

공학 석사학위 논문

그리드 컴퓨팅 환경에서의 성능 개선을 위한
작업 스케줄링 및 데이터 복제 기법

A Research on Job Scheduling and Data Replication
Mechanism in Grid Computing Environments

아주대학교 정보통신전문대학원
정보통신공학과
박 상 민

그리드 컴퓨팅 환경에서의 성능 개선을 위한
작업 스케줄링 및 데이터 복제 기법

A Research on Job Scheduling and Data Replication
Mechanism in Grid Computing Environments

지도교수 김 재 훈

이 논문을 공학 석사학위 논문으로 제출함

2003 년 12 월 23일

아주대학교 정보통신 전문대학원

정보통신 공학과

박 상 민

박상민의 공학 석사학위 논문을 인준함.

심사 위원장 김 재 훈 (인)

심 사 위 원 류 기 열 (인)

심 사 위 원 고 영 배 (인)

아주대학교 정보통신 전문대학원

2004 년 1 월 9 일

감 사 의 글

학부 과정의 말기에 처음 연구실에 찾아와 서먹하게 인사했던 시절에는 참으로 어렵게 느껴졌던 교수님과 선배, 동기들이었는데, 2 년의 시간을 함께 보내고 이제 졸업을 앞두고 보니 아쉬운 마음이 큼니다. 25 년간의 제 인생 중 지난 2 년간은 가장 활동적이었고, 어려움도 많았으며 성취도 컸던 시간이었습니다. 처음 연구실에서 공부를 시작할 때는 혼자서는 아무것도 할 수 없던 학생 이었는데 2 년간의 시간은 저를 활동적이고 스스로 미래를 준비할 줄 아는 사람으로 변화시켰습니다.

우선, 김재훈 교수님과 고영배 교수님께 깊은 감사를 드립니다. 지도 교수님으로서 분산 컴퓨팅 분야에서 매우 심도 깊은 지식을 전달해 주셨고, 또한 여러 논문을 국내외 학술대회에 발표할 수 있도록 적극적으로 지도해 주셨습니다. 교수님들의 노력이 없이는 제 성과들은 불가능했을 것입니다. 또한 두분 교수님 외에도 다양한 분야의 전문 지식들을 따뜻하게, 때로는 엄격하게 지도해 주신 아주대학교의 여러 교수님들께 깊은 감사를 드립니다.

연구실의 동기들과 선후배 들께도 감사를 드립니다. 2 년의 시간 동안 연구 분야에 대해 심도 깊은 지식을 교환하고, 때로는 공부에 지쳐 삭막해지기 쉬운 시간들을 따뜻한 대화로 즐겁게 이끌어준 참으로 좋은 사람들이었습니다. 특별히 동기로서 함께 연구실에 들어와 많은 도움을 주었던 종무형, 수진 누나, 민준형에게 고맙다는 말을 전하고 싶습니다.

사랑하는 교회의 모든 지체들은 기도와 대화로 제게 용기를 주었습니다. 늘 시간 없다는 핑계로 기대를 채워주지 못한 은경에게는 참으로 미안함과 고마움을 함께 느낍니다. 끝으로 누구보다도 저를 적극적으로 후원해주신 부모님께는 감사하다는 말만으로 제 마음을 다 전할 수 없을 것입니다.

요 약

그리드 컴퓨팅의 목표는 인터넷의 분산된 자원을 모아 고에너지 물리나 바이오 인포매틱스와 같은 고성능 어플리케이션이 요구하는 빠른 속도와 안정성을 제공하는 것이다. 최근 그리드 분야에서 데이터와 모바일 그리드가 중요한 주제로 부각되고 있다. 데이터 그리드는 인터넷에서의 대용량 데이터 관리 기법과 어플리케이션이 인터넷의 분산된 데이터 자원을 처리하는 방법을 제공한다. 모바일 그리드는 그리드 컴퓨팅을 노트북과 PDA와 같은 이동 단말기가 컴퓨팅 자원을 구성하는 무선 이동컴퓨팅 환경으로 확장하는 기법을 다룬다. 데이터와 모바일 그리드 환경에서 작업 스케줄링은 성능을 결정하는 중요한 요소이며 기존의 분산 시스템 스케줄링과는 차이가 있다. 그러나 최근까지 두 가지 특수한 형태의 그리드 컴퓨팅을 지원하기 위한 스케줄링 연구는 많지 않았다. 본 논문에서는 데이터와 모바일 그리드 어플리케이션의 성능을 높이는 작업 스케줄링 기법을 각각 제시하고 성능을 평가하였다. 각각의 스케줄링 기법은 기존 그리드 환경에서 고려되지 않았지만 성능에 영향을 미치는 요소를 자원의 스케줄링시에 고려함으로써 성능을 향상 시킨다. 또한 스케줄링외에 데이터 그리드 어플리케이션의 성능에 큰 영향을 미치는 것은 동적 복제 데이터 관리 기법이다. 본 논문에서는 기존의 동적 복제데이터 관리 기법에 비해 성능 향상을 보장하는 새로운 동적 데이터 관리 기법을 스케줄링 기법에 이어 제시하고 성능을 평가하였다. 새로운 복제 데이터 관리 기법은 인터넷의 계층성을 고려함으로써 성능을 향상시킬 수 있다.

Abstract

The aim of grid computing is to aggregate Internet-wide distributed resources so that high performance applications such as High Energy Physics and Bioinformatics can be executed guaranteeing fast speed and reliability. Two important topics newly emerging in the grid community are data and mobile grid. The data grid focuses on providing the way to solve large scale data management problems in Internet and to execute data intensive applications which needs access to distributed large data sets. The mobile grid promises extension of grid computing to wireless mobile environment so that thin mobile devices such as Laptops, PDAs, Handhelds can participate in grid as convenient tools to access grid or more valuable resource providers. In the context of data and mobile grid, scheduling of a job is a key research issue to improve the application performance and should be dealt differently from that of traditional distributed systems. Not many researches, concerning on this scheduling problem in these two specific forms of grid computing, have been performed yet. In this thesis, I present two scheduling algorithms suitable for data and mobile grid applications, respectively. These scheduling algorithms can yield better performance by considering unique characteristics of data and mobile grid environment. In addition to the scheduling, dynamic replica management is also one of the key techniques for delivering good performance of data grid application. I present novel strategy of dynamic replica management which guarantees better performance compared to existing strategies. The performance improvements could be achieved by taking hierarchy of Internet into account as one of the source of file access locality.

Contents

Chapter 1. Introduction	1
--------------------------------------	----------

Chapter 2. A Study on Effective Job Scheduling Algorithms

 In Data Grid Environments	3
2.1 Introduction	3
2.2 Related Work	4
2.3 Scheduling Model	5
2.3.1 System Factors	
2.3.2 Cost Models	
2.4 Scheduler Implementation and Results	11
2.4.1 Chameleon – Scheduler Implementation	
2.4.2 Testbed and Applications	
2.4.3 Experimental Results	
2.5 Conclusions	17

Chapter 3. A Study on Effective Job Scheduling Algorithms

 In Mobile Grid Environments	18
3.1 Introduction	18
3.2 Related Work	19
3.3 Two Roles of Mobile Devices in Mobile Grid Computing	20
3.3.1 Mobile Interface Role to Grid	
3.3.2 The Service Provider Role to Grid	
3.4 The Proposed Job Scheduling Algorithm	25
3.5 Performance Evaluation	29
3.6 Conclusions	34

Chapter 4. A Study on Replica Optimization Strategy	
In Data Grid Environments	35
4.1 Introduction	35
4.2 Related Work	36
4.3 Dynamic Replication Strategy based on Bandwidth Hierarchy	37
4.4 Experiments	39
4.4.1 Simulation Tool	
4.4.2 Configuration	
4.4.3 Results	
4.6 Conclusions	43
Chapter 5. Conclusions	44
References	46

LIST OF FIGURES

Fig. 2-1. Data Grid Environment	5
Fig. 2-2. Data movement in models.....	8
Fig. 2-3. Partial replication case	14
Fig. 2-4. Performance of PSI-BLAST in partial replication case	14
Fig. 2-5. Performance of FASTA in partial replication case	15
Fig. 2-6. Performance of PSI-BLAST in non-replication case	16
Fig. 2-7. Response time as the number of replica Increases	17
Fig. 3-1. Mobile Grid System.....	21
Fig. 3-2. Mobile resource provision to Grid using MQS.....	23
Fig. 3-3. Job execution process on a mobile node	25
Fig. 3-4. State transition diagram of mobile node	25
Fig. 3-5. Job scheduling algorithm for the mobile grid system	29
Fig. 3-6. Benefits and losses from increasing the participating nodes to the computing... ..	29
Fig. 3-7. Job execution time in four mobile environments	31
Fig. 3-8. Job execution time in stable environments	33
Fig. 3-9. Job execution time in highly disconnective environment.....	33
Fig. 3-10. Job execution time in unstable environment.....	33
Fig. 3-11. Job execution time in highly connective environments	34
Fig. 4-1. Bandwidth hierarchy in Internet.....	38
Fig. 4-2. BHR replication algorithm.....	39
Fig. 4-3. Grid topology in simulation	40
Fig. 4-4. Total job times with parameters shown in Table 4-1 and Table 4-2	42
Fig. 4-5. Total job time with varying bandwidth and storage size.....	43

LIST OF TABLES

Table 2-1. Parameters in chapter 1	13
Table 2-2. Increasing the number of replica in distributed sites	16
Table 3-1. Assumed mobile environments	30
Table 3-2. Experimental environment	30
Table 3-3. Parameters for mathematical analysis and simulations	31
Table 4-1. General configuration of parameters	41
Table 4-2. Bandwidth and storage spaces.....	41

Chapter 1

Introduction

A Grid is large scale resource sharing and problem solving mechanism in virtual organizations [Fos01]. Resources such as computational nodes and data stores are aggregated through Internet and grid software, so that users can execute high performance applications with relatively low cost. Most popular type of grid application include Bioinformatics, high energy physics, and earth observations which typically needs extensive number of processing cycles and large data sets. Grid promises fast and reliable execution of high performance applications by aggregating large amount of resources distributed over the world.

While grid shows us such a rosy scene of future computing platform, there are lots of problems which should be dealt accordingly in order to fully enable grid scenarios. Scheduling from the perspective of grid is different from that of traditional distributed systems. In grid, resources are extremely heterogeneous and distributed in large area, and connected through relatively slow and unreliable Internet. Thus existing distributed scheduling algorithms which typically assume homogeneous resources within local area network is not suitable to be applied to grid applications. New scheduling algorithms fitted for grid scenarios have been proposed so far [Ber97,Abr00,Gou00].

In recent, data and mobile grid are becoming important and distinct area in grid communities. In data grid, data is regarded as the center of grid architecture because grid application in data grid needs access to distributed large data sets. Mobile grid is another newly emerged area which extends grid resources to wireless mobile devices such as laptops, PDAs, and handheld devices. New scheduling algorithm in the context of data and mobile grid are required since characteristics of data and mobile resources have not been properly considered. In this thesis, I propose new scheduling algorithms which take unique characteristics of data and mobile grid into account.

In addition to the scheduling problems in data grid, dynamic replication is another key to the performance of data grid applications. Data grid adopts the replication technique in order to improve data access speed, and dynamic replication aim to intelligibly locate replicas by predicting usage pattern of data. Most existing dynamic replication strategies focus on the usage pattern of data in a local site level, but in this thesis, I propose novel replication strategy which focuses on the network level optimization.

Therefore, I present three schemes separately in this thesis like followings:

- (1). Job Scheduling Algorithm in Data Grid Environment
- (2). Job Scheduling Algorithm in Mobile Grid Environment
- (3). Dynamic Replication Strategy in Data Grid Environment

Chapter 2

A Study on Effective Job Scheduling Algorithms in Data Grid Environments

2.1 Introduction

A Grid is large scale resource sharing and problem solving mechanism in virtual organizations [Fos01]. 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 [Fos99]. 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 [Che01].

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 chapter, 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.

2.2 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 [Hos00]. The analysis of data from the experiments is computationally intensive. The integrating architecture motivated by these considerations is a Data Grid [Che01]. 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 [All01]. 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 [Che01]. Stockinger et al. describe basic cost models for analyzing replicated and geographically distributed data in reference [Sto01]. 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 [Ber97], 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 chapter 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 [Ran02]. 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.

2.3 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. Fig. 2-1 shows the Data Grid model we assume. This model takes a similar form to the one CERN presents [Hos00]. 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.

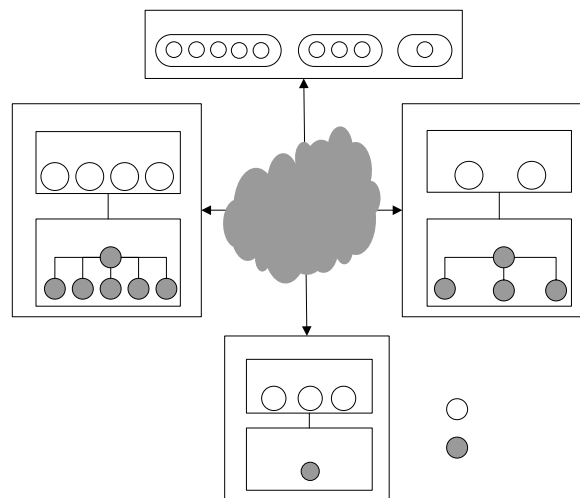


Fig. 2-1. Data grid environment

2.3.1 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.

2.3.1.1 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 [Wol99] 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.

2.3.1.2 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.

2.3.2 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 [Hos00]. 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

Fig. 2-2 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.

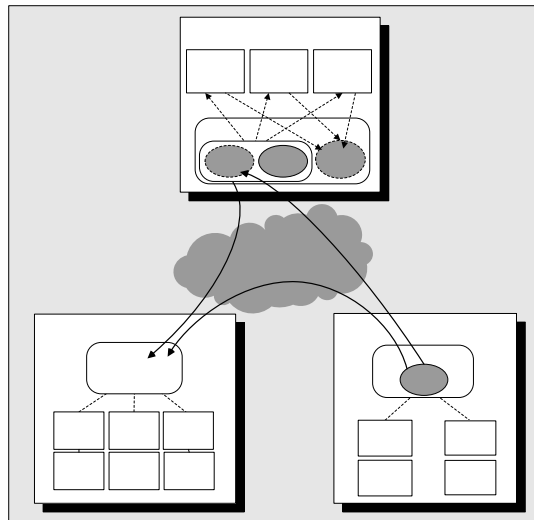


Fig. 2-2. Data movement in models

2.3.2.1 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 Fig. 2-2. 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).

2.3.2.2 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 Fig. 2-2. 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.

2.3.2.3 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.

2.3.2.4 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.

2.3.2.5 Remote Data and Different Remote Execution

$$Time_5 = \frac{D_{input}}{BW_{WAN}(remote_i-remote_j)} + \frac{D_{app} + D_{output}}{BW_{WAN}(local-remote_j)} + \frac{N_{remote_j} \times (D_{input} + D_{app}) + D_{output}}{BW_{LAN}(remote_j)} + Exec_{remote_j}$$

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.

2.4 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.

2.4.1 Chameleon

Chameleon is built on top of Globus [Globus]. NWS (Network Weather Service) [Wol99] 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.

2.4.2 Testbed and Applications

We perform experiments on a Grid testbed in S. Korea, named ANTZ [Antz], and ApGrid testbed [Apgrid]. 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 [Gib01], 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 [Gib01]. 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.

2.4.3 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 2-1. 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 2-1. Parameters in Chapter 1

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.

2.4.3.1 Partial Replication Case

Fig. 2-3 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. Fig. 2-4 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.

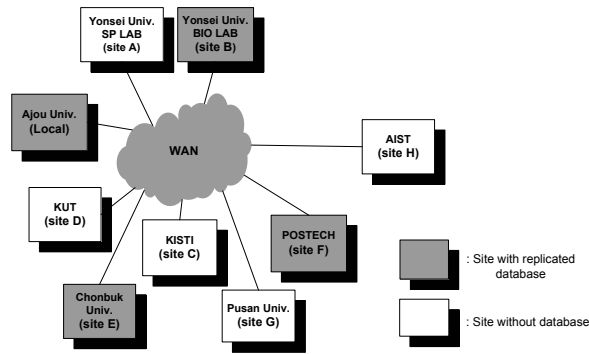


Fig. 2-3. Partial replication case

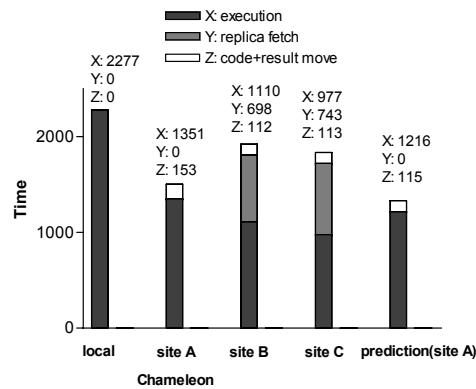


Fig. 2-4. Performance of PSI-BLAST in partial replication case

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.

Fig. 2-5 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.

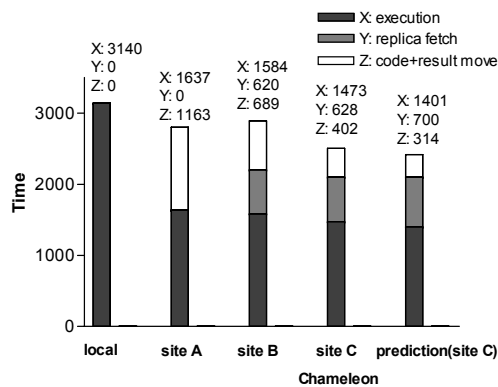


Fig. 2-5. Performance of FASTA in partial replication case

2.4.3.2 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 Fig. 2-6, 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.

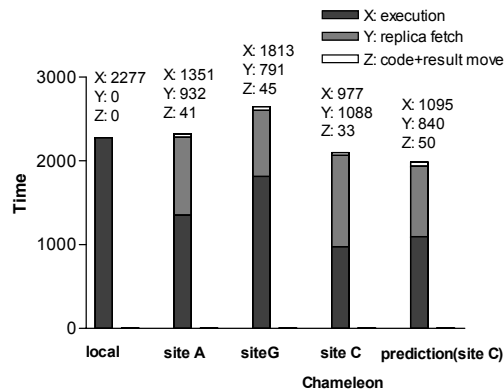


Fig. 2-6. Performance of PSI-BLAST in non-replication case

2.4.3.3 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 [Ran01]. In this experiment, we increase the number of replica on distributed sites as illustrated in Table 2-2. Site names in table 2-2 are from Fig. 2-3. Fig. 2-7 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 2-2. 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

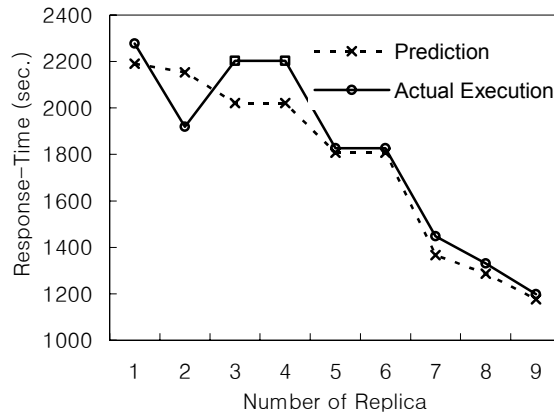


Fig. 2-7. Response time as the number of replica Increases

2.5 Conclusions

In this chapter, 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.

Chapter 3

A Study on Effective Job Scheduling Algorithms in Mobile Grid Environments

3.1 Introduction

A grid computing system [Fos01] is a large-scaled distributed system, designed for effectively solving very complicated scientific or commercial problems such as gene analysis, drug design, and climate simulations. In the grid, the computing resources are autonomously managed at different locations in a distributed manner. They are aggregated through a global Internet-wide network, forming a computational grid, by which a huge workload can be run and completed with more improved performance in terms of computation speed and throughput. Mobile computing [Sat96] is another distinct paradigm of traditional distributed systems, considering mobility, portability and wireless communications. The recent advances in wireless communication technologies and portable mobile appliances make more number of people be eligible to access information services through a shared network infrastructure (e.g., Internet) with their own mobile computing devices, regardless of their physical location.

The current grid architecture and algorithms do not take into account the mobile computing environment since mobile devices have not been seriously considered as valid computing resources or interfaces in grid communities. It has been just recently given attention to integrate these two emerging techniques of mobile and grid computing –for example, in [Pha02,Bri02], although they do not elaborate on how the mobile devices may be incorporated in the current grid architecture. In our view of the mobile grid computing integration, there are two possible roles of mobile devices in grid. First, mobile devices can be used as interfaces to the grid. Thus, a mobile device can initiate the use of grid resources, monitor the jobs being executed remotely, and take any results from the grid. Secondly and more interestingly, mobile devices can be

assumed to participate in grid as computing resource providers, not just service recipients. We believe that recent advancement of technologies on mobile devices and wireless communications make this scenario more feasible.

Of course, clearly, many issues become the challenges when we consider mobile devices as one of grid computing resources or interfaces. Some examples of limitations that possibly hinder the integration are relatively poor local resources (in terms of computation speed, memory), battery constraints, unreliable connectivity status, weak security and so on. These limitations and constraints should be dealt with accordingly before mobile grid integration is fully enabled. We present several technical issues of mobile grid system in the first part of this chapter. However, among those arise, we particularly focus on a dynamic nature of device connectivity and develop a new scheduling algorithm that provides reliable performance in a mobile grid system where the devices are prone to frequent disconnections.

3.2 Related Works

One of the most critical characteristics of the mobile grid system is the intermittent connectivity of mobile devices. We can find similar situations in Peer-to-Peer computing area. In general, P2P system consists of huge number of computing devices and they can act either as a client or a server. In P2P, each machine's CPU cycles, storages, and contents can be shared in order to broaden their resource limitations [Led03,Wil02,Seti,Kon02]. SETI@home project [Seti] provides a successful story that large P2P system can be effectively used for high performance applications by aggregating cycles of desktop PCs. In a P2P System, users are free to join and leave the network and one study showed that a node remained in a connection state only for 28% of time on average [Wil02]. These frequent disconnections of the node degrade the computing performance significantly and providing the redundancy of workloads is the way used to compensate the deficiency [Kon02]. In other words, the same work unit is disseminated into some number of nodes and the first result generated by the most reliable and the fast node is received. However, this strategy of redundancy cannot be applied in mobile grid systems. While the redundancy guarantees the improvement of performance, it also wastes the resources greatly. Although the strategy can be successfully applied in P2p systems where resources are

abundant, relatively small number of nodes and high sensitivity of resource dissipation in mobile grid system will not allow large number of duplication of work units. In this chapter, we present different approach to deal with performance degradation. We propose a new scheduling algorithm that takes into account of the intermittent connectivity of mobile nodes.

Since grid consists of several number of autonomously managed local resources (MPPs, clusters, and workstations), the scheduling paradigm would fall into two categories, global and local scheduling. Various global scheduling algorithms have been operational such as application-level scheduling [Ber97], high-throughput scheduling [Condor], economy-based scheduling [Abr00], and data-centric scheduling [Par03]. Once the grid job is scheduled and submitted by global scheduler, it is scheduled again by local resource managers like PBS [Openpbs], LSF [Lsf], and Condor [Condor]. First-in, first-out is the most prominent scheduling policy in the local managers, yet more sophisticated algorithms are provided within the specific managers. The scheduling algorithm that we present in this chapter would be classified in local scheduling since the scheduling decision is only applicable to local mobile resources.

3.3 Two Roles of Mobile Devices in Mobile Grid Computing

In this section, we present possible architecture of the mobile grid system and several technical issues that are to be dealt with in further researches. We depict an expected view of mobile grid system in Fig. 3-1. As illustrated in the Figure, the grid system is divided into three parts: static grid sites, a group of mobile devices, and a gateway interconnecting static and mobile resources. Mobile devices are wirelessly connected to Internet through a wireless LAN or a long-range cellular network like GSM and CDMA. Note that, in the figure, grid middleware, e.g. Globus [Globus], is installed on the gateway to provide interoperability between the virtual mobile grid site and other static grid sites. Also note that such a middleware layer is not necessary to be in a mobile device itself. This is important because our model does not require a heavy grid middleware, like Globus, to be installed on the thin mobile device. Newly added grid components realizing mobile grid computing are comprised of M-Agent (Mobile Agent), and

MQS (Mobile Queuing Server). They are both on the gateway and mobile device side.

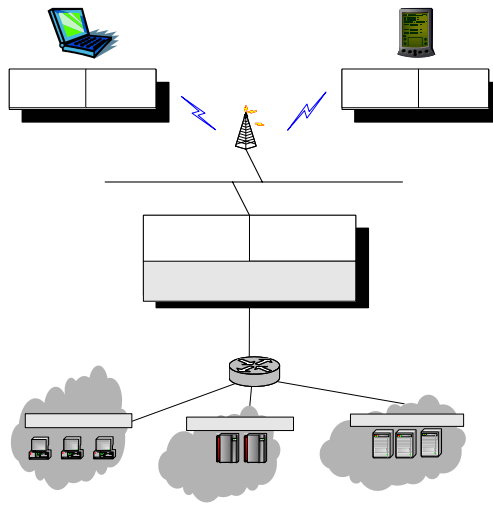


Fig. 3-1. Mobile Grid System

3.3.1 Mobile Interface Role to Grid

Mobile devices are exposed to frequent disconnections from the network. Also the mobile system itself is unreliable. Hence, it is not a good idea for the mobile devices to interact with static grid sites directly. Simply imagine that the mobile devices are suddenly disconnected during the data exchange with grid sites. Sometimes, the jobs running on grid sites need interactions with mobile user while mobile user is not in the connection state. Thus, a reliable static system should perform job submission, monitoring, cancellation, and completion process on behalf of the mobile user. In our architecture, M-Agent on the gateway performs that role of agent for the unreliable mobile devices. The M-Agent may perform two key roles as follows:

- **Various mobile device interfaces support:** M-Agent on mobile device interacts with the mobile user, that is, the mobile user submits, monitors the grid job, and observes the results, through M-Agent. Various kinds of mobile devices have different appearances. While laptop has similar interfaces with PCs, interfaces of PDA or Handheld device is far from that of PC's. They

have smaller displays, no keyboards, etc. Hence, the interface of M-Agent on mobile device should adapt to the specific class of mobile devices while sustaining the same functionality.

- **Reliable job management:** The mobile device has several constraints. The most serious problem is that its availability is unreliable due to intermittent connectivity or limited battery life problems. Thus, it is not a good approach to manage the running job directly from the mobile device. In order to solve problems induced by unreliable availability, we propose an object, **Job-Proxy**. When a mobile user submits a job through M-Agent, the M-Agent on gateway creates a Job-Proxy which performs job related tasks according to the received information from the mobile user. If the M-Agent detects an error with mobile devices, Job-Proxy becomes operational and starts to monitor executing job on the grid sites. If some interactions between the executing job and mobile user are needed (e.g., when an additional input data should be given), the Job-Proxy performs the interaction on behalf of the mobile user, based on the information received from the mobile user beforehand. When the grid job is completed, the result from the job is temporarily stored on the Job-Proxy. If the mobile device is in a good condition to receive the results, it immediately relays the result data. Otherwise, it waits until the mobile device becomes ready for receiving the result data. The mobile user may set the time-out duration of the Job-Proxy, i.e., the running job in static grid sites will be canceled after a certain amount of time and all the resources in gateway allocated for the Job-Proxy is freed. This is because the mobile user may not be able to explicitly request cancellation of the job although running job is not needed anymore. If we do not cancel the job automatically, sustaining job execution unnecessarily consumes resources in both static grid sites and the mobile grid gateway.

3.3.2 The Service Provider Role to Grid

We present another aspect of mobile grid system that supports the integration mobile devices as grid resources. Even though, currently, both long-range access technologies (GSM and CDMA) and short-range connectivity(WLAN) is available, we assume that the WLAN is provided as a wireless medium for the mobile grid system. This is because not only does the wireless LAN guarantee much faster data transfer rate (11Mbps for the IEEE 802.11b), but also it is easier to control with respect to number of nodes, security concerns, and network

administrations. This is because not only does the wireless LAN guarantee much faster data transfer rate (11Mbps for the IEEE 802.11b), but also it is easier to control with respect to number of nodes, security concerns, and network administrations. In addition to them, WLAN is the most favored method of wireless communication for computing devices (laptops and PDAs), while the cellular networks are still dominantly used for voice communications. Except that the computing nodes are not connected through physical lines, the mobile grid system becomes similar to the conventional high performance clusters.

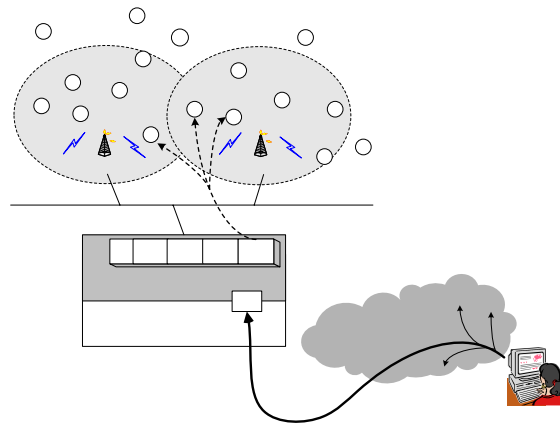


Fig. 3-2. Mobile resource provision to Grid using MQS

As shown by Fig. 3-2, MQS (Mobile Queuing Server) on the gateway is a core component for the integration. The grid user submits jobs to several grid sites, and one of the grid sites performing the job is comprised of mobile devices. The group of mobile devices in the mobile grid system is viewed as a single virtual machine having several computing nodes. The virtual composing is provided by MQS in our architecture. When the job is submitted to this mobile grid site, the grid middleware (e.g., Globus) on the gateway relays the job information such as executable file location, input parameters and the location to store output files, to the MQS. Basic functions of MQS are similar to the conventional Job Queuing systems such as PBS, LSF, and NQE [Cza98]. In addition to the basic job queuing functions, MQS cares the heterogeneity and unreliability of mobile nodes in order to hide unique characteristics of them. The mobile resources are viewed as a reliable single machine like the one in static grid site so that the grid

job initiator can make use of them without further complex considerations. Followings are detailed explanation of services MQS should provide. Except for the scheduling issue, we leave these services as future research topics.

- **Fault Tolerant Service:** Unlike in job queuing service in static systems, availability of mobile resources changes dynamically. The node may suddenly disappear due to disconnections or exhausted power, thus MQS should carefully monitor the status of mobile nodes. Fault tolerant services should be provided by MQS so that the unreliability of mobile node does not decrease the performance much. Time out strategy which re-submits the job when the result is not received in a determined time limit would be one of the viable approaches.

- **Hiding Platform Heterogeneity:** One of the most complex problems to deal with in mobile grid system is the hardware and OS heterogeneity. Large portion of laptop PC's are operated in Microsoft Windows, while, currently, most grid applications are running on Unix-based platforms. When we take the PDA and Handheld devices into account, the problem becomes more serious. Enforcing the care of heterogeneity to grid programmer would not be feasible, thus the MQS should hide the heterogeneity of individual mobile device.

- **Job Decomposing Service:** The job may contain many workloads. As an example, if we assume the job is Bioinformatics application which analyzes the gene sequences, the job will be composed of many gene sequences to be analyzed. A sequence is viewed as a workload in this case. The overall workloads in a job can be processed by a single node in a static grid site within a reasonable time unit. However, a single mobile node may not process the same amount of workloads in a reasonable time due to its resource limitation and unreliability, thus the workloads in a job should be decomposed and distributed to several mobile nodes according to their performance and reliability. MQS needs information about the application's workload composition to perform the job decomposing service, and this information can be provided by job initiator.

- **Job Scheduling:** In a static grid environment, job scheduling policy on a grid site is generally based on a simple FCFS (First Come, First Served). In some systems, it is based on the computing capability, i.e., if the job arrives at a grid site, the job is distributed to the node which is likely to compute it within the shortest time. Reliability of computing node is not a first-class concern for the job scheduling in the static grid environment. However, if the mobile environment is taken into account, we should carefully consider the inherent unreliability of

mobile devices in order to make optimal scheduling decision so that the reliable performance could be guaranteed. We cover this scheduling issue in the following section 3.4.

3.4 The Proposed Job Scheduling Algorithm

Among various challenging issues like energy sensitivity and weak security, we concentrate on the unreliable connectivity of mobile environment hereafter. Frequent disconnections of mobile node affect the job performance significantly. Job scheduling is made based on the expected execution time on various mobile nodes. When there is available mobile node, job (workload) is extracted from the job queue and delivered to the mobile node with related input data. After completing job execution on a mobile node, job output is returned to the MQS (Mobile Queuing Server) and the mobile node performs next job. Fig. 3-3 represents the job execution process performed on the mobile node. Since Master-Worker style applications, whose tasks can be scheduled independently of one another, are preferred form of applications in grid [Condor,Abr00,Gou00], the job on the mobile node does not need communication during its execution. However, the mobile node should be in connection state during input and output data transfer. If the mobile node is not in the connection state, MQS waits for its reconnection. Fig. 3-4 shows the state transition diagram of a mobile node. State C and D denotes the connection state (C) and disconnection state (D), respectively. λ is the disconnection rate, and μ is the reconnection rate of the mobile node, which are governed by Poisson process.



Fig. 3-3. Job execution process on a mobile node

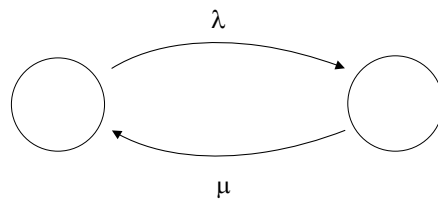


Fig. 3-4. State transition diagram of mobile node

When a number of jobs arrive at the MQS, or a number of workloads are decomposed from a single job, the MQS selects certain mobile nodes which perform the jobs. In a static grid environment, we achieve improving performance as the number of participating computing nodes grows. However, the rule changes when the mobile devices participate in the computing. Links of some mobile nodes are in unreliable state, that is, they are likely to be in the disconnection state for a long time while others are not. The unreliable mobile nodes certainly decrease the overall job execution performance because the MQS should wait for the reconnection of any disconnected node in order to complete the jobs. Hence, we cannot guarantee maximum performance even if all the mobile nodes participate in the computing. Including an unreliable mobile node in the computing may decrease the job execution performance. We should determine whether it is better to include a mobile node in the computing or not. Followings are the analysis to determine the participating nodes to achieve the best job performance.

We consider all the mobile nodes (i.e., member nodes) as the potential participants in the computing whether they are initially in the connection state or not, at the time of job distribution. Followings are the analysis to obtain the expected time to execute a job on the mobile nodes.

Let $f_c(t)$ and $f_d(t)$ denotes the expected time required to transfer the data (input and output) of t time unit (data transfer requires t time unit without disconnection) when the mobile node is initially in the connection state and disconnection state, respectively. Let P_C and P_D be the probability of being in connection state and disconnection state, respectively. We can obtain following equations from state transitions shown in Fig. 3-4:

$$\begin{aligned} P_C + P_D &= 1 \\ -\lambda P_C + \mu P_D &= 0 \end{aligned}$$

We can compute P_C and P_D as follows:

$$P_C = \frac{\mu}{\mu + \lambda}, \quad P_D = \frac{\lambda}{\mu + \lambda}$$

Then, $C(t, \varepsilon) = f_c(t + \varepsilon) - f_c(t)$ is the time required to perform ε time units of data transfer starting from time t when the mobile node is initially in the connection state. We can compute $C(t, \varepsilon)$ for two cases as follows:

- No disconnection occurs during ε (probability is $e^{-\lambda\varepsilon}$): ε time units are required to

transfer the data.

- A disconnection occurs during ε (probability is $1 - e^{-\lambda\varepsilon}$): $\frac{1}{\mu}$ time units are expected to stay in disconnection state.

ε time units are required for data transfer and $\frac{1}{\mu}$ time units are required on average for

waiting for reconnection. Hence, we can obtain $C(t, \varepsilon)$ as follows:

$$\begin{aligned} C(t, \varepsilon) &= f_c(t + \varepsilon) - f_c(t) \\ &= e^{-\lambda\varepsilon} \varepsilon + (1 - e^{-\lambda\varepsilon}) \left(\varepsilon + \frac{1}{\mu} \right) \end{aligned}$$

To compute $f_c(t)$:

$$\begin{aligned} \frac{\partial f(t)}{\partial t} &= \lim_{\varepsilon \rightarrow 0} \frac{f(t + \varepsilon) - f(t)}{\varepsilon} \\ &= \lim_{\varepsilon \rightarrow 0} \frac{e^{-\lambda\varepsilon} \varepsilon + (1 - e^{-\lambda\varepsilon}) \left(\varepsilon + \frac{1}{\mu} \right)}{\varepsilon} = \frac{\lambda}{\mu} + 1 \end{aligned}$$

Thus, we obtain expected time required to transfer the data of t time unit connection (data transfer requires t time unit without disconnection) when the mobile node is initially in connection state as follows:

$$f_c(t) = \left(\frac{\lambda}{\mu} + 1 \right) t$$

Expected time required to transfer the data of t time unit connection when the mobile node is initially in disconnection state are obtained as follows (In addition, $\frac{1}{\mu}$ time units are required for reconnection on average.):

$$\begin{aligned} f_d(t) &= \frac{1}{\mu} + f_c(t) \\ &= \frac{1}{\mu} + \left(\frac{\lambda}{\mu} + 1 \right) t \end{aligned}$$

The average expected data transfer time, $f(t)$, is,

$$\begin{aligned}
f(t) &= P_c \cdot f_c(t) + P_d \cdot f_d(t) \\
&= \frac{\mu}{\lambda + \mu} \cdot \left(\frac{\lambda}{\mu} + 1 \right) t + \frac{\lambda}{\lambda + \mu} \cdot \left\{ \frac{1}{\mu} + \left(\frac{\lambda}{\mu} + 1 \right) t \right\}
\end{aligned}$$

If $f(t)$ is a function of t , λ , and μ ,

$$f(t, \lambda, \mu) = \left(\frac{\lambda}{\mu} + 1 \right) t + \frac{\lambda}{\lambda + \mu} \cdot \frac{1}{\mu}$$

Let t_{job} , t_{in} and t_{out} denotes job processing time on a mobile node, expected time required to transfer the input data to mobile node, and expected time required to transfer the output data from mobile node, respectively. The response time of single job execution on a mobile node i can be represented as follows:

$$g_i(t_{in}, t_{job}, t_{out}) = f_i(t_{in}) + t_{job} + f_i(t_{out}) \quad \dots \quad (1)$$

Fig. 3-5 shows the algorithm to select the participating mobile nodes in the computing. In step 3 and 4, nodes are listed in an increasing order of response time using equation (1). In step 5, $g_i - g_{i-1}$ means expected additional waiting time, caused by including i^{th} node, after $(i-1)^{th}$ node return the result, while $\frac{W_{total}}{i-1} - \frac{W_{total}}{i}$ means the expected reduced execution time achieved by including i^{th} node in the computing. Thus, i^{th} node is included if it turns out to reduce overall response time comparing to including up to $(i-1)^{th}$ node.

Fig. 3-6 demonstrates the effects of the algorithm. As we include more nodes in the grid computing, each node performs fewer amounts of jobs, thus job processing time per mobile node decreases. However, as we include more nodes, the worst node is likely to be in disconnection state long. Thus, the waiting delay for input and output data transfer will be increased. Thus, it is needed to find the optimal number of participating nodes from tradeoff between deficiency and efficiency.

Step 1. $N :=$ number of mobile nodes in the mobile grid resource pool.
Step 2. $W_{total} :=$ total execution time of jobs to be processed.
Step 3. Obtain the expected response time of all mobile nodes using the equation (1).
Step 4. List the mobile nodes in increasing order of response time.
Step 5. *FOR* ($i=N; i>1; i--$)

$$if \left[-(g_i - g_{i-1}) + \left(\frac{W_{total}}{i-1} - \frac{W_{total}}{i} \right) > 0 \right]$$

$$break;$$

Fig. 3-5. Job scheduling algorithm for the mobile grid system

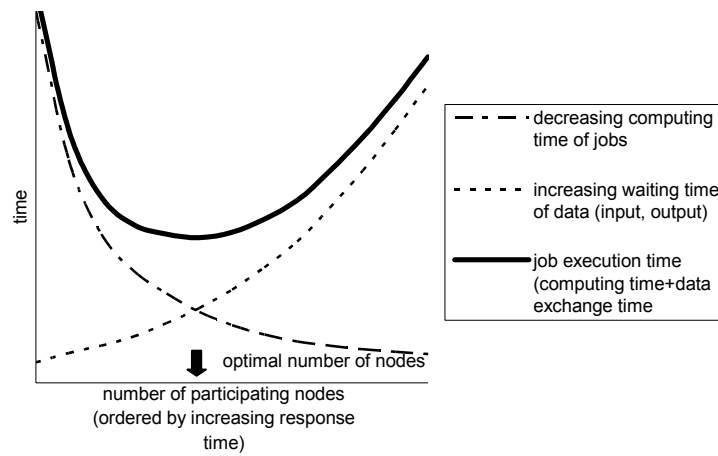


Fig. 3-6. Benefits and losses from increasing the participating nodes to the computing

3.5 Performance Evaluation

We evaluate the proposed job scheduling algorithm using mathematical analysis, simulations and experiments on the implemented prototype. We implement simple job queuing system as prototype of MQS. In the prototype, unreliable connectivity of mobile node is considered as usual behavior. Mobile nodes are repeatedly connected to and disconnected from the master system (MQS). When an available node appears (idle mobile node becomes connection state), the first job in the queue is distributed to the node and executed regardless of node's further connection state. After the job is completed, the node sends result data to master

if it's in connection state, while it waits for the reconnection to the master when it's in disconnection state (e.g., out of communication range).

We assume that each mobile node in the mobile grid system has different value of λ (disconnection rate) and μ (reconnection rate). We generate the λ and μ value (uniformly random) of each mobile node between the fixed minimum and maximum of λ and μ . By differentiating the maximum and minimum value of λ and μ , we assume various mobile environments. We consider four different mobile environments with the low and high rates of disconnection (λ) and reconnection (μ). Assumed environments are stable (low λ , low μ), unstable (high λ , high μ), highly connective (low λ , high μ), and highly disconnective (high λ , low μ), as classified in [Rad99]. The maximum and minimum value of λ and μ representing the environments are presented in Table 3-1.

Table 3-1. Assumed mobile environments

Mobile environments	Max λ	Min λ	Max μ	Min μ
Stable	0.003	0.001	0.003	0.001
Highly disconnective	0.027	0.009	0.003	0.001
Unstable	0.027	0.009	0.027	0.009
Highly connective	0.003	0.001	0.027	0.009

Table 3-2. Experimental environment

Mobile Node Specification (10 Nodes)	CPU	P4 1.6 GHz
	Memory	256 MB
	Network Connection	802.11b wireless LAN
Application	Blast (Bioinformatics)	
Workload amount	Number of protein sequences needed for 1000 seconds of processing	
Network disconnection & reconnection	Exponentially random value with respect to the λ and μ of mobile node	
Data	Input	Decomposed protein sequences (5 KB/number of nodes)
	Output	Information achieved from protein (200 KB/number of nodes)

Table 3-3. Parameters for mathematical analysis and simulations

Parameter	W_{total} (total execution time of jobs)	N (Number of participating nodes)	t_{in} (time required for transferring input data)	t_{out} (time required for transferring output data)
Value	1000 sec.	10 nodes	1 sec.	1 sec.

In the simulation, the disconnection and reconnection of a mobile node is governed by Poisson process with the rate of λ and μ , respectively. We perform experiments on the implemented prototype. The experimental environment is presented in Table 3-2. We adopt a Bioinformatics application, Blast [Gib01], for the experiments. Blast is the gene sequence analysis application comparing newly found protein sequences with well known sequence database and predicts the structure and functions of the newly found one. It needs very intensive processing cycles, thus grid is utilized for executing Blast in many Bioinformatics and grid projects. In the experiments, we analyze a number of sequences which need total 1,000 seconds to process on a single node. The sequences are divided according to the number of participating nodes, and then distributed to the nodes. We manually configured the disconnection and reconnection of mobile nodes in an exponentially random value with respect to the λ and μ in Table 3-1. Upon receiving the protein sequences and job related information (input data), the mobile nodes execute Blast and return the information about the protein (output data) to the master. Table 3-3 describes parameters used in the mathematical analysis and simulations. The parameters correspond to the experimental environment presented in Table 3-2.

Fig. 3-7 describes the analytical results for four different mobile environments based on the analysis in previous section.

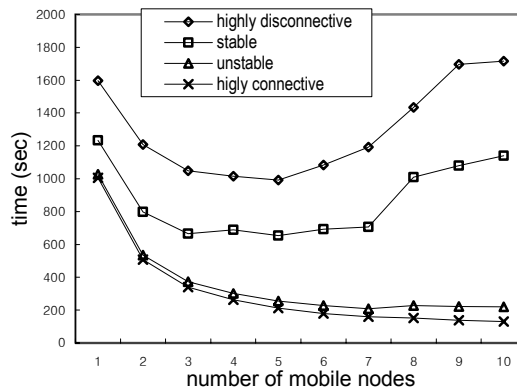


Fig. 3-7. Job execution time in four mobile environments (by mathematical analysis)

We use the turnaround time, which is the time required to complete a batch of tasks, as a performance metric. The x-axis represents the number of participating nodes to compute the jobs. The nodes are arranged in increasing order of expected response time (i.e., the first node is the best one). The y-axis is the turn around time. The turnaround time in highly disconnective environments is the longest among all, and that in highly connective environment is the shortest. The result in unstable environment is comparable to the highly connective case, and result of stable environment is in the middle of them. In unstable and highly connective environments, the mobile nodes remains in disconnected state shortly, thus short time period is consumed to wait to transfer the input and output data to and from the disconnected mobile nodes. On the other hand, longer time is needed to transfer the input and output data in highly disconnective, and stable environments.

As the number of participating nodes increases, the turnaround time in stable and highly disconnective environment decreases until including the nodes close to the critical point. However, it does not decrease any more as the number of participating nodes grows; instead it increases. Consequently, including nodes more than the x value of the critical point makes the turnaround time grow. Thus, the number of participating nodes should be limited to the x value of the critical point. The results from unstable and highly connective environment show different aspect. The turnaround time decreases steadily in these environments. This is because the nodes are not in the disconnection state for a long time so that having more nodes participate to the computing always produces better performance. Yet, the critical point may appear if the scale of nodes grows.

Along with the analytical result, we demonstrate the simulation and experimental results of each assumed environment in Fig. 3-8~3-11. In Fig. 3-8 and 3-9, you can see that the turnaround time of job could be the shortest when participating node is limited to a certain number. The number of nodes which produces the best performance in real experiment is very close to the optimal number (pointed by an arrow) determined by mathematical analysis and simulation which are based on the proposed scheduling algorithm. In Fig. 3-10 and 3-11, results of experiments, simulation and mathematical analysis show that including all available nodes produces the best performance. The reason is the same to the case of above analytical result.

To sum up, we are able to assure that the proposed job scheduling algorithm is a viable approach to adapt in mobile environment where links are prone to frequent disconnections,

especially when mobile nodes remain in disconnection state for longer duration.

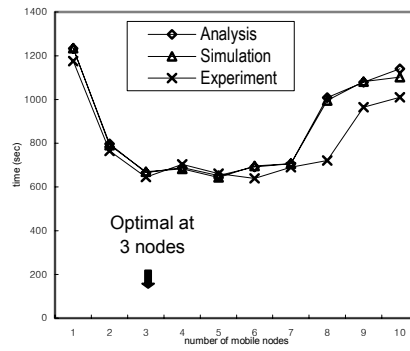


Fig. 3-8. Job execution time in stable environments.
(Analytic optimal number of participating nodes is 3)

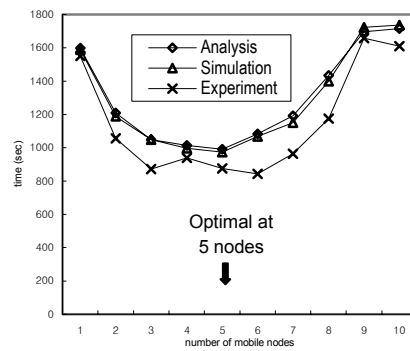


Fig. 3-9. Job execution time in highly disconnective environment.
(Analytic optimal number of participating node is 5)

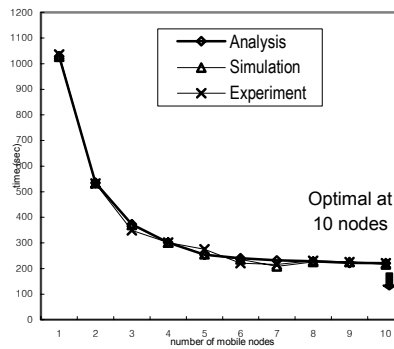


Fig. 3-10. Job execution time in unstable environment.
(Analytic optimal number of participating nodes is 10)

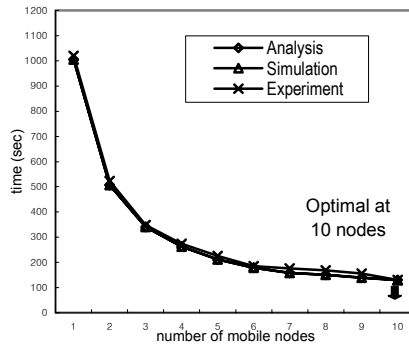


Fig. 3-11. Job execution time in highly connective environments (Analytic optimal number of participating nodes is 10)

3.6 Conclusion

In this chapter, we discuss on the issue of integrating mobile devices into grid. We propose our own view of the mobile/grid integrated system where a gateway links the static grid sites to the group of mobile devices, thus mobile users can make use of static grid resources, and also provide their mobile devices as grid resources. We present the newly emerging technical issues for realizing this mobile grid system, and particularly focus on the job scheduling algorithm to achieve more reliable performance.

Chapter 4

A Study on Replica Optimization Strategy in Data Grid Environments

4.1 Introduction

A grid is large scale resource sharing and problem solving mechanism in virtual organizations [Fos01]. Large number of computational and storage resources are linked globally to form a grid. In some scientific application areas such as high energy physics, bioinformatics, and earth observations, we encounter huge amounts of data. People expect the size of data to be petabyte scale in some applications [Hos00]. Managing such huge amounts of data in a centralized manner is almost impossible due to extensively increased data access time. Data replication is a key technique to manage large data in a distributed manner. By its nature, we can achieve better performance (access time) by replicating data in geographically distributed data stores. In data grid, user's jobs require access to large number of files. If the required files are replicated in the site in which the job is executed, job is able to process data without a communication delay. However, if required files are not in the site, they should be fetched from other sites. Data fetch takes very long time because size of single replica may reaches giga-byte scale in some applications and network bandwidth between sites is limited. In result, job execution time becomes very long due to delay of fetching replicas over Internet. Dynamic replication is an optimization technique which aims to maximize chances of data locality. In other words, dynamic replica optimizer running in a site tries to locate files which are likely to be requested in the near future. As the number of file hit ratio increases, job execution time reduces significantly. Various dynamic replication strategies have been introduced so far.

In this chapter, we propose novel dynamic replication strategy; called BHR (Bandwidth

Hierarchy based Replication). The existing replication strategies try to maximize locality of the file to reduce data access time. However, grid sites may be able to hold only small portion of all files since very large quantity of data is produced in data grid and the storage space in a site is limited. Effect from this locality is limited to a certain degree. BHR strategy takes benefit from other form of locality, called network-level locality. Although the required file is not in the site performing job, there will be not long delay of fetching replica if the replica is located in the site having broad bandwidth to the site of job execution. We call this condition as network-level locality. In data grid, some sites may be located within a region where sites are linked closely. For instance, a country can be referred to as this network region. Network bandwidth between sites within a region will be broader than bandwidth between sites across regions. Thus, hierarchy of network bandwidth may appear in Internet. When the required file is located in the same region, less time will be consumed to fetch it. In other words, benefit of network-level locality can be exploited. BHR strategy reduces data access latency by maximizing this network-level locality.

Our BHR approach is currently evaluated using data grid simulator, OptorSim [Optor], initially developed as part of EU Data Grid project. We plug in our strategy in OptorSim and perform experiments with various data grid scenarios. The results from the experiments show that BHR strategy successfully reduces data access time in comparing to other strategies when hierarchy of bandwidth apparently appears.

4.2 Related Works

Dynamic replication is a long-term optimization technique which aims at reducing average job execution time in data grid. Since very large quantity of data files are deployed in data grid, there will be certain limitation of amount of files which can be stored at each site. If SE (Storage Element) at a grid site is already filled up with replicas, some of them should be deleted in order to store newly requested data.

Kavitha Ranganathan et al. present various traditional replication and caching strategies and evaluate them from the perspective of data grid in [Ran01]. They measure access latency and bandwidth consumptions of each strategy with simulation tool and their simulation results show

that Cascading and Fast Spread perform best among traditional strategies.

Economy based replication is proposed in [Bel03, Car02]. In economic approach, a kind of auction protocol is used to select the best replica for a job and to trigger long-term optimization (dynamic optimization) by using file access patterns. The authors show the improvement compared to traditional replication techniques by performing simulation with OptorSim. OptorSim is a data grid simulation tool developed as part of European Data Grid Project [Optor]. General data grid scenarios are modeled in OptorSim and one can evaluate various replication strategies implemented in it.

The existing replication techniques mentioned above are based on file access pattern at each site. If a grid site requests some files more frequently than others, it is better for the site to hold these files for near future usage. Even though this site-level locality can reduce data access time to some extent, there remains limitation. Performance gain from site-level locality can make sense when grid sites have enough space to store large portion of data and certain predictable file access patterns come out. However, we cannot assure in many cases that a single grid site will have enough space to store large portion of whole data and there will be predictable file access patterns. We find another key to the performance improvement by broadening our view of locality to the network level.

4.3 Dynamic Replication Strategy based on Internet Hierarchy

In this section, we propose novel dynamic replication strategy, called BHR, which is based on bandwidth hierarchy of Internet. Our BHR strategy takes benefit from network-level locality of files. The idea of proposed strategy is motivated from the assumption that hierarchy of bandwidth appears in Internet. Fig. 4-1 shows this assumption.

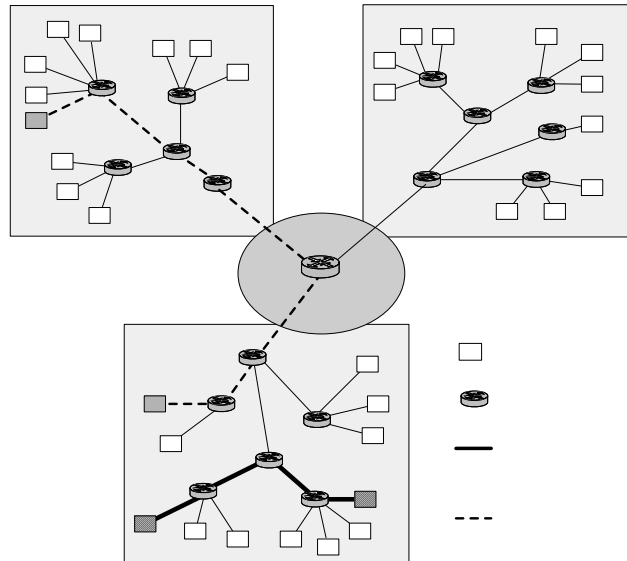


Fig. 4-1. Bandwidth hierarchy in Internet

We assume that group of sites are located on the same network region. A network region is a network topological space where sites are located closely. This network region can be seen as an Internet topology within a country. It is generally known that lower bandwidth can be allocated for network link between sites across countries than link between sites within a country. In many cases, network region may be usually correspondent to geographical space like a country or a continent. If the required replica is in the same region, the job is able to fetch the replica easily since broader bandwidth can be provided within a region. In contrast, if the required replica is located at the site in other region, much time will be consumed to fetch this replica via many links including highly congested one. Thus, a form of locality emerges which we call network-level locality. Main purpose of BHR strategy is to maximize this network-level locality within job execution model in data grid. BHR tries to replicate files which are likely to be used frequently within the region in near future. BHR optimizer runs both on a region and on a site cooperating with each other. Fig. 4-2 describes detail of BHR algorithm.

The access frequency gathered by region-optimizer means number of file requests made by jobs run on the sites within a region. It reflects regional popularity of files. If the job fetches a file from other sites and the SE is already filled up with replicas, we should determine whether storing newly received file is beneficial. If it turns out to be profitable, then we choose a file that should

be deleted in order to store new replica. We apply 2-step decision process. First one is avoiding duplication. The procedure 4 in Fig. 4-2 locates variety of replicas as many as possible in the region without duplication. Secondly, we take account of popularity of files as represented by procedure 5. In data grid, there can be popularity of file accesses, that is, certain files will be requested more frequently than others by grid job. While the previous strategies consider popularity of files at the site level, we focus on access popularity at the region level. BHR replaces unpopular files from the regional point of view. By applying above two steps, chance of hitting network-level locality can be maximized.

```

BHR Optimizer (region):
  Keep track of names of stored files and access frequency of each file within the region;
BHR Optimizer (site):
1.  if (needed replica not in a site)
    Fetch replica from other site;
2.  Proceed to execute the job with the replica;
3.  if (free space in SE to store new replica)
    Store it;
    else {
4.      if (new replica is duplicated in other sites within region) {
        Terminate optimizer; // avoid duplication
      }
      else {
        Sort files in SE in the order of less frequently accessed;
        for (each file in SE) {
          if (file is duplicated in other sites within region)
            delete it;
          if (enough free space to store new replica)
            break;
        }
      }
5.      if (!enough free space) {
        Sort files in SE in the order of less frequently accessed;
        // it's based on the access history gathered by region-optimizer
        for (each file in sorted list) {
          if (access frequency of new replica > access frequency of the file)
            delete file;
          if (enough free space)
            break;
        }
      }
    }
    if (enough free space)
      Store new replica;
  }

```

Fig. 4-2. BHR replication algorithm

4.4 Experiments

4.4.1 Simulation Tool

We evaluate the performance of BHR by implementing it in OptorSim, a data grid simulator developed to test dynamic replication strategies [Optor]. In OptorSim, general job execution scenario for data grid is modeled and various dynamic replica optimizers are implemented to test their effectiveness. After jobs are distributed to grid sites through Broker, they run on CE (Computing Element) at each site. Each job in CE has list of required replicas. For the first phase of replica optimization, Optimizer selects the best site to fetch the replica based on the available network bandwidth between sites. Then, Optimizer performs the second phase of optimization (dynamic optimization) by deciding whether storing (replicating) fetched file is beneficial or not.

4.4.2 Configuration

We perform simulation with assumed grid network topology and job execution scenarios. Fig. 4-3 describes the network topology assumed in our simulations. We assume there are 4 regions and each region has 13 sites on the average. File transfer time is decided according to the narrowest bandwidth along the path to the destination. Broader bandwidth can be provided between sites within a region whereas bandwidth between sites across region is relatively narrow. Since many sites within a region try to fetch files from other region through single inter-region link, this inter-region link is highly congested with network traffic and it causes hierarchy of bandwidth.

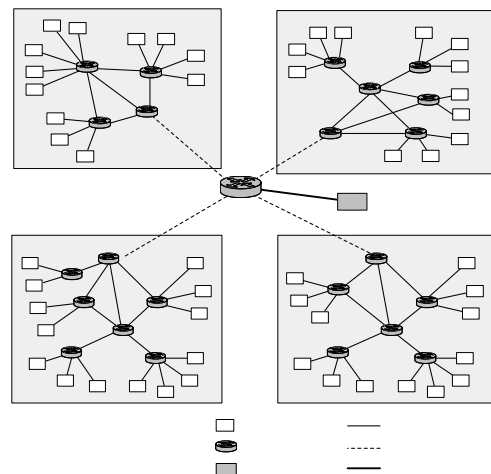


Fig. 4-3. Grid topology in simulation

In data grid environment, various job execution scenarios will be present. We try to apply general job execution scenarios as presented by Table 4-1. In the simulation, 1000 jobs are distributed to grid sites through broker. According to the file set each job accesses, we classify jobs into 50 job types. Each file set consists of 15 files. We assume that certain preference of job types appears. This preference of job type makes popularity of files. Each job sequentially requests access to files in a file set. There is no overlap between file sets each job type accesses and the size of single file is 1 GB. Therefore, total size of data in this configuration is 750 GB (50 job types * 15 files in a file set * 1 GB for each file). We assume all files are initially held at master site. Replication takes place after jobs start to execute at each site.

Table 4-1. General configuration of parameters

Parameters	Values
Number of jobs	1000
Number of job types	50
Number of file accessed per job	15
Size of single file	1 GB
Total size of files	750 GB

4.4.3 Results

We compare the performance of BHR with site-level file replacement schemes, LRU Delete and Delete Oldest. In LRU Delete, the least recently accessed file is chosen for deletion whenever replacement takes place. Delete Oldest is another replacement-based scheme which deletes the oldest file in SE first when newly required replica is received and replacement is necessary. In order to easily interpret the result, we assume that all network links within region (intra-region) show same bandwidth. And also all inter-region link bandwidths are assumed to be the same in the scenario. Initially, we roughly set the bandwidth and storage space as shown in Table 4-2. We set the bandwidth between master site and its adjacent router as 2000 Mbps. It is much broader than other links to avoid effects from network traffic congestion at master site.

Table 4-2. Bandwidth and storage spaces

Parameters	Values
Intra-region bandwidth	1000 Mbps
Inter-region bandwidth	1000 Mbps
Master-router bandwidth	2000 Mbps
Storage space at site	50 GB

Fig. 4-4 shows the achieved results with initial parameters. BHR takes the least total job execution time among strategies. It takes 33,174 seconds which is about 30 % less than other strategies. Since size of SE at each site, 50 GB, is not enough to hold large portion of overall data (750 GB), we cannot achieve much performance improvements with site-level replacement schemes. However, BHR strategy takes benefit from network-level locality by locating variety of files in a region as many as possible. Also, it locates files which are likely to be used in the region based on regional access history. In this simulation, Delete LRU and Delete Oldest show almost the same job execution time. The reason is that we do not assume any specific file access pattern in the data grid system.

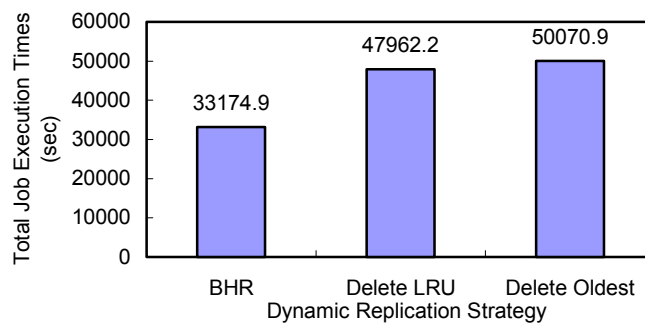


Fig. 4-4. Total job times with parameters shown in Table 4-1 and Table 4-2

We continue performance evaluation with varying bandwidths and storage spaces. As we increase the inter-region bandwidth, file transfer via inter-region link does not take long although it is highly congested with data traffics. Thus, hierarchy of bandwidth becomes indistinct. Here we show results with varying bandwidths in Fig. 4-5 (a). When we set narrow bandwidth on the inter-region link, BHR outperforms other strategy considerably. However the differences of job execution time become smaller as broader inter-region bandwidth is set. Finally, the difference becomes negligible when more than 1800 Mbps is provided for inter-region bandwidth. We can conclude that BHR strategy can be effectively utilized when hierarchy of bandwidth appears apparently. Size of SE in a grid site also affects the result significantly. As we mentioned, traditional replacement-based scheme can be effective when large storage space is provided in a grid site. In Fig. 4-5 (b), as the size of storage space decreases in grid sites, BHR outperforms other strategies greatly. However, as the storage size increases, job execution time of two

replacement-based schemes reduces sharply. The reason is that the file hit ratio in a site increases when large number of replicas can be stored in a site, and regional file hit ratio also increases though no region-based optimization strategy is applied. After all, efficiency of all three strategies become almost the same when large quantity of storage is provided at a site. Our BHR strategy can be more effective when grid sites have relatively smaller storage. One may argue that 100 GB is not an impractical storage size for a grid site. However, enough size of storage which makes the site-level replacement schemes effective is relative to the total size of data in a data grid system. In this simulation, only 750 GB of data is assumed to be in data grid while, in practice, terabyte or even petabyte scale of data is expected to be common in data grid.

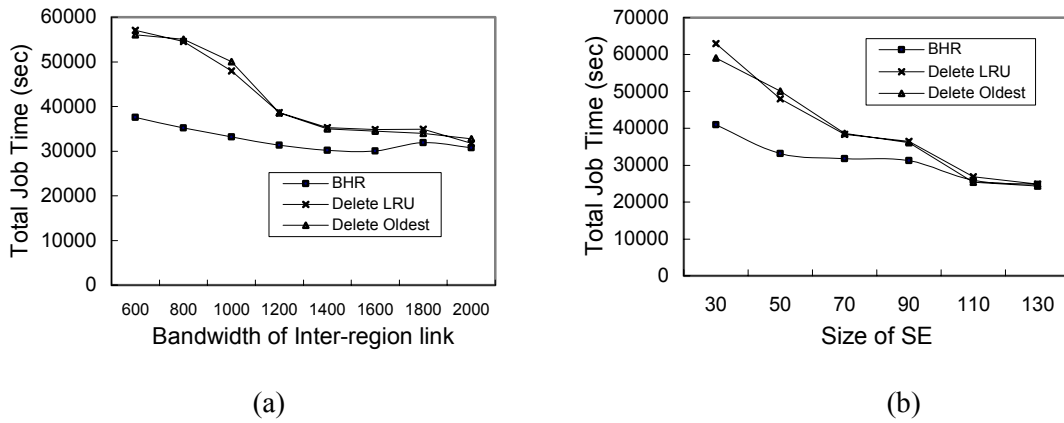


Fig. 4-5. Total job time with varying bandwidth and storage size

4.5 Conclusions

In this chapter, we propose novel dynamic replica optimization strategy which is based on the network-level locality. BHR tries to replicate popular files as many as possible within a region, where broad bandwidth is provided between sites. The simulation results show that BHR takes less job execution time than other strategies especially when grid sites have relatively small size of storage and hierarchy of bandwidth clearly appears. BHR extends current site-level replica optimization study to more scalable way by exploiting network-level locality.

Chapter 5

Conclusions

In this thesis, I have presented several novel schemes to enhance performance of grid applications in data and mobile grid environment. These schemes include scheduling algorithms in both data and mobile grid environment, and the dynamic replication strategy in a data grid environment. Summary of this thesis can be categorized with respect to the chapters:

- In chapter 2, I addressed scheduling mismatch problem in Data Grid environment. I proposed scheduling models that consider both computational capability and location of data replica. The model consists of five scheduling scenarios that may occur in real situations. I have identified cost model for each scenario. I also implemented Grid scheduling tool, called Chameleon, based on the proposed models. I performed meaningful experiments on a Grid testbed and could achieve better performances by applying Chameleon to the gene sequence comparison applications.
- In chapter 3, I discussed on the issue of integrating mobile devices into grid. I proposed the novel view of the mobile/grid integrated system where a gateway links the static grid sites to the group of mobile devices, thus mobile users can make use of static grid resources, and also provide their mobile devices as grid resources. I presented the newly emerging technical issues for realizing this mobile grid system, and particularly focused on the job scheduling algorithm to achieve more reliable performance.
- In chapter 4, I proposed novel dynamic replica optimization strategy which is based on the network-level locality. The proposed algorithm, BHR, tries to replicate popular files as many as possible within a region, where broad bandwidth is provided between sites. The simulation results show that BHR takes less job execution time than other strategies especially when grid sites have relatively small size of storage and hierarchy of bandwidth

clearly appears. BHR extends current site-level replica optimization study to more scalable way by exploiting network-level locality.

I believe that the schemes presented in this thesis could contribute to the data and mobile grid communities in several respects, and hope that more enhanced techniques would be derived from my contributions.

References

- [Abr00] D. Abramson, J. Giddy, I. Foster, 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, May 2000.
- [All01] 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.
- [Antz] ANTZ: <http://www.antz.or.kr>
- [Apgrid] ApGrid: <http://www.apgrid.org>
- [Bak00] 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.
- [Bel02] William H. Bell, David G. Cameron, Luigi Capozza, A. Paul Millar, Kurt Stockinger, and Floriano Zini. "Simulation of Dynamic Grid Replication Strategies in OptorSim," In Proc. of the 3rd Int'l. IEEE Workshop on Grid Computing (Grid'2002), Baltimore, USA, November 2002. Springer Verlag, Lecture Notes in Computer Science.
- [Bel03] William H. Bell, David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Kurt Stockinger, and Floriano Zini. "Evaluation of an Economy-Based File Replication Strategy for a Data Grid," In International Workshop on Agent based Cluster and Grid Computing at CCGrid 2003, Tokyo, Japan, May 2003. IEEE Computer Society Press.
- [Ber97] F. Berman and R. Wolski. "The AppLes project: A status report," Proceedings of the 8th NEC Research Symposium, Berlin, Germany, May 1997.
- [Bri02] Clarke, Brian and Marty Humphrey. "Beyond the 'Device as Portal': Meeting the Requirements of Wireless and Mobile Devices in the Legion Grid Computing System," In Proceedings of the Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing at the International Parallel and Distributed Processing Symposium. IEEE Press, 2002.
- [Buy02] 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.
- [Car02] Mark Carman, Floriano Zini, Luciano Serafini, and Kurt Stockinger. “Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid,” In International Workshop on Agent based Cluster and Grid Computing at International Symposium on Cluster Computing and the Grid (CCGrid'2002)., Berlin, Germany, May 2002. IEEE Computer Society Press.
- [Cern] CERN DataGrid Project: <http://www.cern.ch/grid/>
- [Che01] 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.
- [Condor] The Condor Project. <http://www.cs.wisc.edu/condor>
- [Cza98] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke. “A Resource Management Architecture for Metacomputing Systems,” *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pg. 62-82, 1998.
- [Dul01] 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.
- [Eudata] EU Data Grid Project: <http://www.eu-datagrid.org>
- [Fos99] I. Foster and C. Kesselman. “The Grid: Blueprint for a New Computing Infrastructure,” Morgan Kaufmann, 1999.
- [Fos01] I. Foster, C. Kesselman and S. Tuecke. “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” *International J. Supercomputer Applications*, 15(3), 2001.
- [Gib01] Cynthia Gibas. “Developing Bioinformatics Computer Skills,” O'REILLY, April 2001.
- [Globus] The Globus Project: <http://www.globus.org>
- [Gou00] J.-P Goux, S. Kulkarni, J. T. Linderoth, and M. E. Yoder. "An Enabling Framework for Master-Worker Applications on the Computational Grid," *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, 2000.

- [Guy02] 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.
- [Hos00] 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.
- [Hol01] Koen Holtman. "CMS Data Grid System Overview and Requirements," CMS Note.
- [Kon02] Derrick Kondo, Henri Casanova, Eric Wing, Francine Berman. "Models and Scheduling Mechanisms for Global Computing Applications," Proceedings of IPDPS 2002, April 15 - 19, 2002, Fort Lauderdale, California
- [Led03] Jonathan Ledlie, Jeff Shneidman, Margo Seltzer, John Huth. "Scooped, Again," Proceedings of Second International Workshop on Peer-to-Peer Systems (IPTPS'03), 20-21 February 2003 in Berkeley, CA, USA.
- [Lsf] LSF. <http://www.platform.com/products/LSF/>
- [Openpbs] OpenPBS. <http://www.openpbs.org>
- [Optor] OptorSim – A Replica Optimizer Simulation: <http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html>
- [Par03] Sang-Min Park and Jai-Hoon Kim. "Chameleon: A Resource Scheduler in a Data Grid Environment," 2003 IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'2003), Tokyo, Japan, May 2003.
- [Pha02] T. Phan, L. Huang, and C. Dulan. "Challenge: Integrating Mobile Wireless Devices Into the Computational Grid," In Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom '02), September 25-27, 2002, in Atlanta, GA.
- [Rad99] S. Radhakrishnan, N. S. V. Rao G. Racherla, C. N. Sekharan, and S. G. Batsell, "DST - a routing protocol for ad hoc networks using distributed spanning trees," IEEE Wireless Communications and Networking Conference, pp. 100-104, 1999.
- [Ran01] 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.
- [Ran01] Kavitha Ranganathan and Ian Foster. "Identifying Dynamic Replication Strategies for a

- 'High Performance Data Grid," International Workshop on Grid Computing, Denver, November 2001.
- [Ran02] 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.
- [Sat96] M. Satyanarayanan. "Fundamental Challenges in Mobile Computing," In Proceedings of the fifteenth annual ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, 1996.
- [SETI] SETI@home. <http://setiathome.ssl.berkeley.edu>, March 2001.
- [Sto01] 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.
- [Sto02] Heinz Stockinger, Asad Samar, Bill Allcock, Ian Foster, Koen Holtman, and Brian Tierney. "File and Object Replication in Data Grids," Journal of Cluster Computing, 5(3):305-314, 2002
- [Vaz01] 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.
- [Wil02] Bryce Wilcox-O'Hearn. "Experiences Deploying a Large-Scale Emergent Network," Proceedings of Second International Workshop on Peer-to-Peer Systems (IPTPS'02), 7-8 March 2002 in Cambridge, MA, USA.
- [Wol99] 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.