

## Strategies Toward Optimal Access to File Replicas in Data Grid Environments\*

KUAN-CHING LI, HSIAO-HSI WANG<sup>1</sup>, KUO-YANG CHENG<sup>2</sup> AND TSUNG-YING WU<sup>2</sup>

*Department of Computer Science and Information Engineering*

<sup>1</sup>*Department of Computer Science and Information Management*

*Providence University, 433 Taiwan*

*E-mail: {kuancli; hhwang}@pu.edu.tw*

<sup>2</sup>*National Center for High-Performance Computing*

*Hsinchu, 300 Taiwan*

*E-mail: {kuoyang; alex}@nchc.org.tw*

Data Grid is one of key technologies to build up large-scale dataset storage system by connecting scattered storage resources dispersedly located in the Grid. One major challenge in data grids is how to provide good and timely access to huge amount of data in distributed locations, given the high latency of interconnection networks. Parallel downloading methods can improve download efficiency and performance, and for such, processes should have started from appropriate locations. In this research paper, we present the design framework of PU-DG Optimizer toolbox (also known as PU-DG Optibox) for data grid environments. The proposed toolbox is a package containing a number of high-end techniques and algorithms running as middleware on top of data grid platforms, in order to optimize file downloads, by improving its efficiency and performance. Moreover, PU-DG Optibox not only provides users and developers possibilities for setting their own priority strategies, as also different downloading modes. Furthermore, workload balancing to avoid low quality computing node to execute highly complex job is also included in this design. Experimental results of techniques packaged in the proposed toolbox demonstrate its potential and effectiveness.

**Keywords:** data grid, download optimizer toolbox, file replica selection, replica placement, hierarchy-based list

### 1. INTRODUCTION

Grid is evolving rapidly as a practical extension to distributed computing technology, with the vision of dynamic and diverse resource sharing across organizations [1]. Resource sharing over a grid improves the utilization of existing computing and storage resources in cost-effective manner, providing alternatives to solve large-scale problems requiring an enormous amount of computing power and storage, with the utilization of these resources coordinately. Grid resources are widely distributed and managed under different local policies and domains, which can be highly diverse and their availabilities can change dynamically.

An important concept of Grid is Virtual Organization (VO) [10], which basically involves in the combination of geographically distributed resources and organizations such personal computers, clusters of workstations, supercomputers, government, and

---

Received February 1, 2008; accepted November 28, 2008.

Communicated by Yau-Hwang Kuo, Pau-Choo Chung and Jar-Ferr Yang.

\* This paper was based upon work supported in part by National Science Council of Taiwan, R.O.C. under grants No. NSC 95-2221-E-126-006-MY3, NSC 96-2221-E-126-004-MY3 and NSC 95-2745-E-126-002-URD.

business, among others. A Grid middleware is therefore needed, to hide the heterogeneity of the computing resources and to bridge the gaps between different local policies and management. Resources may not be stable in the Grid, and therefore, VOs change resources dynamically, since failures and replacements of computing or storage facilities in these organizations happen without notice.

Data Grids are becoming increasingly useful for managing large-scale distributed data across multiple heterogeneous resources. Examples in utilization of data grids can be found, for example, in physics community, biomedical applications, astronomy, bioinformatics, Earth systems science, ecology, education, geology, geography, earthquake and plate tectonic systems. Not only for their large-scale data integrations need by their nature, to have these diverse applications highlighted their various functionalities through data grids [10-12]. In particular, data grids show their large data handling capability, data aggregation and movement features, parallel data transfer functionality, collaborative strengths and metadata management facilities inherent in a data grid system.

Traditional data transmission and storage technologies are not well suited for data grid environments. Therefore, a number of technologies focused on this drawback have been developed, such as Storage Resource Broker (SRB) [6] and GridFTP [2, 4, 5] of Globus Toolkit [2]. Systems as Storage Resource Broker (SRB) integrate heterogeneous storage devices to a unified view and provide a platform to manage user, file location, storage resources and metadata information.

Traditional download methods consist of single and multiple nodes. In case of multiple nodes, parallel data download can be activated, since it has more than two replicas stored in data grid [3, 9, 12-15]. A number of technologies are available to generate replicas, in order to improve system's download efficiency. For instance, affinity replica location policy places replicated data on or near local site to decrease the network latency and download time [3]. Unfortunately, this policy causes network load problems. If affinity sites are overloaded, users in the queue get into the waiting queue, which seriously affect the network latency and transmission efficiency. The proportional share replica location policy [3] avoids the network overload and support concepts of balancing sites in terms of network load, to upgrade transmission efficiency and at the same time to avoid the affinity sites overload, which unfortunately may lead to shutting down of affinity sites.

Not only upgrading transmission efficiency and decreasing network latency, these replicas may integrate parallel download methods to improve download efficiency and performance of overall system. Although parallel download methods can upgrade download efficiency and improve performance, a ranked list of download sites must be provided with higher quality locations for such download. In addition, to include a number of sites worse ranked ("with lower quality") might affect seriously the performance of a system. Therefore, the issue of choosing a set of appropriate locations for download is crucially important in data grid, in the effectiveness point of view. In this paper, we propose PU-DG Optibox to optimize file downloading by ranking best and appropriate locations.

The structure of this paper is as follows. In section 2, we introduce the background of data grid, whilst in section 3 we present PU-DG Optibox's complete framework and its components. In addition, the algorithm designed and complexities involved are also discussed. In section 4, methods components of the PU-DG Optibox are evaluated, in

order to demonstrate the effectiveness of such toolbox acting as middleware in Data Grids. Finally, some conclusion remarks and future work are given in section 5.

## 2. BACKGROUND

The term “Data Grid” traditionally represents the network of distributed storage resources, from archival systems, to caches, to databases, that are linked using a logical name space to create global, persistent identifiers and provide uniform access mechanisms. Data grids are becoming increasingly important and playing important role in scientific communities for sharing large data collections and for archiving and disseminating them in a digital library framework. A transparent middleware for sharing data across distributed, heterogeneous data resources separated by different administrative and security domains is needed to archive data sharing and data management.

A number of facilities must be inherent in this middleware, to provide good resource utilization and application response time. These facilities can be listed as: large data handling capability, data aggregation and movement features, parallel data transfer functionality, collaborative strengths and metadata management facilities. In addition, it must provide a transmission platform to communicate with distributed storage resources such as file systems database systems, High Performance Storage System (HPSS), among others.

Data grids are used to combine with grid computing to decrease file transmission time, and therefore improve the overall grid computing performance. Grid computing is continuously received huge amount of requests from distributed users through grid platform. Those requests will be processed by resource broker. Resource broker collects all of requests and according to job scheduling algorithm to scatter to specific computing nodes in grid environments. Job scheduling algorithm according to specific method to assign jobs, for instance random, shortest queue size, access cost and queue access cost [6-8]. Details of job scheduling algorithms are as follows:

- Random scheduling randomly selects a computing node to execute specific job,
- Shortest Job Queue scheduling calculates all of the queue length of computing nodes and schedule computing node with the shortest job queue to run the job,
- Access Cost scheduling submit the job to specific computing node with the lowest file access cost,
- Queue Access Cost scheduling selects the computing node with the lowest of access cost and queue size to execute the job.

In addition, a job consists of set of files that access file sequential to affect efficiency of execution. The choice of access patterns for resource broker are sequential, random, Gaussian random walk, Gaussian distributed, Zipf-like distribution [6-8].

In order to upgrade the efficiency of job execution, it is important to create a file replica in specific site local storage to decrease the file transmission time [3, 9, 12-15]. Many strategies are discussed about placement of file replicas that is created on suitable location. The advantage of file replicas can reduce transmission time of download and decrease the network workload. The file replica can be created to affinity sites that are

able to support the huge amount of users to download. Due to the sites network load quickly increase to lead to quality of network seriously decrease. It is possible to seriously affect the benefit of file replica.

File replicas are continuously created into specific site that is possible to full all storage space. We delete the replica when storage space is full. We can adopt replication replacement strategies to remove one of the replications in storage system and then swap new replication into the storage system. Several replication replacement strategies that consist of No Replication, LRU and LFU *etc.* are proposed in the past [6-8]. Details of replication replacement strategies are as follows:

- No Replication strategy has not replica to be created and then files are always accessed remotely.
- LFU (Least Frequently Used) strategy always replicates files to local storage system that supports the computing node to use. If the local storage space is full, we remove the replica that has been accessed the fewest times and then release the space for new replica.
- LRU (Least Recently Used) strategy always replicates files to local storage system. If the local storage space is full, we delete the oldest replica for new replica.

These strategies consist of job scheduling algorithm, access pattern of job, replica selection, replica placement and replica replacement that improve efficiency of data download and upgrade the overall grid computing performance.

### 3. PU-DG OPTIBOX

#### 3.1 Overview

Different download sites may have distinct quality ranking, depending on three factors to generate a final grading, from which we evaluate the quality of download sites. These three factors consist of: (1) network bandwidth, (2) transmission distance and (3) history of download. We will discuss each of these factors independently.

Network bandwidth is the key factor in data grid environments. In the design of PU-DG Optibox algorithm, we classify this factor into three sub-factors, according to their network bandwidth. These sub-factors are classified in 10MB/s, 100MB/s and more than 100MB/s. It is quite obvious that, as network bandwidth increases, the transmission time is reduced. The quality ranking determines the sites which will succeed or not, and at which order this happens. It is important to note that quality of transmission is a fundamental element in network environments.

As second factor, the distance for transmissions affects seriously efficiency of download. As transmission distance increases, the transmission time will raises also. Therefore, we divide this factor into four sub-factors, according to the distance for such transmission. The four sub-factors consist of local sites, regional sites, national sites and international sites [3]. The transmission time of local sites are expected to be the shortest. On the contrary, international sites are expected to show highest transmission time, since the actual distance for such transmission is the longest. No matter the data is located in

local or international sites, the goals in improvement this factor are to reduce transmission time between sites, and to upgrade efficiency of network transmission in data grid environments.

Finally, PU-DG Optibox is designed to keep each user’s download back history. This history contains the following information: download time, download location and data transferred. We divide into three sub-groups according to download time history information. Since download time may affect the efficiency of download, download site local time (time difference in other countries) should be analyzed first. PU-DG Optibox’s objective is to provide most suitable locations for download at that particular download time. Past download location information can provide most suitable locations for download, from the user’s history in the system, so that system may refer to this information when performing downloads next time. User’s characteristics are classified according to his needs. If a user needs to download urgently specific set of data, PU-DG Optibox adjusts the deployment and upgrades the efficiency of download for that user at that particular moment. By referring to download history, we utilize methods in data mining to upgrade the efficiency of download as also improving the performance.

Traditional methods in existing middlewares provide solely the list of files to download, and users choose by themselves. In the proposed PU-DG Optibox system, it provides two modes for end users: manual mode and automatic mode.

In the former mode, PU-DG Optibox system ranks and lists the quality of downloading locations, according to priority defined by users themselves or system administrators. In manual mode, users can select favorite download locations that contains replica this user needs. Selecting the configuration settings correctly influence tremendously in the efficiency of download. In the latter, PU-DG Optibox automatic mode, this combines network bandwidth with distance and history of download to find out the best location for this particular download. Grid users only need to connect to PU-DG Optibox of a data grid environment. In addition, the PU-DG Optibox will support automatically service of data download on demand.

As conclusion, PU-DG Optibox system provides an optimal solution of download for different users’ workflow and needs. The automatic mode of our PU-DG Optibox can simplify the procedure of file download in data grid environments, and these can be observed in Fig. 1.

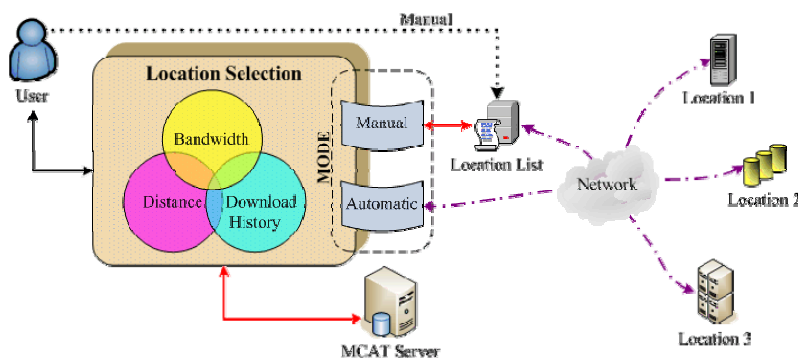


Fig. 1. Replica location with manual mode and automatic mode.

### 3.2 Components and Procedures

When a user needs to download specific file from a data grid platform, he must provide the download priority, as shown in Fig. 1. The PU-DG Optibox will find out a set of all replica locations that contains this file and rank these locations by priority, and then, it chooses best location for downloading this file. Emphasizing, these procedures are all “automatic mode” of PU-DG Optibox. Users do not need to understand nor follow complex instruction steps and detailed techniques to proceed in the file downloading process. The PU-DG Optibox will choose automatically best download locations for this user.

Fig. 2 illustrates main procedures and technical characteristics in details of the proposed PU-DG Optibox, that selects automatically best locations for a file download, that is, in terms of minimum downloading time. PU-DG Optibox is divided into three major components: Candidate Node Collection, Priority Sort, and Download Solution Selection, each handling different parts of “automatic” mode.

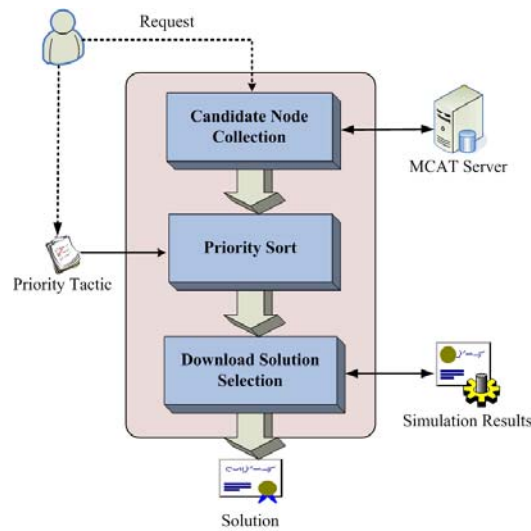


Fig. 2. Components of replica selection with automatic mode.

As the PU-DG Optibox toolbox receives the download request from user, server will invoke first component “Candidate Node Collection” and query the MCAT database, to collect the list of replica locations. Other information about the file or connection network is also collected in this phase. For instance, the file size, location, bandwidth, distance, *etc.*

These download information are collected into a location list, ranked and input to the second component “Priority Sort”. This “Priority Sort” component will re-rank the location list based on priority constraints provided by the user. There are three main factors are considered in our PU-DG Optibox, which are bandwidth (B), distance (D), and history record (H). Therefore, the priority tactic has totally six different considerations: BDH, BHD, DBH DHB, HBD, and HDB, in which the user can choose and one to be the priority tactic (or constraint).

For instance, if a user selects “BDH” as his priority tactic, PU-DG Optibox will sort the location list, looking the bandwidth (B) first and ranks in decreasing order all locations that contains that replica file, and classify these locations into  $k$  groups. For each group, sort the locations within the same group in order of increasing the distance and classify into  $n$  group, and put the locations which were history records for this user on the top of each group. Finally, the location list can be sorted by BDH tactic and classified into  $2kn$  groups. The proposed PU-DG Optimizer toolbox sorts the location list, according to the selection made by users in this phase. It is useful for us to select a better location set for file download for different needs among different users. This new location list will be forward to third components to select the set of download locations.

This component “Download Solution Selection” is important, in the sense that it will select best locations among sorted location list, to finally download the selected file. This phase can be partitioned into three steps. First, in order to decide efficiently, we use simulation to get more results of file download for various properties of file or status of network. We choose an initial location set on the location list according to these simulation results. As second step, we calculate the download time of location set chosen in the first phase, according to the status of network and file size. The download time can be calculated as show next in Eq. (1).

$$f(n) = \text{Max}_{i=1}^n \left[ \frac{(T/n)}{S_i} \right] + O_n \quad (1)$$

where:  $T$  is the total size of file.

$S_i$  is the transfer speed from site  $i$ .

$O_n$  is the overhead for file download from  $n$  sites.

$f(n)$  means the download time to download a file from top  $n$  replica sites from the list of locations. Since PU-DG Optibox can download file in parallel, we utilize the maximum download time to be the final result. The formula  $f$  can be transformed into Eq. (2) using dynamic programming, since there is recurrence relation of replica sites.

$$f(n) = \text{Max} \left\{ \frac{[f(n-1) - O_{n-1}] \times (n-1)}{n}, C_n \right\} + O_n \quad (2)$$

where  $C_n$  is the download time for downloading  $T/n$  size of file, from the  $n$ th replica site in the list of locations. Therefore, the formula  $C_n$  is shown next in Eq. (3).

$$C_n = \frac{(T/n)}{S_n} \quad (3)$$

We can use Eq. (2) to calculate the download time  $f(n)$  using top  $n$  sites in constant time, if download time  $f(n-1)$  of top  $n-1$  sites is already calculated, and the total time complexity is  $O(n)$  to calculate  $n$  candidate replica sites.

In the third phase, we adjust our selection on this location list, in order to try to obtain less download time from other cases. We compare download time of fewer and more replica sites with initial number of locations and to adjust the number of location list to

find out a better location for download. If we can find out locations with lower download cost, the PU-DG Optibox starts to download file according this download automatically. The algorithm for the implementation of PU-DG Optibox is shown in section 3.3.

### 3.3 Algorithm

In this section, we show the algorithm of PU-DG Optibox and discuss the detail of key components. These key components consist of Priority Sort and Download Solution Selection. The Algorithm of Replica Selection in PU-DG Optibox, as shown in Fig. 3.

```

LOCSELECT( $R, P, F$ )
Input: Replica sites  $R$ , sort priority  $P$ , and file properties  $F$ .
Output: The replica list  $L(R)$  and the number of sites for downloading this file.
1  $L(R) \leftarrow \text{BUCKETSORT}(R, P)$   $\triangleright L(R) = \langle r_1, r_2, \dots, r_n \rangle$  is the result of bucket sort.
2  $k \leftarrow \text{SIMRESULT}(F)$   $\triangleright$  Find the initial number of sites by searching simulation results.
3  $f(1) \leftarrow [T \div s_1] + O_1$   $\triangleright$  initial score.
4 for  $i \leftarrow 2$  to  $k + 1$  do  $\triangleright$  visit  $r_{k-1}, r_k,$  and  $r_{k+1}$  in  $L(R)$ .
5    $f(i) \leftarrow \max \left\{ \frac{[f(i-1) - O_{i-1}] \times (i-1)}{i}, C_i \right\} + O_i$ 
6 if  $f(k+1) < f(k) < f(k-1)$  then  $\triangleright$  Case adjustment.
7    $i \leftarrow k + 1$ 
8   do
9      $i \leftarrow i + 1$ 
10     $f(i) \leftarrow \max \left\{ \frac{[f(i-1) - O_{i-1}] \times (i-1)}{i}, C_i \right\} + O_i$ 
11    while  $f(i) < f(i-1)$  and  $(i < n)$ 
12    return  $i - 1$  and  $L(R)$ 
13 else if  $f(k+1) > f(k) < f(k-1)$  then
14   return  $k$  and  $L(R)$ 
15 else
16   for  $i \leftarrow k - 1$  to 1 do
17     if  $f(i-1) > f(i)$  then
18     return  $i$  and  $L(R)$ 

```

Fig. 3. Algorithm of replica selection in PU-DG Optibox.

The Priority Sort of PU-DG Optibox, we utilize Bucket Sort mechanism to sort the initial location list. In total, we need to execute Bucket Sort three times, if we have three priority factors for users, and the total time complexity of priority sort phase is  $O(n)$ . We make use of file properties to search the simulation results and obtain an initial number of replica sites  $k$  for download. Then, we calculate the download time of top  $k + 1$  replica sites in location list, where we can obtain the time of  $f(k - 1)$ ,  $f(k)$ , and  $f(k + 1)$ .

We compare value of priority sort with simulation result. If the value of priority sort is more than simulation result, three cases are considered in Table 1. According to simulation result, we attempt to adjust the parallel downloading sites to find out the best

**Table 1. The key factors of algorithm.**

Priority Sort $\geq$ (Simulation Result = $k$ )		
Case 1	If $f(k+1) < f(k) < f(k-1)$	$k++$ ; Until $f(k+1) > f(k)$ ; Return $k$ ;
Case 2	If $f(k+1) > f(k) < f(k-1)$	Return $k$ ;
Case 3	If $f(k+1) > f(k) > f(k-1)$	$k--$ ; Until $f(k-1) > f(k)$ ; Return $k$ ;

number of parallel downloading sites for decreasing download time. Otherwise, if the value of priority sort is less than simulation result, we adopt the value of simulation result to parallel download this file.

The Download Solution Selection of PU-DG Optibox, where the set of replica sites is adjusted and attempted to find out a better download solution for this specific file, according to user's selection. Since this location list is sorted according to the priority tactic defined by the user when he initialized the system, replica sites on top of this list are most suitable for users. Finally, this algorithm will return a location list with priority sorted  $L(R)$  and the appropriate number for download from replica sites. Then, we can make use of data grid environment to parallel download this file from the list of replica sites, so finally merge into a complete file.

The time complexity is  $O(n)$ , what proves its viability and timely interesting to implemented the proposed algorithm over the Data Grid, making this PU-DG Optibox coupled with Data Grid environment, to globally improve the efficiency of file downloading.

### 3.4 Min-Max Balancing Workload Method

In order to decrease file transmission time, we must find out the best network bandwidth for grid computing. Therefore, we take into account three factors that consist of network bandwidth, transmission distance and history of download given above. Therefore, according to the three factors can find out the best location to download specific file. In addition to find out the best location, many factors must be considered in grid computing environments. For instance, these factors could consist of replica placement, replica replacement and job scheduling, job interval and size of queue that are important issues about grid computing.

We propose the method to find out the best location for grid computing to deliver the specific file between storage systems. However, we attempt to adopt our method in grid computing environments and we submit huge amount of jobs to computing nodes. Each job consists of the set of files, but it is possible to be stored in distributed locations. In order to execute a job, we must look up the requirement files for job in grid environments. Therefore, we adopt our method to find the best location to download the requirement files for job, so we can decrease file transmission time. We not only decrease file transmission time, but also upgrade efficiency of job execution.

In recent years, increasing number of users submit large amount of jobs of different and various sizes and types in grid environments. Resource broker places a job to computing node according to job scheduling algorithm, but various type of job adopts same scheduling algorithm to process, it is possible to lead to reduce the overall performance. If the high complexity of job is assigned to the low quality of computing node, the job execution time seriously increases.

We propose the concept of balancing workload to avoid that low quality computing node execute jobs of high complexity. We classify these jobs into two categories, in Min job and Max job, according to job size. Our method determines threshold of job size, if the job size exceed the threshold, the job is Max job. Otherwise, the job is considered Min job.

Next, we classify the computing nodes into two categories, in Min bandwidth and Max bandwidth, according to their network bandwidth. Our method determines the threshold of network bandwidth, and therefore, if bandwidth of computing node exceeds the threshold, the computing node is classified as Max node. Otherwise, the computing node is Min node.

Finally, resource broker submits a job to computing node, according to scheduling algorithm. That is, we schedule the Min job to Min node and Max job to Max node. In other words, we avoid the Min node to be assigned the Max job, what seriously affect the overall performance.

For instance, there are six jobs and have different size that consists of job1 = 1GB, job2 = 2GB, job3 = 4GB, job4 = 8GB, job5 = 16GB and job6 = 32GB. In addition, there are eight job execution sites and have different network bandwidth that consists of Site1 = 1Mb/s, Site2 = 2Mb/s, Site3 = 4Mb/s, Site4 = 8Mb/s, Site5 = 16Mb/s, Site6 = 32Mb/s, Site7 = 64Mb/s and Site8 = 128Mb/s. We determine the threshold of job size to be 10GB that is average of all of job sizes. The threshold of network bandwidth is 31Mb/s that is average of all of network bandwidth. Therefore, we classify all of jobs into two categories according to job size. These Min jobs consist of Job1, Job2, Job3 and Job4. Otherwise, these Max jobs consist of Job5 and Job6. These Min nodes consist of Site1, Site2, Site3, Site4 and Site5 and these Max nodes consist of Site6, Site7 and Site8. Finally, we assign these Min jobs to Min nodes and Max jobs to Max nodes. Therefore, we only assign these jobs that consist of Job5 and Job6 to Site6, Site7 or Site8. Otherwise, we submit these jobs that consist of Job1, Job2, Job3 and Job4 to job execution sites that consist of Site1, Site2, Site3, Site4 and Site5. In our method, we avoid the Min node to be assigned the Max job hence seriously affect the overall performance.

## 4. PERFORMANCE EVALUATION

### 4.1 Simulation Environment

We have selected simulator OptorSim [8] to evaluate proposed strategies and techniques. In particular, the set of experiments concerned to this paper, performance evaluation for Min-Max Balancing Workload strategy. It is possible to build a data grid platform using OptorSim, consisting of Resource Broker (RB), Replica Manager (RM), Replica Optimiser (RO), Computing Elements (CEs), and Storage Elements (SEs), which are essential elements to build the environment under investigation.

### 4.2 Simulation Results

Using PU-DG Optibox, we will find out the best job scheduling algorithm to upgrade the overall performance. We present the comparison of fours job scheduling algorithms

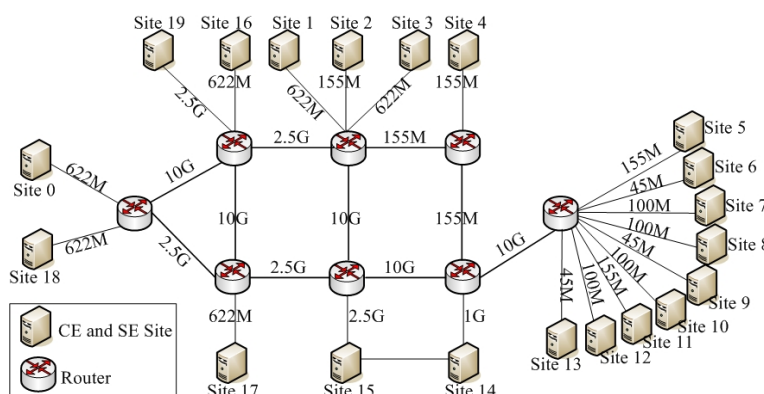


Fig. 4. The topology of PU-DG Optibox.

Table 2. Simulation parameters.

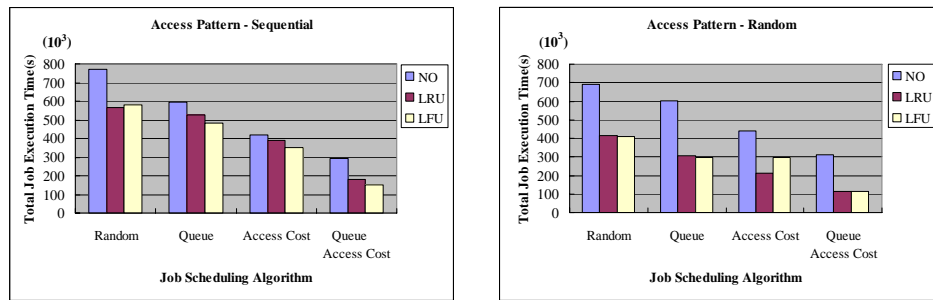
Parameter	Case 1	Case 2	Case 3
Number of Jobs	100	1000	10000
Time interval between Jobs (ms)	5000	5000	5000
Number of various Job sizes	8	8	8
Unit of file size (MB)	1000	1000	1000

in our simulation. We introduce our network topology and simulation parameters before we compare with these job scheduling algorithms. The network topology and simulation parameters are shown in Fig. 4 [7] and Table 2 with 100 jobs. In our simulation contains 28 sites that consist of 18 sites have CEs and SEs, 2 sites only have SEs and 8 sites have no CEs nor SEs acting as Routers. Site14 and Site19 initially own a storage capacity of 500GB each and a master copy of each file is stored at one of these two sites. Every site of 18sites owns a CE and a storage capacity of 50GB.

#### 4.2.1 Comparison of job scheduling algorithm

In our simulation, we compare with four job scheduling algorithms that consist of (1) Random, (2) Shortest Job Queue, (3) Access Cost and (4) Queue Access Cost. In order to find out the best job scheduling algorithms, we adopt various access patterns and replica replacement strategies to be able to completely simulate the real Data Grid environments. The total job execution time for various replica replacement strategies and comparison of job scheduling algorithms with access pattern of Sequential are shown in Fig. 5 (a). As Fig. 5 (a), the simulation results indicated that Queue Access Cost is the best because not only shortest job queue but also access cost is considered in Queue Access Cost.

The total job execution time for various replica replacement strategies and comparison of job scheduling algorithms with access pattern of Random are shown in Fig. 5 (b). The best job scheduling algorithm also is Queue Access Cost. Not only shortest job queue but also access cost is considered in Queue Access Cost.



(a) Sequential.

(b) Random.

Fig. 5. Job scheduling algorithms with access pattern.

We discuss the disadvantage of the other job scheduling algorithms, respectively. In Random Scheduling randomly submits jobs to CEs to lead to one of CEs that is possible to receive huge amount of jobs. The total job execution time obviously increased because this CE must execute a large number of jobs and it is possible to be idle in other CEs.

In Shortest Job Queue Scheduling, jobs are assigned to specific CE according to shortest job queue. Though each CE receive approximately jobs, it is unfair to lower network bandwidth of specific CEs. These CEs with lower network bandwidth must spend a lot of transmission time to download specific files and then overall job execution time is raised.

In Access Cost Scheduling, jobs are submitted to particular CEs according to access cost. The advantage is to take into account access cost because it can avoid the lower network bandwidth CEs to execute a large number of jobs. The queue size is not considered, and therefore these CEs with lower access cost probably receive huge amount of jobs to execute. However, overall performance and total job execution time is decreased because these CEs receive huge amount of jobs to exceed the workload limit.

Finally, the Queue Access Cost consists of the advantage of Shortest Job Queue and Access Cost. Therefore, the Queue Access Cost is the best job scheduling algorithms to upgrade overall performance and decrease total job execution time.

#### 4.2.2 Min-Max balancing workload method

We adopt job scheduling of Queue Access Cost and access pattern of Random to compare with replica replacement strategies. These replica replacement strategies consist of “NO Replication”, LRU and LFU. The network topology and simulation parameters are shown in Fig. 6 and Table 1, respectively. There are eight types of job size that consists of 2GB, 6GB, 12GB, 14GB, 33GB, 42GB, 51GB and 58GB.

The total job execution time for 100 jobs and comparison of Traditional method and Min-Max method with various replica replacement strategies are shown in Fig. 6 (a). Total job execution time is about 8.9%, 13.5% and 4.2% faster using Min-Max than Traditional method with “NO Replication”, LRU and LFU replica replacement strategy, respectively.

The total job execution time for 1,000 jobs and comparison of Traditional method and Min-Max method with various replica replacement strategies are shown in Fig. 6 (b). Total job execution time is about 19.3%, 45.7% and 41% faster using Min-Max than

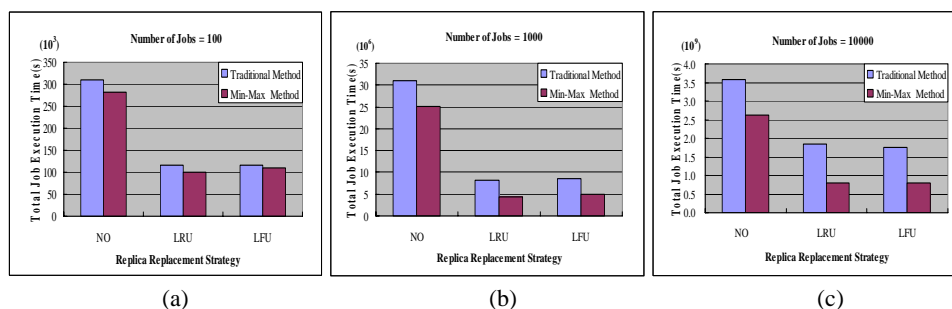


Fig. 6. Min-Max method and traditional method with (a) 100 jobs; (b) 1,000 jobs; (c) 10,000 jobs.

Traditional method with “NO Replication”, LRU and LFU replica replacement strategy, respectively.

The total job execution time for 10,000 jobs and comparison of Traditional method and Min-Max method with various replica replacement strategies are shown in Fig. 6 (c). Total job execution time is about 26.3%, 57% and 54.6% faster using Min-Max than Traditional method with “NO Replication”, LRU and LFU replica replacement strategy, respectively.

Finally, we increase the number of jobs from 100 to 10,000 and then the total job execution time extremely reduces. Min-Max Balancing Workload can avoid the Traditional method that submits a large job to the lower network bandwidth node. Therefore, our Min-Max method is not only balance the workload but also reduce total job execution time and upgrade the overall performance.

## 5. CONCLUSIONS AND FUTURE WORK

Data grids are becoming increasingly important and playing important role in scientific communities for sharing large data collections and for archiving and disseminating them in a digital library framework. A transparent middleware for sharing data across distributed, heterogeneous data resources separated by different administrative and security domains is needed to archive data sharing and data management.

In this paper, we present the design of PU-DG Optibox, which not only provides users for selecting their personal priority choices, but also provide manual and automatic download modes for users, independently if they are expert or not in computing. We expect that such proposal may reduce the burden most users may have, of operating and managing files in a data grid environment.

In order to improve the performance of PU-DG Optibox, mathematical formulations for download time are transformed into dynamic programming problem, reducing the final time complexity to  $O(n)$ , where  $n$  is the number of candidate replica sites. Our proposed PU-DG Optibox can easily help users to find out, analyze, and choose best set of replica sites to download their files. According to PU-DG Optibox, users with or without knowledge about data grid or SRB can also download files from data grid environment simply and quickly, in the fastest way.

From simulation results, the usage of CEs and total job execution time indicate the

Min-Max is better than Traditional method, since each node can run the job with suitable data size and avoid particular nodes have huge amount of jobs to execute.

There are still many issues remained and to be considered, as we will add more factors into the proposed PU-DG Optibox to improve performance in terms of location selection and reliability of file downloading process.

## REFERENCES

1. SDSC SRB, <http://www.sdsc.edu/srb>.
2. The Globus Project, <http://www.globus.org>.
3. J. H. Abawajy, "Placement of file replicas in data grid environments," in *Proceedings of International Conference on Computational Science*, LNCS 3038, 2004, pp. 66-73.
4. W. Allcock, "GridFTP: Protocol extensions to FTP for the grid," in *Global Grid Forum GFD-R-P.020*, 2003.
5. B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke, "Data management and transfer in high performance computational grid environments," *Parallel Computing*, Vol. 28 2002, pp. 749-771.
6. W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini, "OptorSim – A grid simulator for studying dynamic data replication strategies," *International Journal of High Performance Computing Applications*, Vol. 17, 2003, pp. 403-416.
7. D. G. Cameron, R. C. Schiaffino, P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "Evaluating scheduling and replica optimization strategies in OptorSim," in *Proceedings of the 4th International Workshop on Grid Computing*, 2002, pp. 52-59.
8. D. G. Cameron, R. C. Schiaffino, J. Ferguson, P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "OptorSim v2.0 installation and user guide," 2004, [http://edg-wp2.web.cern.ch/edg-wp2/optimization/downloads/v2\\_0/edg-optorsim/doc/userguide-optorsim.ps](http://edg-wp2.web.cern.ch/edg-wp2/optimization/downloads/v2_0/edg-optorsim/doc/userguide-optorsim.ps).
9. M. Ciglan and L. Hluchy, "Towards scalable grid replica optimization framework," in *Proceedings of the 4th International Symposium on Parallel and Distributed Computing*, 2005, pp. 43-50.
10. I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, Vol. 15, 2001, pp. 200-222.
11. H. Lamahmedi, B. Szymanski, Z. Shentu, and E. Deelman, "Data replication strategies in grid environments," in *Proceedings of the 5th International Conference on Algorithms and Architecture for Parallel Processing*, 2002, pp. 378-383.
12. P. Liu and J. J. Wu, "Optimal replica placement strategy for hierarchical data grid systems," in *Proceedings of the 6th ACM/IEEE International Symposium on Cluster Computing and the Grid*, 2006, pp. 417-420.
13. S. Takizawa, Y. Takamiya, H. Nakada, and S. Matsuoka, "A scalable multi-replication framework for data grid," in *Proceedings of the Symposium on Applications and the Internet Workshops*, 2005, pp. 310-315.

14. S. Vazhkudai, S. Tuecke, and I. Foster, "Replica selection in the globus data grid," in *Proceedings of the 1st IEEE/ACM International Conference on Cluster Computing and the Grid*, 2001, pp. 106-113.
15. T. Y. Wu, K. C. Li, H. H. Wang, C. H. Hsu, K. Y. Cheng, W. S. Su, and M. H. Lee, "Towards design of a file location selection system in grid environments," in *Proceedings of International Conference on Hybrid Information Technology*, Vol. 2, 2006, pp. 250-256.



**Kuan-Ching Li (李冠憬)** received the Ph.D. and M.S. in Electrical Engineering and Licenciatura in Mathematics from University of Sao Paulo, Brazil in 2001, 1996 and 1994, respectively. He is currently Associate Professor of the Department of Computer Science and Information Engineering at the Providence University (PU), Taiwan. Prior to join PU in 2003, he had been post-doc scholar at the University of California - Irvine (UCI) and University of Southern California (USC), U.S.A. He has served as steering and program committee member, program chair, or general chair of more than 100 conferences and workshops.

His research interests include parallel and distributed computing, parallel software design, software optimization techniques for multicore processors, and life science applications. He is a senior member of the IEEE.



**Hsiao-Hsi Wang (王孝熙)** received the B.S. degree in Information Science from National Chiao Tung University, Taiwan in 1984, and the Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan in 1993. Dr. Wang is currently a Professor of the Department of Computer Science and Information Management, Providence University, Taiwan. His research interests include cluster computing, distributed processing, distributed shared memory, and wireless networks.



**Kuo-Yang Cheng (鄭國陽)** received a B.S. degree in Computer Science and Information Engineering from Chung Hua University, Taiwan in 2004, and the M.S. degree in Computer Science and Information Management from Providence University, Taiwan in 2007. In that same year, he joined National Center for High-performance Computing, Taiwan, where he is currently an Assistant Researcher. His research interests include data grid and distributed shared memory.



**Tsung-Ying Wu (吳宗穎)** received Master degrees in Computer Science, Finance and Business Administration from Southern New Hampshire University, USA, between 1995 and 1999. In 2003, he joined National Center for High-performance Computing, Taiwan, where he is currently an Associate Researcher. His research interests include grid computing, data grid, and grid related applications.