protein alignment benchmark and C_j is the computing capacity of the node j (measured protein alignment length per hundred seconds). It should be noticed that this parameter is application relevant. So, it can be calculated before job submission.

DIS can be used to obtain the following run-time information of the system: $ACPU_j$ (available CPU percent of node j). The run-time computing capacity of node j can then be defined as:

$$RC_j = C_j * ACPU_j \tag{2}$$

It is a metric that measures the current fulfillment of application's computing requirement by node j per unit time. Thus, for time period T, the work can be done by node j is:

$$W_j = RC_j * T \tag{3}$$

Self-scheduled work queue algorithm is a popular algorithm for scheduling sets of independent tasks. The algorithm assigns jobs to hosts as soon as they become available in a greedy fashion [11,15]. Here, an improved self-scheduled work queue algorithm is used. New algorithm does not use a parameter to decide whether the hosts are free (e.g. when the available CPU percent is 30%, the node is defined to be free). Improved self-scheduled work queue algorithm just allocates the jobs based on the dynamic CPU load. Although the host is busy and not free (e.g., available CPU percent is 10%), new algorithm still allocate some small size jobs to this host to occupy the available CPU percent. The scheduling process is shown as Figure 8.

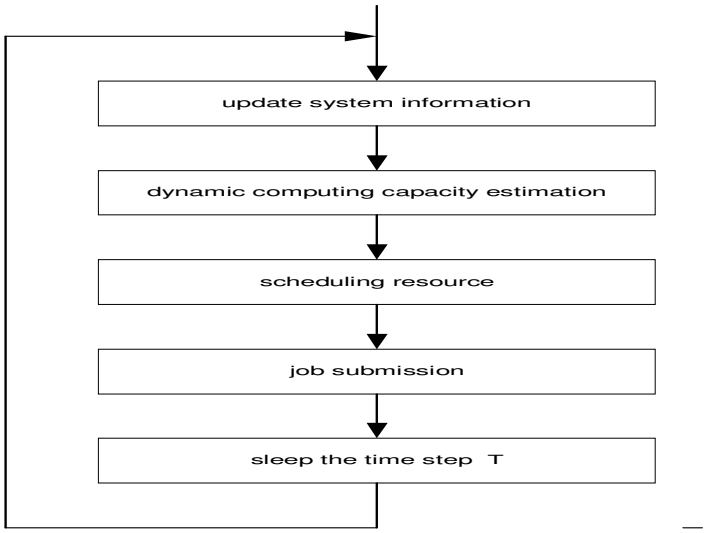

Figure 8 Schedule process

The platform updates the dynamic information periodically and schedule jobs. It estimates the run-time computing capacity and submits jobs to each slave nodes. The period of job submission is T. In the protein alignment analysis application, each job generated has a parameter that indicates the size of the protein data to be searched. The scheduler will make use of formula (3) to calculate W_j and translates it to the parameter of the job to be generated for node j.

3.4 Failure Recovery Scheme

Failure recovery scheme is developed to address the problem that application programs hang up and waist the CPU cycles of computing resources. The system failure of resources can be detected by DIS.

Time out scheme is used here. After the task is submitted, if the limit of response time reaches and there is no result back, the task in the slave node then is killed by the master and re-submitted to other slave node.

The limit of response time is determined as follows:

$$Limit_{ij} = (1.5 \sim 2) \times ERT_{ij} \tag{4}$$

where,

$Limit_{ij}$ is limit of response time of task i on slave j,

ERT_{ij} is expected response time of task i on slave j.

Expected response time is determined as follows:

$$ERT_{ij} = \frac{Task_i}{RC_j} \tag{5}$$

where,

$Task_i$ is the size of task i that running on node j (in term of protein alignment length),

RC_j is the run-time computing capacity of node j, which is given in Equation (2).

4. Replica Selection Framework

After the task is submitted to the slave node, the slave node makes a replica selection for running the task. Different types of task may require different data sets (protein alignment paradigms) for protein alignment analysis. These data sets are stored in different Protein Banks, which locate in geographically distributed sites.

An LDAP [23, 24] server is configured to present replica directory service. Replica catalog supports to register the Protein Banks as logical collections of data sets and provides mappings between data sets and Protein banks (see also figure 9).

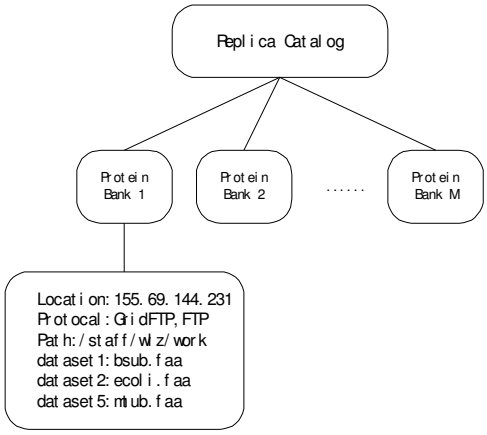

Figure 9 Replica catalog

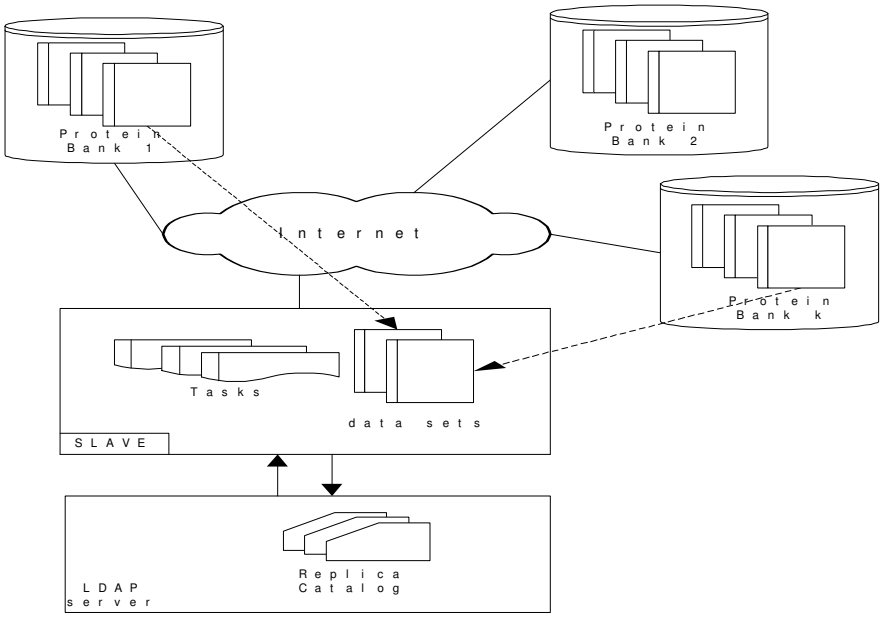

Figure 10 Replica Selection Framework

The replica selection in the slave node follows these steps (see also figure 10):

(i) Get information of tasks submitted to the slave node and determine the data sets needed for tasks execution.

(ii) Query LDAP server to locate the proper Protein Banks that can present the data sets needed.

(iii) From every proper Protein Bank to the slave node, the peer-to-peer communication performance is tested.

(iv) The slave node selects the Protein Banks that give minimum data transfer time as the replica source.

5. PERFORMANCE EVALUATION

5.1 Test bed

Table 1. Test bed Configuration

Location	Resource		Operating System
NTU, Singapore	Computing Resource (Cluster)	Intel PIII-733 MHz *12	Condor / Linux
		Intel PII-450 MHz *6	PBS / Linux
		SUNW,UltraSPARC-II *10	Sun Grid Engine / Solaris 2.6
	LDAP server	Intel PIII-733 MHz	Linux
	Protein Bank	Intel PIII-733 MHz	Linux
IHPC, Singapre	Computing Resource (Cluster)	IBM xSeries 330, PIII 933 MHz *13	Condor / Linux
	Protein Bank	IBM xSeries 330, PIII 933 MHz	Linux
Osaka Univ., Japan	Computing Resource (Single Node)	Intel PII-450MHz*1	Fork / Free BSD

Table2 Experiments Results

Index	Protein Alignment Length	Execution Time(s)		
		Static scheduling algorithm (T1)	Improved self-scheduled work queue algorithm (T2)	Performance Efficiency Enhancement (1-T2/T1)
(a)	925	3071	2563	16.5%
(b)	2821	8911	7224	18.9%
(c)	4289	12973	10227	21.2%

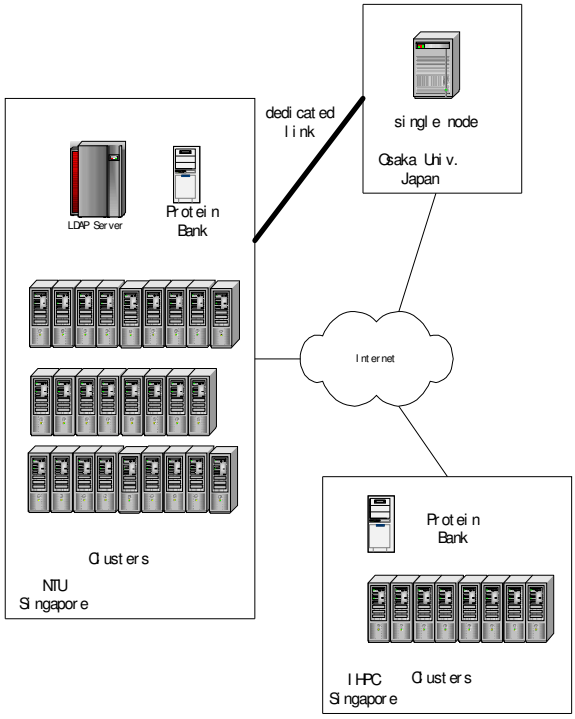

Figure 11 Test bed

In order to evaluate the performance of the integrated resource management framework, a test bed is configured (see also Figure 11). It includes 3 clusters in NTU (Nanyang Technological Univ., Singapore), one cluster in IHPC (Institute of High performance Computing, Singapore) and a single node in Osaka University (Japan). A LDAP is configured in NTU for replica service. Two Protein Bans are set inside NTU and IHPC separately. Table 1 shows the hardware and software configuration of the test bed. Globus Toolkit™ is installed on the head nodes of clusters and the single node in Japan. Globus GRAM will talk with the local job manager (e.g., Condor) about how to execute the jobs submitted by Globus GRAM.

5.2 Performance Evaluation

Three experiments were conducted with different problem sizes. We compare the execution time of static scheduling algorithm and improved algorithm (Figure 12). Table 2 presents the experiment results in detail.

The results show that the performance efficiency increases as the problem size grows. In general, we can see that the resource management framework can help this application achieve better performance.

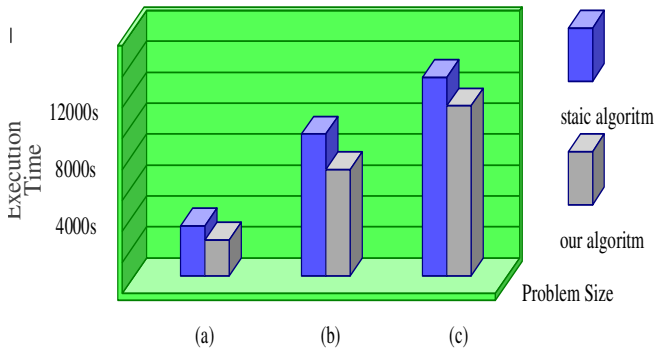

Figure 12 Execution Times

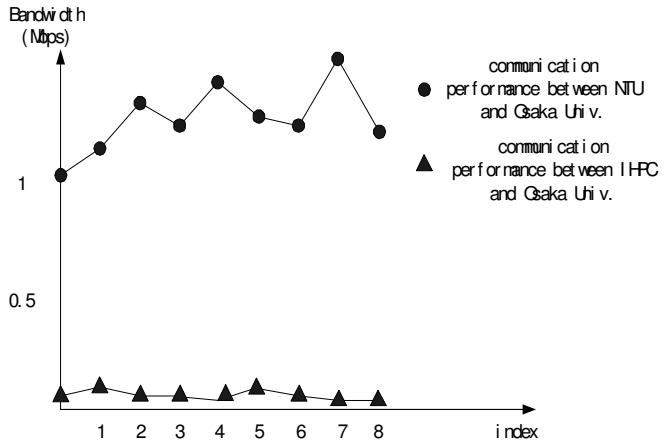

Figure 13 Communication Performance

Table 3 shows the replica placement for the Protein Bank. Now, single node in the Osaka Univ., as an example, is the slave node and replica selection result is showed below. Table 5 shows the replica selection for different replica requirement of tasks running on the slave node.

Table 3 Replica placement

Protein Bank	Site	Replica placement
1	NTU	dataset1, dataset2, dataset5
2	IHPC	dataset1, dataset3, dataset4

6. CONCLUSION AND FUTURE WORK

The objective of resource management framework is to help bio-information applications achieve better performance in the grid environment. In the framework, we improve self-scheduled work queue algorithm to schedule the resources for these applications. Experiment results show that applications can achieve better performance than using static scheduling algorithm. On the other hand, replica selection framework helps slave nodes to make replica selection dynamically. Thus, better performance is also reached.

Table 4 Replica Requirement and Selection

Task index	Replica requirement	Replica decision
1	dataset1	Protein Bank 1
2	dataset2	Protein Bank 1
3	dataset2	Protein Bank 1
4	dataset3	Protein Bank 2
5	dataset5	Protein Bank 1
6	dataset1	Protein Bank 1
7	dataset2	Protein Bank 1
8	dataset4	Protein Bank 2

Our future work is to include performance prediction for the computing resources and communication network. Thus, with the prediction information, we expect to improve the performance of the applications.

REFERENCES

[1] BioGRID project in EUROGRID project, http://biogrid.icm.edu.pl/, 2002

[2] BioGRID Computing Symposium 2001, http://www.bic.nus.edu.sg/biogrid/biogrid01/ , 2001

[3] BioGRID project in Japan, http://www.biogrid.jp/, 2002

[4] Bertil Schmidt, Heiko Schroder and Thambipillai Srikanthan, A SIMD Solution to iosequence Database Scanning, Proceedings PaCT'2001, Lecture Notes in Computer Science 2127, Springer 2001, pp. 498-509.

[5] Lizhe Wang, Wentong Cai, et. al. Bio-grid Computing Platform: Parallel Computing for Protein Alignment Analysis, to appear 6th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia), 2002.

[6] Alignment and Modeling System, http://www.cse.ucsc.edu/research/compbio/sam.html, 2002.

[7] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler, Hidden Markov models in computational biology: Applications to protein modeling. Journal of Molecular Biology, 235:1501--1531, February 1994.

[8] Parallel Protein Information Analysis system, http://www.cbrc.jp/papia/papia.html, 2002.

[9] Neil Spring and Rich Wolski. Application Level Scheduling of Gene Sequence Comparison on Metacomputers. 12th ACM International Conference on Supercomputing, Melbourne, Australia, July 1998.

[10] F. Berman and Rich Wolski, R., Scheduling from the Perspective of the Application. In Proceedings of the 5th International Symposium on High-Performance Distributed Computing (HPDC-5), pages 100-111, August 1996.

[11] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid. In Proceedings of the Supercomputing Conference (SC'2000), 2000.
[12] F. Berman and R. Wolski. The AppLeS Project: A Status Re- port. In Proceedings of the 8th NEC Research Symposium, Berlin, Germany, May 1997.
[13] F. Berman, R. Wolski, et.al., Application Level Scheduling on Distributed Heterogeneous Networks. In proceedings of Supercomputing, 1996.
[14] Henri Casanova, Graziano Obertelli, F. Berman, and R. Wolski, The AppLeS Parameter Sweep Template User-level Middleware for the Grid, Proceedings of IEEE Supercomputing 2000,4-10 November 2000, Dallas, Texas, USA
[15] T. Hagerup, Allocating Independent Tasks to Parallel Processors: An Experimental Study, Journal of Parallel and Distributed Computing 47(1997), 185-197
[16] M. Litzkow, M. Livny, and M. W. Mutka. Condor – A Hunter of Idle Workstations. In Proceedings of 8th International Conference of Distributed Computing Systems, June 1988.
[17] Condor Project, http://www.cs.wisc.edu/condor/
[18] PBS Project, http://www.openpbs.org/
[19] Legion Project, http://legion.virginia.edu/
[20] Globus Project, http://www.globus.org
[21] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid Information Services for Distributed Resource Sharing. In Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
[22] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids. ACM 2000 Java Grande Conference, 2000.
[23] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, B. Tierney. File and Object Replication in Data Grids Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
[24] T. A. Howes and M. C. Smith. LDAP Programming Directory-Enabled Application with Lightweight Directory Access Protocol. Technology Series. MacMillan, 1997.