

# Chameleon: A Resource Scheduler in A Data Grid Environment\*

Sang-Min Park

Jai-Hoon Kim

Graduate School of Information and Communication  
Ajou University, South Korea  
{sm@dmc.ajou.ac.kr, jaikim@ajou.ac.kr}

## Abstract

*Grid computing is moving into two ways. The Computational Grid focuses on reducing execution time of applications that require a great number of computer processing cycles. The Data Grid provides the way to solve large scale data management problems. Data intensive applications such as High Energy Physics and Bioinformatics require both Computational and Data Grid features. Job scheduling in Grid has been mostly discussed from the perspective of computational Grid. However, scheduling on Data Grid is just a recent focus of Grid computing activities. In Data Grid environment, effective scheduling mechanism considering both computational and data storage resources must be provided for large scale data intensive applications.*

*In this paper, we describe new scheduling model that considers both amount of computational resources and data availability in Data Grid environment. We implemented a scheduler, called **Chameleon**, based on the proposed application scheduling model. Chameleon shows performance improvements in data intensive applications that require both large number of processors and data replication mechanisms. The results achieved from Chameleon are presented.*

## 1. Introduction

A Grid is large scale resource sharing and problem solving mechanism in virtual organizations [11]. We can consider Grid service as two parts, Computational and Data Grids. A large number of computers are linked globally to form a computational Grid [10]. A transparent

access to Computational Grid is provided so that users can achieve better performance, especially in terms of speed and throughput. A Data Grid provides two basic services, a secure, reliable, and efficient data transport protocol and replica management [8].

In high energy physics, bioinformatics, and earth observation applications, we encounter huge amounts of data. By its nature, we can achieve better performance (access time) by replicating data in geographically distributed data stores. Computationally intensive data analysis is also required. We can achieve performance improvement by using Computational Grid that provides large number of processors and Data Grid that provides efficient data transport protocol and data replication mechanisms. In such environments, effective resource scheduling is a challenging research issue. One must consider not only abundance of computational resources but data locations. A site that has enough available processors may not be the optimal choice for computation if it doesn't have required data. The processors might wait for a long time to bring the remote data. Similarly, a site with required data may not be a good place to compute if it doesn't have enough computational resources.

In this paper, we propose a new scheduling model that considers both replicated data locations and maximizing processor utilizations to solve mismatch problems for the scheduling. We describe system factors that should be considered for executing applications in Grid: dynamic system factors and application specific factors. We present scheduling scenarios that may occur in real situations with these factors. Then we propose cost analysis models for each scenario. We implemented a scheduler, called **Chameleon**, based on the proposed scheduling models. Experiments were conducted by applying Chameleon on a Grid testbed. Experimental results show that Chameleon is competitive approach for resource scheduling in Grid.

---

\* This study was supported by a grant of the International Mobile Telecommunications 2000 R&D Project, Ministry of Information & Communication, Republic of Korea.

## 2. Related Work

Large data collections are emerging as important resources in many applications. In some area, the volume of interesting data is already measured in terabytes and will be soon petabytes. For an example, High-energy physics experiments in CERN are expected to generate petabytes of scientific data by 2005 [15]. The analysis of data from the experiments is computationally intensive. The integrating architecture motivated by these considerations is a Data Grid [8]. The Data Grid community tries to solve emerged problems by developing replica management services and efficient data transmission protocols. GridFTP is a secure, reliable and efficient data transport protocol developed as a part of the Globus project [3]. The Globus project also provides a basis of replica management, called Replica Catalog, which contains mapping information from a logical file or collection to one or more physical instances of the objects [8]. Stockinger et al. describe basic cost models for analyzing replicated and geographically distributed data in reference [19]. They assume that the Grid sites have both data stores for replicated data and computational capabilities to analyze the data. They suggest the cost model in two ways, data-migration and application-migration. We also assume the Grid environments with similar view to theirs and conduct real experiments based on the assumptions. There have been many efforts for job scheduling on Grid. One representative approach is the application-level scheduling. In AppLes project [5], the performance model for a specific application is provided by the user. Each Grid application is scheduled by its own AppLes and customized to fulfill its user-derived performance model. Our scheduling model presented in this paper is also a kind of application-level scheduling, that is specific kinds of application are discussed in our model. Kavitha Ranganathan et al. describe scheduling considerations in Data Grid environment in [16]. They describe that the choice of scheduling algorithm has a significant impact on a system and argue that they must explicitly address both job scheduling and data scheduling. Their simulation results show that job scheduling considering data location shows better performance. We have the same opinion as theirs from a consequence of real experiments.

## 3. Scheduling Model

We constructed a scheduling model for Data Grid environment. The Data Grid consists of distributed sites that have both data stores and computational capabilities. Figure 1 shows the Data Grid model we assume. This

model takes a similar form to the one CERN presents [15]. Input data sets for applications are replicated among geographically distributed sites and each data has logically unique name. All sites have different computational capabilities and different data stores. Replica Management Server provides mapping function between logical name and physical data copies. In such environments, we should consider some system factors that affect scheduling decision. Factors and the cost models for resource scheduling are discussed in the following subsections.

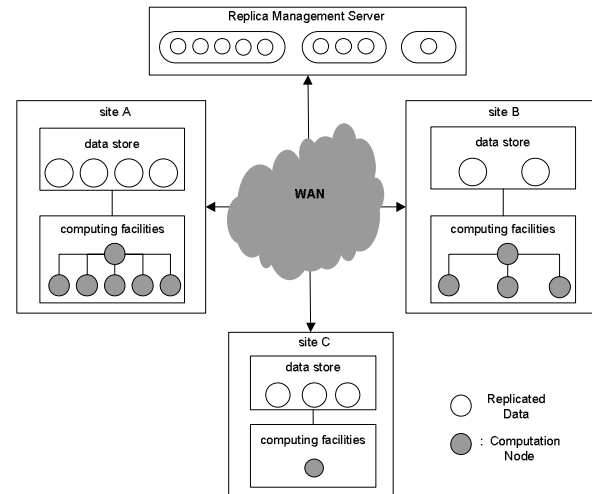


Figure 1. Data grid environment

### 3.1 System Factors

We can distinguish system factors between dynamic factors and application specific factors. While dynamic system factors change over time, the application specific factors are determined before the applications are executed.

#### 3.1.1 Dynamic System Factors

Since dynamic system factors change over time, it is difficult to measure accurately. There are three dynamic factors as follows:

- **Network bandwidth:** Data transfer time is in proportion to the network bandwidth. Since there may be frequent data transfers between sites and nodes, network bandwidth is one of the most important factors. We should measure and forecast the network bandwidth accurately because data transfer time is tightly dependent on current network load and the size of data in Data Grid environment is very large. NWS [21] is a good approach to measure and forecast the network bandwidth. Both WAN bandwidth between sites and LAN bandwidth between nodes in a site should be

measured. In general, WAN bandwidth is more critical because most of data transfer time is consumed on WAN.

- **Number of available nodes:** At each site, the number of available nodes changes over time. In a computational Grid environment, the number of available nodes is one of the most important factors for scheduling. The number of available nodes at a site will be decided according to the job load and amount of processors allocated for the users. If we do not consider data availability at each site, the resource selection will be mostly based on this factor. As the number of node which performs computation grows in a site, there may be more data transfers between nodes.
- **System attributes at each site:** We cannot predict job execution time accurately if we only consider the number of available nodes at each site. Each site has different system architectures. Some sites are based on high performance parallel machines like MPPs and others consist of clusters. Besides different static architectures, each site has different system attributes that change dynamically. For example, processor speed, memory size, and I/O performance can change over time and they are different from each other site. Sometimes an application may especially require large memory size, good I/O performance, etc. We must consider static architectures and changing system attributes at each site.

### 3.1.2 Application Specific Factors

Since various applications such as high energy physics and bioinformatics execute in Grid environment, we must distinguish unique factor that each applications has. Most application-level scheduling models are based on the unique characteristic of each application. Followings are the application specific factors that we consider in our scheduling models.

- **Size of input data:** In general, data intensive applications analyze large amounts of data. In Data Grid environment, data is replicated in geographically distributed sites. It means that if data is not replicated in the selected site for the job, the data should be fetched from other sites. Data fetch from other sites will degrade overall performance because data transfer speed is limited by network bandwidth.
- **Size of application code:** We can consider two methods when local site does not have replicated data. We assume that application codes are always on the local site. First thing is fetching replicated data from remote sites and processing it on local processors. Second one is moving local application codes to remote site in which data replica exists and processing data on that site with migrated application codes. If we choose the

second case, then we must consider size of application codes. In general, the size of application codes will not decrease the overall performance significantly because it is much smaller than input data to process and output data.

- **Size of produced output data:** If computing job takes place at a remote site, the produced output data should be transmitted to the local site so that users can take result. Each data intensive application produces different amount of data. For example, one of the gene sequence comparison applications, FASTA, produces about 5 times larger output than the similar application, BLAST, when the same amount of input protein sequences is processed. Since the amount of output data is proportional to the input data in many cases, we can expect the amount of output data before execution.

## 3.2 Cost Models

In this section, we propose cost models for job execution in Data Grid environment where the input data is replicated in distributed sites. The cost is the response time upon user's job request. We can separate the response time into two parts, data transmission time and job execution time. We make an assumption that the data is read-only. This is reasonable assumption as it is discussed in several Data Grid scenarios [15]. We also assume that application code is in local since the user in local has application codes which will be used while distributed sites cannot have all kinds of data intensive applications. We describe five scenarios that may occur when the job request is scheduled. System parameters in our models are as follows:

- $Time_i$ : The response time of user's job request in scenario  $i$
- $N_i$ : The number of available nodes in site  $i$
- $D_{input}$ : Size of input data
- $D_{app}$ : Size of application codes
- $D_{output}$ : Size of produced output data
- $BW_{LAN(i)}$ : Bandwidth of LAN connection in site  $i$
- $BW_{WAN(i-j)}$ : Bandwidth of WAN connection between site  $i$  and  $j$
- $Exec_i$ : Expected execution time on site  $i$

Figure 2 describes data transfers in our models. The bold arrows with subsection numbers represent the input data transfer in the models discussed in the following subsections.

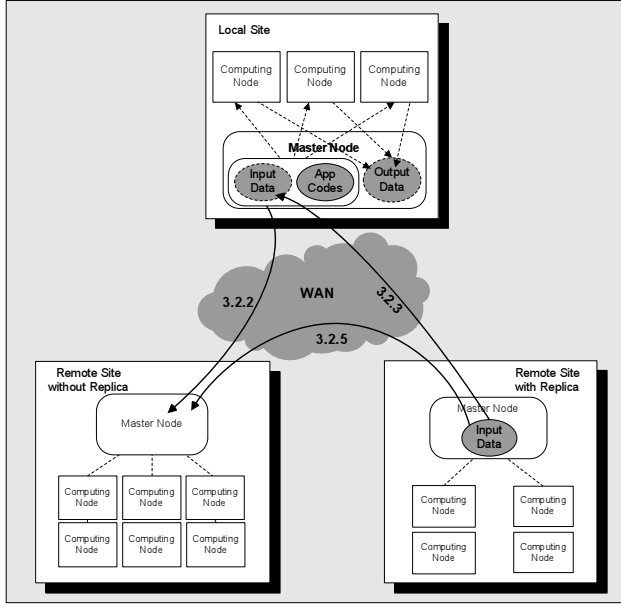


Figure 2. Data movement in models

### 3.2.1 Local Data and Local Execution

$$Time_1 = \frac{N_{local} \times (D_{input} + D_{app}) + D_{output}}{BW_{LAN} (local)} + Exec_{local}$$

The input data is located in local site where processing is performed. The formula is for measuring response time in this scenario. We assume that there exists a master node in a site for delivering the job request to computing nodes that perform process. In the scenario, cost consists of data transmission time between master and computing nodes on a local LAN and estimated job execution time on local available processors. The input data and application codes maintained by master node should be copied to every computing node and produced output data will be aggregated to the master node, as illustrated in upper box in Figure 2. We assume that the input data should be replicated to all computing nodes. It's because the input data can be a form of database for some applications. In that case, the application tries to search all contents of its input data (database), thus whole input data should be copied to all computing nodes. We know that sometimes the input data can be divided into small parts and distributed to the computing nodes. But we choose the worst case because the data transfer time via LAN is very small compared to job execution time or data transfer time via WAN. This assumption of data transfer via LAN is applied to all scenarios regardless of location of the site where the job is executed (local or remote)

### 3.2.2 Local Data and Remote Execution

$$Time_2 = \frac{D_{input} + D_{app} + D_{output}}{BW_{WAN} (local-remote_i)} + \frac{N_{remote_i} \times (D_{input} + D_{app}) + D_{output}}{BW_{LAN} (remote_i)} + Exec_{remote_i}$$

Although local site has replicated data copy, the response time can be reduced when the job is executed in remote site with enough computing nodes like the lower left box in Figure 2. When a remote site has much more available processors for job execution, the input data should be transmitted from local to the selected remote site for execution. The input data and application codes are transmitted to the remote site through WAN, then delivered to all computing nodes through LAN. On the remote site, the input data is processed on computing nodes and result will be aggregated to master node. Finally, the result data should be transmitted to the local site.

### 3.2.3 Remote Data and Local Execution

$$Time_3 = \frac{D_{input}}{BW_{WAN} (local-remote_i)} + \frac{N_{local} \times (D_{input} + D_{app}) + D_{output}}{BW_{LAN} (local)} + Exec_{local}$$

When the input data is not located in local site, we can bring it from remote site having the data and process it using local processors. In this scenario, total data transfer time consists of fetching input data from remote site through WAN and exchanging data set between master and computing nodes via LAN in local site. Total response time is the sum of data transfer time and job execution time in local. When the size of application code and produced output data is excessively large, it is better to select this scenario because transfer of application code and output data through WAN is not required.

### 3.2.4 Remote Data and Same Remote Execution

$$Time_4 = \frac{D_{app} + D_{output}}{BW_{WAN} (local-remote_i)} + \frac{N_{remote_i} \times (D_{input} + D_{app}) + D_{output}}{BW_{LAN} (remote_i)} + Exec_{remote_i}$$

If the remote site has both required input data and enough number of available processors, then the input data can be processed in that site with its processors. In this scenario, application codes in local site are transmitted to the remote site in which input data is located and the job is executed in that site. The produced output data should be returned to local site.

### 3.2.5 Remote Data and Different Remote Execution

$$Time_s = \frac{D_{input}}{BW_{WAN (remote\_i-remote\_j)}} + \frac{D_{app}+D_{output}}{BW_{WAN (local-remote\_j)}} + \frac{N_{remote\_j} \times (D_{input}+D_{app})+D_{output}}{BW_{LAN (remote\_j)}} + Exec_{remote\_j}$$

In general, input data for processing is replicated on the part of sites. While certain sites may have a replicated copy, others may not have it. Sometimes, computational capabilities of a remote site without replicated data are much superior to the capabilities of other sites with replicated data. In such cases, we can reduce response time by moving input data from one site to another site that has a large number of processors. In this scenario, input data located in site  $i$  is transmitted to site  $j$  which has enough computational capabilities. Also application codes should be transmitted from local to site  $j$ . Then processing is performed in site  $j$  and result data will be transmitted to local site.

The scheduler operated on local site compares the performance from the above scenarios and chooses the best one that may takes the least response time. After deciding the best locations for job execution and data fetching, the job and data movement requests can be submitted.

## 4. Scheduler Implementation and Results

We implemented Grid scheduler: **Chameleon**. Chameleon implementation is based on the model discussed in Section 3. We also performed experiments on a Grid testbed. In this section, we describe features of Chameleon, and introduce a testbed and applications that we use. Lastly, we present the experimental results.

### 4.1 Chameleon

Chameleon is built on top of Globus [13]. NWS (Network Weather Service) [21] is also used for measuring and forecasting a network bandwidth. Chameleon makes use of Globus services, such as resource allocation, information, and Data Grid services. NWS provides information of dynamically changing network bandwidth. Chameleon gathers information such as expected size of output data and expected execution time from users when the job is performed on a specific machine (e.g., The job may takes 2 hours on a machine with P3 1 GHz CPU and 512 MB memory). We assume that users can accurately expect the job execution time and the size of result data. This can be a valid assumption because users know the size of input data to be processed and execution time and the size of output data are related to the size of input data in many applications. If the user knows well about the

behavior of the application, the prediction can be done precisely. Chameleon predicts job execution time on each site and data movement time between sites to estimate overall response time. Job execution time on each site can be calculated by comparing each site's system information (CPU, memory, etc.) with the assumed machine specifications. In other words, if users expect that the job may takes 2 hours on a machine with P3 1 GHz CPU, we can guess that it takes 1 hour when it is performed on a machine with 2 GHz CPU, although it is extremely simplified. Data transfer time is predicted by getting network bandwidth between sites using NWS. After deciding the best locations for job execution and data fetch, Chameleon submits a job request based on the job specifications received from users. It also requests data transfer when required. More detailed chameleon features are as follows:

- **Information Monitor:** This component monitors system factors that we describe in subsection 3.1. MDS in Globus and NWS are used for gathering needed information.
- **Location Finder:** This searches physical locations of logical data to be used. We use Globus Replica Services.
- **Scheduler:** This makes scheduling decision by analyzing gathered information. The scheduling algorithms are based on the model discussed in subsection 3.2.
- **Data Mover:** After the scheduler component makes a decision, Data Mover receives locations for job execution and data fetch. Then it transmits the required data between sites or nodes. Currently, GridFTP is used for transmitting data.
- **Runner:** This receives job specification from the user and executes the job at the selected site. GRAM service in Globus is used for executing job on a remote site having local resource manager.

### 4.2 Testbed and Applications

We perform experiments on a Grid testbed in S. Korea, named ANTZ [1], and ApGrid testbed [2]. The experiments were conducted on 9 Grid sites. Most sites are based on PC Clusters. LSF(Load Sharing Facility), PBS(Portable Batch System) and Sun Grid Engine are used as local schedulers.

We execute two gene sequence comparison applications, FASTA and PSI-BLAST [12], for our experiments. Many of gene sequence comparison algorithms such as FASTA, BLAST, and Smith-Waterman need computationally intensive analysis on the large size database [12]. Scientists try to predict structures and functions of newly found protein by comparing its sequence with well known

protein sequences in databases. The scheduler in local distributes the protein sequences to the selected Grid site and the analysis is performed on the site with available processors. On each processor, the application compares its allocated protein sequences with the sequences of well known protein in database. In order to execute those sequence comparison jobs efficiently, the database is distributed on many sites. Some sites may have the sequence database since the analysis job have taken place at that site previously and other sites may not have it. Thus data replication mechanism for managing distributed databases and efficient data transfer protocol for transmitting large size database should be applied to these kinds of applications. In other words, Data Grid services can fulfill the application's requirements.

### 4.3 Experimental Results

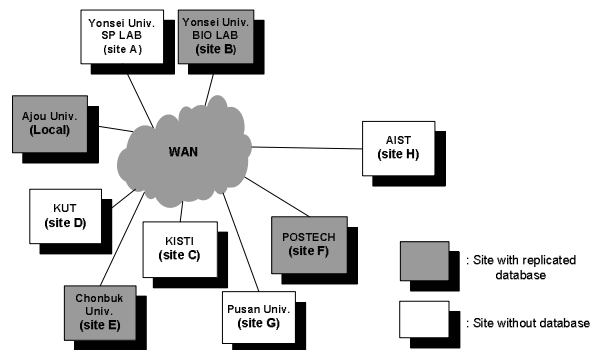
We compared response time of applications between performed by Chameleon and by manual execution. Chameleon predicts the response time for each site and submits job to the selected best site. Parameters are presented in Table 1. All parameters are obtained from the actual applications (PSI-Blast and FASTA). As we mentioned earlier, we assume that users can accurately predict the job execution time and size of result data by considering the size of input data. We executed the applications on a base machine (assumed machine specification) and obtained the execution time and size of result data prior to the real experiments on the Grid sites. We applied the result values to Chameleon as parameters.

**Table 1. Parameters**

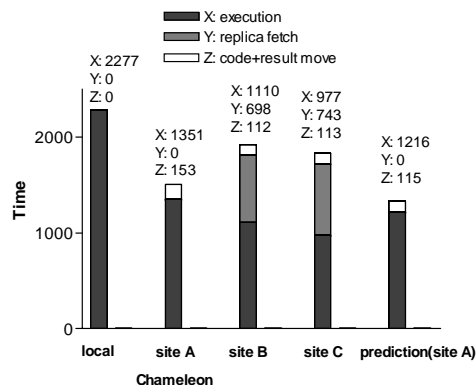
Parameters	PSI-BLAST	FASTA
Size of Input Data (Protein Database)	502 MB	502 MB
Size of Output Data	10 MB	200 MB
Size of Application Codes	7MB	1 MB
Execution Time on a Base Machine (P4 1.6GHz CPU, 512MB Mem.)	21,900 sec.	28,038 sec.

#### 4.3.1 Partial Replication Case

Figure 3 describes partial replication case we assume. We make an assumption that the database is replicated on 4 distributed sites (gray boxes). Network bandwidth between sites shows a wide range of differences. Figure 4 describes the result from PSI-BLAST on the assumed replication case. As you see the results, the site C takes the least execution time for all sites. However, it doesn't have the sequence database, so the database should have been fetched from other sites.



**Figure 3. Partial replication case**



**Figure 4. Performance of PSI-BLAST in partial replication case**

It takes 743 seconds to bring the database from the remote site which has the widest bandwidth from site C (in this experiment, site E was chosen). The job execution time and database replica fetch time is nearly the same at the site C, so overall response time becomes long. Although site A takes longer execution time than site B and C, it was chosen for job execution by Chameleon. Since site A has the required database, it takes only 153 seconds to deliver the application code and result data. In this case, the scenario 4 (Remote Data and Same Remote Execution) is selected. If we only considered computational capabilities, site C would be chosen and performance would be degraded. Although the predicted execution time and data transfer time are not exactly the same as the values from real execution, the prediction errors don't make the wrong decision in this experiment. While the predictions of job execution time have been relatively accurate since the application's performance is tightly dependent on the system attributes (CPU, Memory size, etc.), there were large prediction errors on the data transfer time in our experiments. It is because the data transfer speed using GridFTP was not always proportional to the bandwidth reported by NWS.

Figure 5 describes the result of FASTA on the same replication case. Interestingly, the result is different from the case of PSI-BLAST. The size of output data from FASTA is about 200 MB, which is much larger than that of PSI-BLAST, as it contains much more information about protein functions and structures. During our experiments, the local site has shown narrow network bandwidth, so much time was consumed for transmitting the output data to local site. Although the site A has the required database, it really suffers from transmitting the output data to local since the network link between site A and local site shows the worst performance. As a result, Chameleon selects the scenario 5 (Remote Data and Different Remote Execution) and site C is selected for job execution. The database should be fetched from the other remote site.

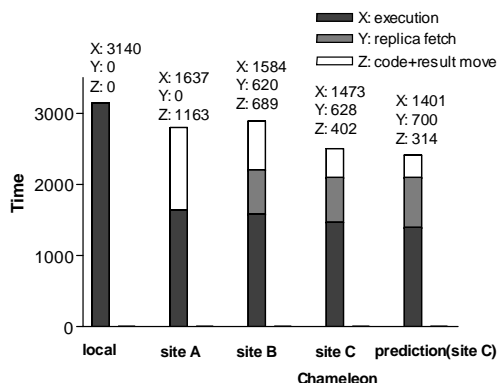


Figure 5. Performance of FASTA in partial replication case

#### 4.3.2 Non-Replication Case

In this case, we examine the situation that the data is only in local. In other words, Data Grid features are not concerned. The result is obtained from the execution of PSI-BLAST. As you see the results in Figure 6, much time is consumed to transmit the large size database from local to remote sites. Although the site A and site G have larger computational capabilities than local site, the response time is longer when PSI-BLAST is executed in Site A or Site G due to the data transfer delay. Among all sites, the site C is expected to take the least response time. As a result, the site C is chosen for job execution and it takes 2,098 seconds to complete the job. Although it takes a little bit shorter time than local (2,277 seconds), the response time is much longer than that of the partial replication case (1,504 seconds). We can argue that data replication mechanism can improve the performance of those kinds of applications when they are executed on a Grid environment.

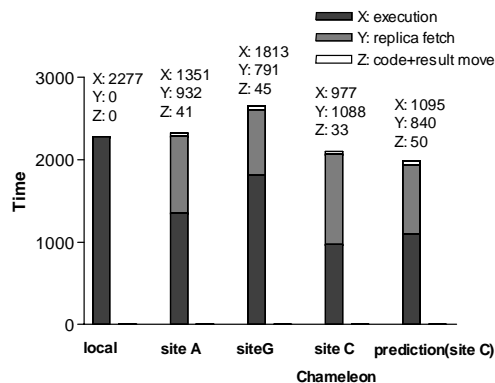


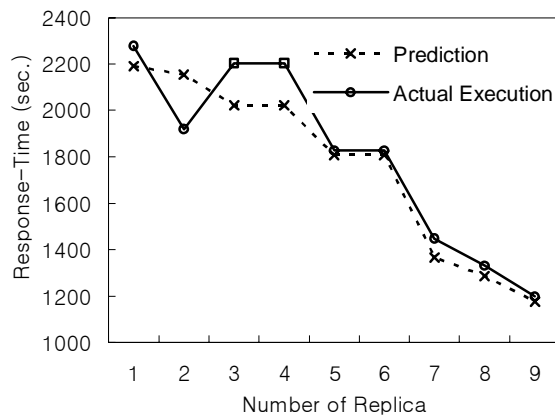
Figure 6. Performance of PSI-BLAST in non-replication case

#### 4.3.3 Effects from Increasing the Number of Replica

In this experiment, we discuss the benefits from increasing the number of replica on the distributed sites. If the dynamic replica management method is applied to the Data Grid, proper number of replica and replica location will be decided according to the replication strategies [18]. In this experiment, we increase the number of replica on distributed sites as illustrated in Table 2. Site names in table 2 are from Figure 3. Figure 7 describes the results. As the number of replica increases, both the predicted response time by Chameleon and the actual response time are decreased. The reason is that the database replica may exist on the site which takes less execution time or the database replica may exist on the site which has better network link to the site of better computation, as the number of replica increases.

Table 2. Increasing the number of replica in distributed sites

Number of Replica	Sites with Replica
1	Local
2	Local, E
3	Local, E, D
4	Local, E, D, F
5	Local, E, D, F, G
6	Local, E, D, F, G, H
7	Local, E, D, F, G, H, B
8	Local, E, D, F, G, H, B, A
9	Local, E, D, F, G, H, B, A, C



**Figure 7. Response time as the number of replica increases**

## 5. Conclusions and Future Work

In this paper, we addressed scheduling mismatch problem in Data Grid environment. We proposed scheduling models that consider both computational capability and location of data replica. Our model consists of five scheduling scenarios that may occur in real situations. We identified cost model for each scenario. We implemented Grid scheduling tool, called Chameleon, based on the proposed models. We performed meaningful experiments on a Grid testbed and we could achieve better performances by applying Chameleon to the gene sequence comparison applications.

Currently, the scheduling models are designed for job execution on one site. We expect that more complex models should be constructed for job execution on multiple sites. In our future work, we will extend our models for the case of job execution on multiple sites. Hence, more advanced features will be added to Chameleon. We will demonstrate by large-scale experiments with the extended job scheduling models for Data Grid environment.

### Acknowledgement

We thank to all ANTZ and ApGrid testbed participants for their providing computing resources.

### References

- [1] ANTZ: <http://www.antz.or.kr>
- [2] ApGrid: <http://www.apgrid.org>
- [3] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke. "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," IEEE Mass Storage Conference, 2001.
- [4] Mark Baker, Rajkumar Buyya and Domenico Laforenza.

"The Grid: International Efforts in Global Computing," International Conference on Advances in Infrastructure for E-Business, Science, and Education on the Internet, SSGRR2000, L'Aquila, Italy, July 2000.

- [5] F. Berman and R. Wolski. "The AppLes project: A status report," Proceedings of the 8th NEC Research Symposium, Berlin, Germany, May 1997.
- [6] Rajkumar Buyya, Kim Branson, Jon Giddy and David Abramson. "The Virtual Laboratory: A Toolset for Utilising the World-Wide Grid to Design Drugs," 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2002), Berlin, Germany, May 2002.
- [7] CERN DataGrid Project: <http://www.cern.ch/grid/>
- [8] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury and Steven Tuecke. "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Applications, 23:187-200, 2001.
- [9] Dirk Düllmann, Wolfgang Hoschek, Javier Jean-Martinez, Asad Samar, Heinz Stockinger and Kurt Stockinger. "Models for Replica Synchronisation and Consistency in a Data Grid," 10th IEEE Symposium on High Performance and Distributed Computing (HPDC-10), San Francisco, California, August 2001.
- [10] I. Foster and C. Kesselman. "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 1999.
- [11] I. Foster, C. Kesselman and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International J. Supercomputer Applications, 15(3), 2001.
- [12] Cynthia Gibas. "Developing Bioinformatics Computer Skills," O'REILLY, April 2001.
- [13] The Globus Project: <http://www.globus.org>
- [14] Leanne Guy, Erwin Laure, Peter Kunszt, Heinz Stockinger, and Kurt Stockinger. "Replica management in data grids," Technical report, Global Grid Forum Informational Document, GGF5, Edinburgh, Scotland, July 2002.
- [15] Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger and Kurt Stockinger. "Data Management in an International Data Grid Project," 1st IEEE/ACM International Workshop on Grid Computing (Grid'2000), Bangalore, India, Dec 2000.
- [16] Kavitha Ranganathan and Ian Foster. "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications," 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002.
- [17] Kavitha Ranganathan and Ian Foster. "Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid," International Conference on Computing in High Energy and Nuclear Physics, Beijing, September 2001.
- [18] Kavitha Ranganathan and Ian Foster. "Identifying Dynamic Replication Strategies for a High Performance Data Grid," International Workshop on Grid Computing, Denver, November 2001.
- [19] Heinz Stockinger, Kurt Stockinger, Erich Schikuta and Ian Willers. "Towards a Cost Model for Distributed and Replicated Data Stores," 9th Euromicro Workshop on Parallel and Distributed Processing PDP 2001, Mantova, Italy, February 2001.
- [20] S. Vazhkudai, S. Tuecke and I. Foster. "Replica Selection in the Globus Data Grid," Proceedings of the First IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001), Brisbane, Australia, May 2001.
- [21] Rich Wolski, Neil Spring, and Jim Hayes. "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," Journal of Future Generation Computing Systems, Volume 15, Numbers 5-6, pp. 757-768, October 1999.