AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 50(1)(2018)39-50
DOI: 10.22060/miscj.2017.12236.5020

An Efficient Data Replication Strategy in Large-Scale Data Grid Environments Based on Availability and Popularity

N. Mansouri*, M. M. Javidi

Computer Science Department, Shahid Bahonar University of Kerman, Kerman, Iran

ABSTRACT: The data grid technology, which uses the scale of the Internet to solve storage limitation for the huge amount of data, has become one of the hot research topics. Recently, data replication strategies have been widely employed in a distributed environment to copy frequently accessed data in suitable sites. The primary purposes are shortening distances of the file transmission and achieving files from nearby locations to requested sites so as to minimize retrieval time and bandwidth usage. In this paper, we propose a new replica selection strategy which is based on response time and security. However, replication should be used wisely because the storage size of each Data Grid site is limited. In addition, we propose a new replica replacement strategy that considers file availability, time of access, access frequency and size of file. The simulation results report that the proposed strategy can effectively improve mean job time, bandwidth consumption for data delivery, and data availability compared with those of the tested algorithms.

Review History:
Received: 8 December 2016
Revised: 31 July 2017
Accepted: 24 October 2017
Available Online: 11 March 2018

Keywords:
Data Grid
Dynamic replication
File access pattern
Job Scheduling

1- Introduction

Chervenak et al. [1] introduced data grid technology as a new architecture that is a specialization and extension of the classic grid. Data grid satisfies the geographically distant distribution of customers and resources and computationally intensive study. Therefore, a layer structure of middleware (Figure 1) similar to the one explained in [1] is designed. A lower layer of the core services and higher layer services are two important layers. The lower level can perform operations such as reading, removing, generating, and modifying file instances. One of the key components of the high level is information and replica management. The main difference between a data grid and a classical grid was the proposal of the replica system and its subsystems: replica selection and replica management components. These subsystems are available on top of the previously explained resource management system.

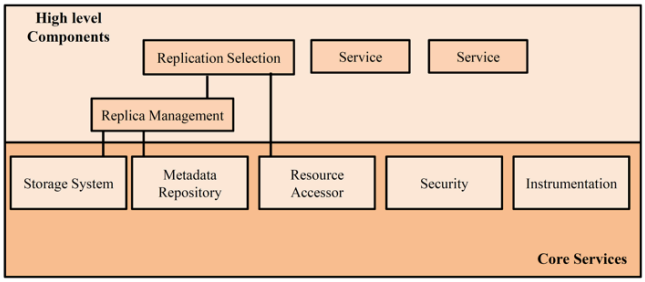

Fig. 1. Structure of the Data Grid architecture.

1- 1- ISSUES RELATED TO DATA GRID

Data Grids have unique issues such as the following:
Massive Datasets: Data-intensive applications work with huge datasets, with the size of Gigabytes (GB) and beyond. For instance, the CMS experiment at the LHC generates 1 PB (1015 bytes) of RAW data and 2 PB of Event Summary Data (ESD) annually [2-3]. Therefore, resource management replicates different data to decrease latencies of bulk data transfers based on replication strategy.
Shared Data Collections: Resource sharing of Data Grids also comprises, among others, sharing distributed data sets. For instance, developers of a specific application need to apply the same repositories and data sets as the sources for necessary operations and for storing the outputs of their investigations.
Unified Namespace: Data file of Data Grid shares the same logical namespace where every data set has a specific logical filename. The logical file name is mapped to one or more physical filenames across different Data Grid sites.
Access Restrictions: Customers want to ensure confidentiality of their critical files or restrict distribution to close colleagues.

1- 2- Data Replication

Service quality is a key requirement in Grid applications. Generally, Data Grid improved the performance and quality of service (QoS) by creating multiple replicas in the geographically distributed locations. Data replication strategy is a powerful strategy to enhance the performance of data sharing through Data Grid environments. The outstanding characteristics of data replication provide fault tolerance, load balancing and improve resource availability and reliability. Usually, the storage restriction limits the amount of data file that can be copied by replication strategy. Thus, replication strategies are needed to decide when replication must take place, what data files must replicate and where the replicas must be stored. Figure 2 shows schematically the replica management system, including storage elements which are connected to each other via high-performance data transport protocols.

Corresponding author; Email: najme.mansouri@gmail.com

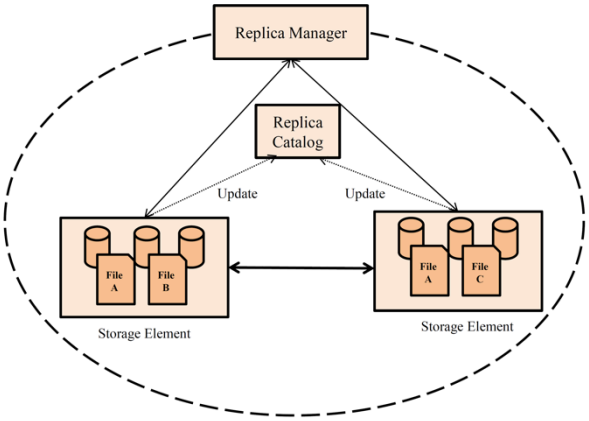

Fig. 2. A Replica Management Architecture.

Generally, replication algorithms are either static or dynamic. In static approaches, the created replica will exist in the same place till user deletes it manually or when its duration is expired. On the other hand, dynamic strategies create and delete replicas according to the changes in grid environments, i.e. users' file access the pattern. However, frequent transfers of huge data files that occur due to such strategies can lead to straining on the network resources. There may be little profit from using dynamic strategies if the resource states are relatively stable in a Data Grid over a long time. Therefore, in such cases, static strategies are used for replication. Several studies have been proposed in the literature to resolve the problem of replication in the Grid environment [4-7].

There are different sites with various capacities and abilities in the grid; therefore, choosing an appropriate site that has most required data is a major decision in the replication process. This key decision called the replica selection. According to the previous works and to the best of our knowledge, most previous investigations that considered the replica selection process have only focused on retrieval time. Munir et al. [8] summarized the QoS factors in Table 1, where the only concern was time.

However, there are criteria other than time and in some cases, they are more significant.

Table 1. QoS parameters in grid distributed between the network and computational aspects.

Network	Computational
Bandwidth	CPU Speed
Latency	System's Memory
Distance between Nodes	Data Access

Sometimes user prefers to choose the required data from secure sites rather than sites with short response times. There is a high risk of hacking in wide area networks by many unauthorized users, particularly when the required files are serious. Also, some sites in the grid environment are active for a limited number of hours, selecting the most available site can enhance performance.

In this work, the replica selection process is investigated to study the effectiveness of replication strategy and to ensure the contentment of different users, providing them with their needed replicas in a secure and quick way. Due to the restriction of storage capacity issues, using efficient replica replacement is more serious for the improvement of performance. A new data replica replacement algorithm has been proposed on the base of file availability, the last time the replica was requested, number of access, and size of the replica as criteria. The proposed strategy evaluates using simulation platform called OptorSim [9], written in Java. OptorSim, i.e., a Data Grid simulator, enables us to evaluate different replication optimization strategies in a Grid environment, effectively. The simulation results show that EDRS successfully enhances the effective network usage. It means that the EDRS replication algorithm can select a popular file and replicate it to a suitable site without increasing the network burden too much.

The remainder of the paper is organized as follows. Section 2 describes the related work in data replication. Section 3 presents the proposed algorithm. In section 4, the simulation results based on the OptorSim, a simulator designed by the European Data Grid Project is provided. The final section contains conclusion and future work.

2- Related work

There are various strategies for handling data replication in distributed environments. In [10-11], the authors provided a survey on the replication algorithms for different distributed storage and content management systems, including Grid architectures. Foster and Ranganathan [12-14], proposed six distinct replica strategies: No Replica, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread) for multi-tier Data Grid. They also introduced three types of localities, namely Temporal locality (The files accessed recently are much possible to be requested again shortly), Geographical locality (The files accessed recently by a client are probably to be requested by adjacent clients, as well) and Spatial locality (The related files to recently accessed file are likely to be requested in the near future). These strategies are evaluated with different data patterns: first, access pattern with no locality. Second, data access with a small degree of temporal locality and finally data access with a small degree of temporal and geographical locality. The results of simulations indicate that different access patterns need different replica strategies. Cascading and Fast Spread performed best in the simulations. Also, the authors combined different scheduling and replication strategies.

Fast Spread is one of the best replication strategies especially for random request patterns [14]. In this strategy which is the main concern, a replica of the requested file is stored at each node along its path to the requester. If the storage of one of these nodes is full, a group of existing replicas (that contains one or more replicas) needs to be replaced with the new replica. Fast Spread can use one of the many replacement strategies to determine the replicas that need to be replaced. Two of the well-known replacement strategies are LRU and LFU. Fast Spread with LRU discards the least recently used replicas first, while Fast Spread with LFU discards the least frequently used replicas first. The problem arises when this group of existing replicas is more important than the new replica.

Bsoul et al. [15] proposed a new strategy named Enhanced Fast Spread (EFS) to solve this problem. The new proposed strategy is based on Fast Spread but superior to it. The EFS takes important factors into consideration when replacing the candidate group with the new replica: the size of the replica and the last time at which the replica was requested. The simulation results show that the new proposed strategy achieved a better performance than Fast Spread with Least

Recently Used (LRU) and Fast Spread with Least Frequently Used (LFU) in terms of total response time and total bandwidth consumption.

Sashi and Thanamani [16] have extended Latest Access Largest Weight (LALW) strategy [17] where the replicas are created based on their weights. LALW algorithm gives higher weighs to recently requested files and replica placement done only in cluster levels and not in the site levels. But Sashi et al.'s algorithm minimizes mean job execution time by placing the replicas in the best site within the cluster by considering the number of file requests and response time.

Park et al. [18] presented a Bandwidth Hierarchy based Replication (BHR) which decreases the data access time by maximizing network-level locality and avoiding network congestions. They divided the sites into several regions, where network bandwidth between the regions is lower than the bandwidth within the regions. Hence, if the required file is placed in the same region, its fetching time will be less. BHR strategy has two deficiencies, first it terminates if replica exists within the region and second it stores the replicas in all the requested sites instead of storing them in the appropriate sites. BHR strategy has a good performance only when the capacity of the storage element is small. Modified BHR [19] is an extension of BHR [18] strategy which replicates a file that has been accessed most and it may also be used in near future.

Horri et al. [20] presented a 3-Level Hierarchical Algorithm (3LHA). They considered a hierarchical network structure that has three levels. In their replica selection method among the candidate replicas, they selected the one that has the highest bandwidth to the requested file. Similarly, it uses the same technique for the file deletion. This leads to a better performance compared with LRU (Least Recently Used) method. For efficient scheduling, their algorithm selects the best region, LAN, and site, respectively. Best region (LAN, site) is a region (LAN, site) with most of the requested files.

Mansouri and Dastghaibifard [21] proposed a dynamic data replication strategy called Dynamic Hierarchical Replication Algorithm (DHRA) that enhances the 3-Level Hierarchical Algorithm (3LHA) in [20]. The DHRA stores each replica in an appropriate site, i.e., appropriate site in the requested region that has the highest number of access for that particular replica.

In [22], the authors developed the Hierarchical Cluster Scheduling strategy (HCS) and the Hierarchical Replication Strategy (HRS) to enhance the data access efficiencies in the Grid. HCS considers hierarchical scheduling and uses cluster information to decrease search time for a proper computing node. HRS replication algorithm uses the concept of "network locality" as Bandwidth Hierarchy based Replication (BHR) strategy. HRS compared with BHR has two advanatges: First, BHR checks all sites to locate the best replica, while HRS locates a replica within the local cluster. Second, HRS uses the popularity of replicas at site level but BHR uses the popularity of replicas at the cluster level. HCS scheduling along with HRS replica strategy improves data access time and the amount of inter-cluster communications in comparison to others scheduling algorithms and replication strategies. However, HRS just considers the bandwidth to delete and select replicas.

In [23], we presented a Dynamic Hierarchical Replication (DHR) algorithm that stores replica in suitable sites where the particular file has been most accessed, instead of storing the file in many sites. It also decreases access latency by selecting the best replica when different sites hold replicas. The proposed replica selection strategy chooses the best replica location for the users' running jobs by considering the replica requests waiting in the storage and data transfer time. The simulation results show that it has less job execution time in comparison with other strategies especially when the Grid sites have comparatively small storage size. According to the previous studies, although DHR represents some improvements in some metrics of performance like mean job time, it shows some deficiencies. Replica selection and replica replacement strategies in DHR strategy are not very efficient. In [24], we proposed a Modified Dynamic Hierarchical Replication Algorithm (MDHRA) that improves DHR strategy. MDHRA considers the last time at which the replica was requested and frequency of access for replica replacement step. It also improves access latency by selecting the best replica when various sites hold replicas. The proposed replica selection selects the best replica location among the many replicas based on response time that can be determined by considering the data transfer time, the storage access latency, the replica requests waiting in the storage queue and the distance between nodes. Also, a job scheduling algorithm called Combined Scheduling Strategy (CSS) is proposed in [24] that use hierarchical scheduling to reduce the search time for an appropriate computing node. It considers the number of jobs waiting in queue, the location of required data for the jobs, and the computation capacity of sites.

Mansouri et al. [25] proposed a dynamic data replication strategy, called Enhanced Dynamic Hierarchical Replication (EDHR) that improves DHR strategy. It employs an economic model for the file deletion when there is not enough space for the replica. The economic model is based on the future value of data file. EDHR also considers the frequency of requests of the replica and the last time at which the replica was requested for replica placement. They concluded that EDHR can be effectively utilized when there is a hierarchy of bandwidth.

Wang et al. [26] proposed a dynamic data replication strategy using historical access record and proactive deletion. The algorithm is named Closest Access Greatest Weight with Proactive Deletion (CAGW_PD) dynamic replication strategy. If the popularity of a file exceeds a threshold, i.e. the average popularity of all the files, the CAGW_PD decides to do a replication. By associating a different weight to each historical data access file, importance of each file is differentiated. A more recent data access file has a larger weight. CAGW_PD applies a proactive deletion technique, which is applied to control the replica number, to reach an optimal balance between the read access time and the written update overhead. A cost model is employed as a means to evaluate and compare the performance of CAGW_PD algorithm and other existing algorithms. The results showed that CAGW_PD algorithm outperforms those algorithms. However, from the results, we can observe that the performance of CAGW_PD algorithm is not as good as we think. The main reason is that their replica placement strategy has some deficiencies. Where to store new replicas requires one to solve a constrained optimization problem which is NP-complete in general.

Yang et al. [27] presented dynamic services for replicating and maintaining data in Data Grid, and directing replicas

to suitable locations for use. To address a problem with Bandwidth Hierarchy-based Replication (BHR) strategy, an algorithm for dynamically maintaining replicas, they proposed Dynamic Maintenance Service (DMS). They also presented a One-way Replica Consistency Service (ORCS) for Data Grid environments, that is a positive method to solve consistency maintenance problems. Using DMS adjusts data to locations suitable to the sites that need the data more often, thus decreasing the amount of time needed by those sites to get the required data and enhance the performance. Using ORCS enhances the performance by replica consistency method and improving Data Grid storage device usage ratios. DMS and ORCS strategies consider storage element capacity in replica placement step and temporary data produced in computing step. This reduces the probability of applications crashing or having to resubmit jobs to other computing resources. Their experimental results represented that DMS and ORCS are more efficient than other algorithms, enhance computation performance, and reduce storage usage.

Rahman et al. [28] proposed an algorithm for replica selection by using a simple technique called the K-Nearest Neighbor (KNN). The KNN rule chooses the best replica for a file by using previous file transfer logs. They also suggested a predictive way to estimate the transfer time between sites. Accordingly, one site can request the replica from a site which has the minimum transfer time. They showed that a neural network-based prediction technique outperforms the multi-regression model.

Each grid site has its own abilities and properties; consequently, choosing one specific site with the best Quality of Service (QoS) among numerous sites that have the required replica is a key, complicated and challenging decision. Replica selection strategies, such as greedy [29-30], random [31-32], partitioned [33-35], and weighted algorithms [36-37], were mainly used in the replica selection function and have exclusively focused on response time as a criterion for the selection process. The proposed solution considers the security factor which is vital for the security of the data and the whole process.

3- Proposed data replication strategy

In this section, we introduce the network structure and then explain our novel dynamic data replication strategy.

3- 1- Network Structure

Fig. 3 shows the Grid structure of simulated architecture. It includes three levels. At the top level are the Regions that are linked with a low bandwidth. The second level comprises local area networks that are placed in regions and have moderately higher bandwidth compared to the top level. Finally, the third level contains the data nodes that are placed in local area networks and linked to each other with a high bandwidth. In our simulation, we consider 4 regions each of which has 12 sites on average. Bandwidth between sites of regions is 100 Mbps whereas bandwidth between sites across each region is 10 Mbps. This is because most sites transfer large files from other sites of the region by the inter-region link. Therefore, this link has a heavy traffic and causes topology with hierarchy bandwidth.

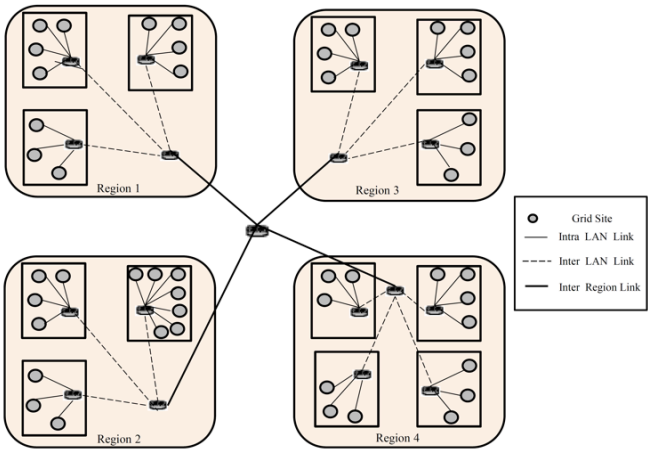

Fig. 3. Grid topology in simulation.

3- 2- Efficient Data Replication Strategy (Edrs)

When a job is going to be executed, the replica manager should provide all the required files that are unavailable. Thus, an efficient data replication algorithm can enhance job scheduling performance by minimizing job turnaround time. EDRS algorithm has three important sections:

Replica selection: Commonly, Data Grid sites have different capabilities in providing various levels of QoS. Four important factors are considered to choose an appropriate replica:

- ***Elapsed Time (ET)***

Kusy et al. [38] defined the elapsed time as the time the replica takes to be transmitted from the sending node to the receiving one. The low value of elapsed time indicates that the nodes will achieve the replicas in less time. When the node has a requested replica, the elapsed time is equal to zero. The *ET* is obtained from:

$$ET = TT + PT, \tag{1}$$

where *TT* indicates the transmission time, and *PT* shows the propagation time.

- ***Bandwidth Consumption (BC)***

The bandwidth usage in replica transmission is defined as bandwidth consumption. The *BC* is determined based on:

$$BC = NN \times R, \tag{2}$$

where *NN* shows the number of intermediate nodes between the sending and receiving nodes, and *R* indicates the replica size.

- ***Storage access latency***

The storage media speed is the main parameter in the average response time determination [39]. *T* can be calculated by the following equation:

$$T = \frac{R}{SS} \tag{3}$$

where *SS* represents the storage speed, and *R* shows the replica size.

- ***Self-Protection Capability***

We determine the self-protection capability of a site based on the following security parameters. Their values are in the interval (0, 1).

IDS Capabilities: It shows the capability of a site in the supporting system from host and network-based intrusions.
Anti-virus Capabilities: It determines the capability of a site to prevent viruses and malicious codes.
Firewall Capabilities: It indicates the capability of a site to defend against other network accesses.
Authentication Mechanism: It represents the capability of a site in identification of a system user.
Secured File Storage Capabilities: It indicates the capability of a site in keeping files needed for a job in a safe manner.
Interoperability: It shows the ability of a site to restrict interfacing between simultaneous tasks.
Therefore, the self-protection capability value is computed based on the following formula:

$$SC = \sum_{i=1}^{n} W_i \times V_i , \tag{4}$$

where n indicates a number of security factors, W_i shows the weights and V_i represents security factors value.

$$F(x,y) = w_1 \times ET + w_2 \times BC + w_3 \times T + w_4 \times \frac{1}{SC}, \tag{5}$$

We tailor the above function since it is a combination of different metrics. The primary goal of replica manager is reducing time transmission for each job. Replica manager checks the existence of the file in the local region for finding unavailable requested files in job execution. When needed, files are duplicated in the same LAN, then our strategy lists them and finds a replica that has the lowest F value. When the needed file is unavailable in the same LAN, then our strategy seeks the local region. When the needed files are duplicated in the same region, then our strategy lists them and finds a replica that has the lowest F value. Otherwise, our strategy lists replicas in other regions and finds the replica with the lowest F value.

Replica placement: If the local site does not have a needed file, then replication should take place. Our strategy finds the best site (BSE) based on the temporal and geographical locality concept. Hence, it finds the site that has the lowest Value (V). We use request frequency of replica and the last time the replica was requested for computing V. With these factors, we can achieve the useful information such as the probability of requesting the replica again.

$$V_i = (CT - LT_i) + \frac{1}{FR_i} + \frac{1}{SP}, \tag{6}$$

where CT shows the current time, LT_i indicates the last request time of replica i, and FR_i shows the request frequency for replica i. SP indicates the storage performance of site x that is found based on the node computing power (in GFLOPS), storage bandwidth (in MBytes), and a portion of storage devoted to the grid (non-null), as presented below.

$$SP(x) = Copmputing\_power(x)\frac{Disk\_bandwidth(x)}{Available\_grid\_storage(x)} \tag{7}$$

Replica Replacement: If enough storage is available in the best site, then we can store the replicas in it. And, if the local LAN contains the particular file, then our strategy accesses the file remotely.
Now, if the best site does not have enough storage for storing replica and requested file does not exist in the local LAN, our strategy removes one or more files based on the following instructions:
It lists replicas that exist in the site and the local LAN based on the LRU strategy. Then, it removes the files from the above list until enough storage is provided.
It lists replicas that exist in the site and the local region based on the LRU strategy. Then, it removes files from the above list until enough storage is provided.
If storage is still insufficient, one or more replicas must be removed. In this situation, LRU strategy may select some important files for deletion that may not exist in the local region. Nearly, they may be requested again. Therefore, such decisions leads to an increase in data transfer cost. Instead of LRU, we use important parameters such as the file availability, the last time the replica was requested, the number of access, and the file size. The number of access and the last time the replica is requested to show the probability of requesting the replica again. It is necessary to replace files with a large size that can decrease the number of replica replacement. The replica cost (C) is determined by

$$C = \frac{w_1 \times NA + w_2 \times P}{w_3 \times (CT - LA) + w_4 \times S}, \tag{8}$$

where S indicates the file size, NA is the number of access to the replica, CT shows the current time, LA represents the last time the replica is requested and P shows the availability of the file. Our strategy deletes replica that has the lowest value of C. Also, weights are determined by user preferences. Increasing the replicas in one site does not help the availability factor since if the site is crashed, all the files on the site are not accessible. Also, such unnecessary replication processes only waste resources such as storage and bandwidth.
In the proposed strategy, a file availability index is associated with each site, which indicates the probability that a file on that site will be available. There are two assumptions. First, the maximum of one replica for a file in each site exists. Secondly, availability indices of all files on the same site are identical. File availability in ith site is denoted by $PASE_i$. Now, the probability for availability of each physical file (P), can be determined. Since a file may exist on more than one site, the probability of the availability of a file f_j is computed by the file availability in each site that has file f_j. Then the availability of file f_j is obtained by

$$P = 1 - \prod_{i=1}^{k} (1 - PASE_i), \tag{9}$$

where the number of copies of the file fj is k. For each file access, the potential unavailability is defined as $1 - P$. The proposed process supposes that a particular file access operation accesses only one file and any two-file access operations are independent. Fig. 4 explains the procedure of the replica replacement strategy. Also, the flowchart of the proposed replication strategy is illustrated in Fig 5.

4- Experiments

This section explains simulation toolkit, the configuration of the network, and simulation results.

4- 1- Optorsim System Architecture

The structure of OptorSim platform is shown in Fig. 6. OptorSim defines a grid as different sites, each of which includes zero or more computing elements (CEs) and zero or

more storage elements (SEs) [40].

```
If (R is available in the local LAN)    continue;

// Now delete those files from BSE that are also available
// in the local LAN
      Create list L of fj's that are both available in the BSE as
      well as in the local LAN.
      Sort list L using LRU.
      While (L is not empty && not enough space for R)
            delete first file from list L in BSE;
      If (enough space exist for R in BSE) {Store R;     continue;}

// Now delete those files from BSE that are also available
// in the local region
      Create list L of fj's that are both available in the BSE as
      well as in the local region.
      Sort list L using LRU.
      While (L is not empty && not enough space for R)
            delete first file from list L in BSE;
      If (enough space exist for R in BSE) {Store R;     continue;}

// Now delete from remaining files in BSE
      Create list L from remaining files in BSE.
      Sort list L using C in ascending order (Eq. 8)
      While (L is not empty && not enough space for R)
      Delete first file from list L in BSE;
      Store R;
```

Fig. 4. Pseudo code for replica replacement strategy.

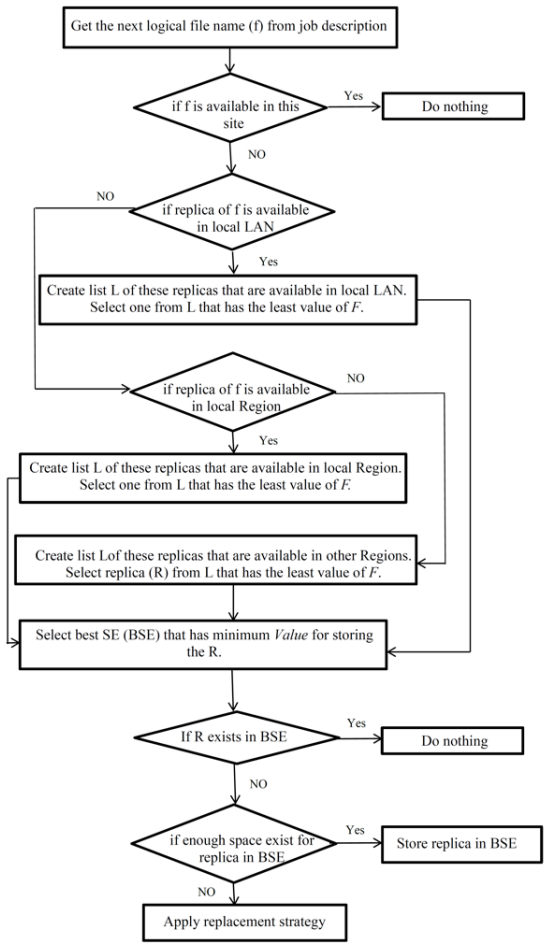

Fig. 5. Flowchart of EDRS.

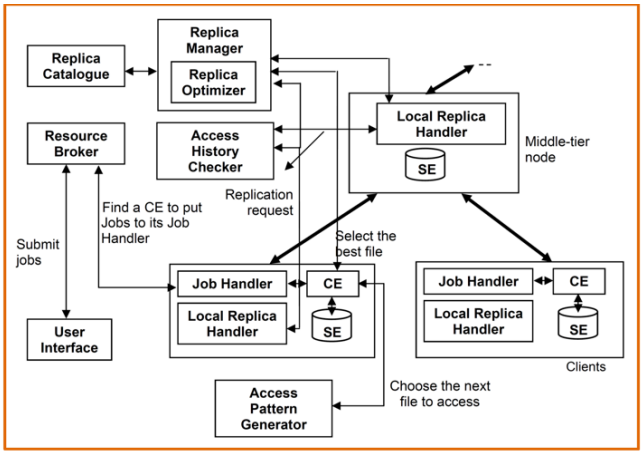

Fig. 6. Architecture of OptorSim toolkit [40].

A resource broker manages job assignment on different sites like meta-scheduler. Each job needs different files that may be stored in other sites. An abstract reference to a file regardless of its location in the sites is known as the logical filename. A physical file name is defined as a specific replica of an LFN placed in a particular site. It is possible that a logical file has many physical locations. Each site needs to communicate with a replica manager to find the physical locations of a file. After achieving necessary information from the replica catalog, the replica manager determines the locations of the physical files for the needed file.

4- 2- Configuration

The network topology that is used in our simulation is shown in Fig. 3. We define 50 job types, and each job type needs 15 files with 2 GB for completion. 1500 jobs are randomly submitted to the Resource Broker at regular intervals. Since some job types may be selected frequently, some files are required repeatedly. Some important simulation parameters are reported in Table 2.

Table 2. EDRS configurations.

Topology parameter	Value
Number of regions	4
Storage space at each site (GB)	50
Inter LAN bandwidth (Mpbs)	1000
Intra LAN bandwidth (Mpbs)	100
Intra Region bandwidth (Mpbs)	10
Grid job parameter	Value
Number of jobs	1500
Number of jobs types	50
Number of file access per jobs	16
Size of single file (GB)	2
Job delay (ms)	10000

Queue Access Cost scheduler is set as a default scheduler. This scheduler sends the job to the site that has the lowest sum of the access cost for all the jobs in the queue. Request files order in a job is specified based on the particular access pattern. There are five important access patterns:

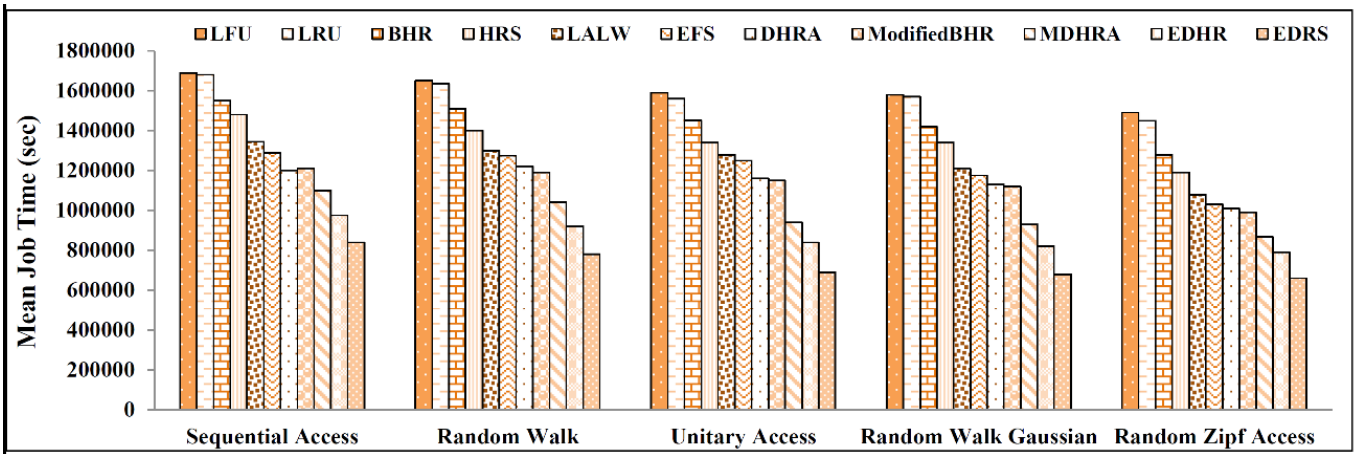

Fig. 7. Mean job time with different access pattern generators.

sequential (file access order is based on the file order in the job configuration file), random (file access order is based on a flat random distribution), unitary random (one file is requested away from previous file request and the direction is random), Gaussian random walk (file access order is based on a Gaussian distribution), and Random Zipf access (file access order is based on the $P_i = K/i^s$, where P_i shows the frequency of the ith ranked element, K indicates the popularity for the most frequently requested element and distribution shape is set by s variable.*)* Most replication strategies usually assume that the data is read-only in Data Grid systems. We do not focus on the consistency issue in our work.

4- 3- Simulation Results And Discussion

To compare and evaluate the performance of proposed and most famous replication strategies implemented in OptorSim, the following metric was employed:

Mean Job Execution Time: The mean job execution time is defined as the ratio of the total required time for all the jobs to the number of completed jobs. This metric is probably the most significant indicator that how the algorithms are performed.

Effective Network Usage (ENU): ENU is used to calculate the efficiency of the network resource usage. E_{enu} is usually depicted as [40]:

$$E_{enu} = \frac{N_{rfa} + N_{fa}}{N_{lfa}}, \qquad (10)$$

in which N_{rfa} is the access time number that CE reads a file from a remote site, N_{fa} is the total number of file replication operation, and N_{lfa} is the number of times that *CE* reads a file locally. E_{enu} changes between 0 and 1. The bandwidth utilization is increased by decreasing the E_{enu} to zero. A lower value represents that the network bandwidth is used more efficiently.

SE Usage: It is a criterion for monitoring storage resources in Grid Sites. SE usage enables us to estimate the storage percentages usage during simulation.

Replication frequency: It is estimated by the division of replication frequency into the frequency of data access from clients. The value of replication frequency can be depicted as the number of replicates per access time.

Hit ratio: It is estimated by the division of a total number of Local File Accesses into the summation of three-factor Local File Accesses, total number of Replications, and total number of Remote File Accesses.

The effect of five access pattern generations, i.e., Random, Unitary Random, Random Walk Gaussian, Sequential and Random Zipf distribution, on the mean job time of 11 dynamic replication strategies is shown in Fig. 7. If there is enough space, the Least Frequently Used (LFU) replicates file in the same site where the job is executing, else it removes the least accessed file in the storage element. The same strategy is taken in Least Recently Used (LRU) with this difference that when there is not enough space for the new replica, the oldest file in the storage element is deleted.

The criterion for storing replica in Bandwidth Hierarchy based Replication (BHR) strategy is bandwidth level as well as replicates the most recent requested within the region. Due to the restrictions of the size of SE at each site, storing a large portion of all data is impossible and performance improvement with site-level replacement policy is afftected. Therefore, BHR strategy is more effective due to the employment of network-level locality by storing as many files as possible in a region.

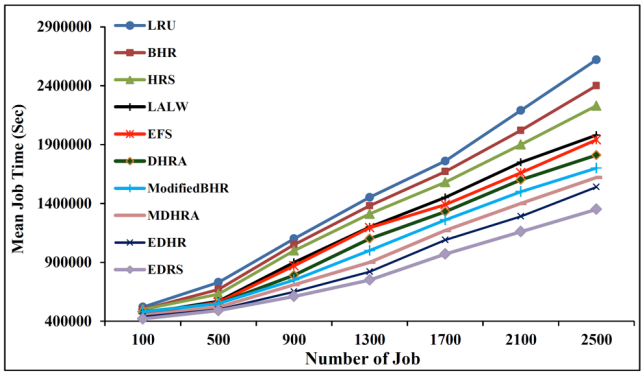

Fig. 8. Mean job time based on varying number of jobs.

As shown in Fig. 7, LFU and LRU strategies have a similar behavior. The mean job time of BHR is higher (to be about 6%) than the one in HRS algorithm with Sequential Access pattern. Since in HRS, instead of searching all site for replica selection, it only investigates the local cluster. Moreover, the mean job time of EFS lowers to 7% compared to LALW algorithm. When there is not enough space, EFS strategy is able to replace a group of replicas with the requested replica only if it is more important than that group. The mean job

time in LALW is about 15% lower than the one in BHR due to the ability of LALW to select a more popular file for replication.

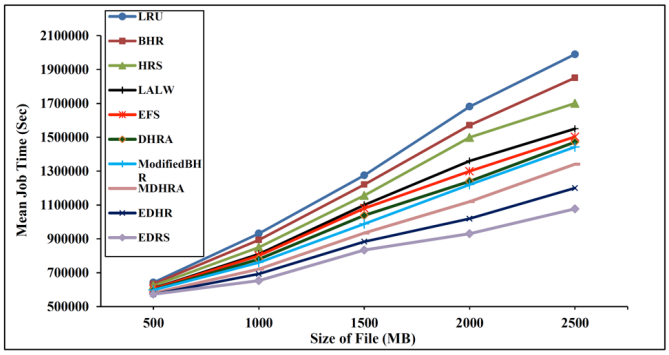

Fig. 9. Mean job time based on the varying size of files.

As shown in Fig. 8, the bandwidth consumption of large scale file even if the presence of replication is so high. Therefore, EDRS tries to select the appropriate replica for high-speed job execution.

By introduing different weights into each historical data access record, the importance of each record changes. The mean job time of EDHR is about 50% lower than LFU algorithm, and 22% lower than that of ModifiedBHR algorithm in Zipf distribution. The minimum mean job execution time belongs to EDRS. As shown in Random access patterns comprising Random, Unitary random, walk Random Zipf and Gaussian random walk, a specific set of files is more likely to be requested by Grid sites, thus, a large number of requested files have been replicated on the previous. Therefore, EDRS strategy and also all the other strategies have considerable improvement for random file access patterns.

The performance evaluation continues by changing the number of jobs and file size. In Fig. 8, by increasing the number of jobs, e.g. a real Grid environment, EDRS outperforms the others. The ability of EDRS strategy to replace a group of replicas with the requested replica based on four criteria, namely, the last time the replica was requested, number of access, size of replica and the availability of the file, minimizes the data access time and avoids unnecessary replication. Consequently, the EDRS successfully decreases the mean job time up to 50% in 2500 jobs.

By increasing the size of the files or the number of jobs, the differences between the algorithms dramatically become distinct . It is necessary to note that for the fixed Grid storage's size, the number of replicas in the Grid must be limited. Consequently, a higher number of replications is necessary and it may consume a considerable amount of network bandwidth. Fig. 9 illustrates the effect of file size on the mean job time for 10 algorithms. The minimum job execution time belongs to EDRS. The performance is significantly affected by the locations of required data in data-intensive applications. Consequently, in a real Data Grid environment with a dataset size equal to several Terabytes the difference will be even more considerable. As depicted in Fig. 9, the performance of EDRS with respect to the other algorithms is better, especially, for heavier loads.

Fig. 10 shows the mean job time of replication algorithms as a function of inter-region bandwidth. Accordingly, the proposed strategy in the narrow bandwidth outperforms significantly on the inter- region link. It is necessary to note that using the frequency of replica requests and the last access time of replica in EDRS as criterion for the placement of replica takes time and consumes network bandwidth. Thus, load balancing is necessary for reducing future network traffic, where any replication is for the sake of reducing future network traffic. ENU of EDRS is effectively lower than the others, indicating the strategy used is better at putting files in the appropriate places.

ENU for the Random Zipf Access Pattern Generator is shown in Fig. 11. The ENU of LRU strategy is higher about 60% compared to that of EDRS strategy. This can be relevant to the ability of Grid sites to access the needed files, instantaneously. Hence, the number of replications will decrease by increasing the number of local accesses. The ENU of EDRS is 16% lower than EDHR and 25% lower than MDHRA, due to its ability to minimize the bandwidth consumption and network traffic. It is necessary to note that due to the absence of replication in NR, remote access will increase the ENU.

Fig. 12 depicts the storage resource usage for various replication strategies. Accordingly, the lowest storage belongs to NR due to the saving of data in only one location, i.e. the file is initially produced, But EDRS creates the replicas in a dynamic way to enhance the performance of system. This strategy stores files after sorting them in a particular site in a way that the storage usage can be reduced. The replicas number in EDRS is lower than LFU and LRU. Consequently, EDRS is able to consider the parameter affecting on performance and effectively decrease the storage usage.

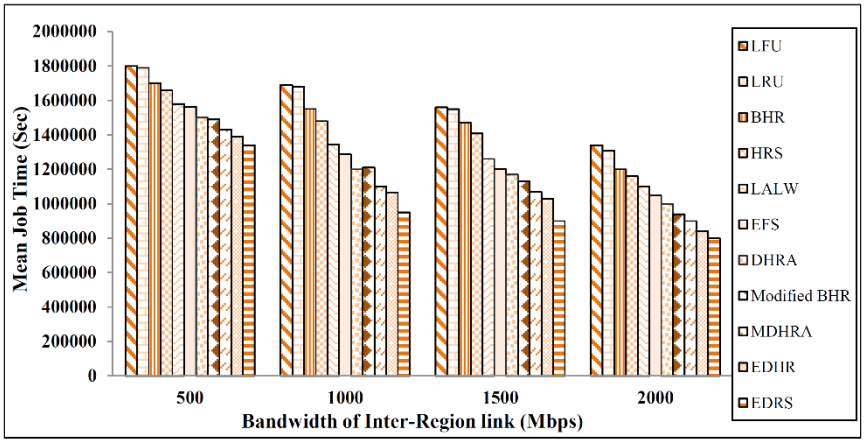

Fig. 10. Mean job time with varying bandwidth.

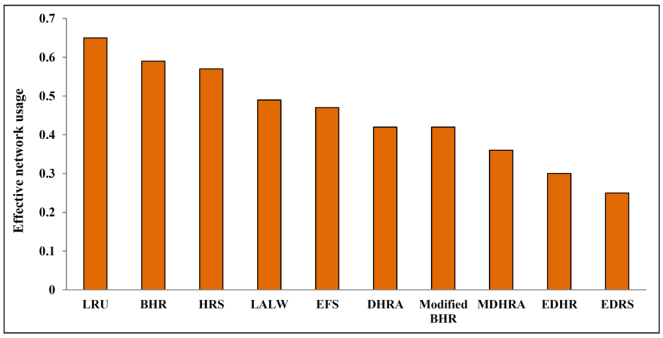

Fig. 11. Effective network usage.

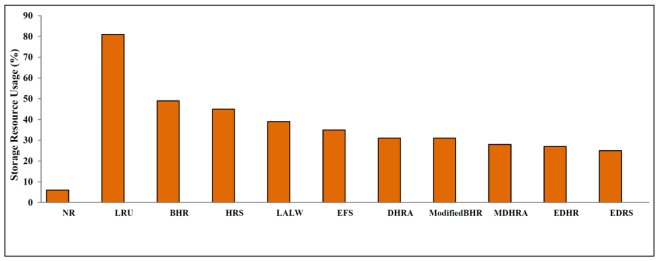

Fig. 12. Storage resources usage.

Moreover, it removes unnecessary communication by selecting suitable replicas using the latency, self-protection capability, bandwidth, and elapsed time as criteria.

Fig. 13 shows the changes of replication frequency for various replication strategies. It is obvious that by increasing the number of replicas, the replication frequency must increase. Replication consumed the network bandwidth resource and consequently increased the replica server load due to usage of disk I/O and CPU. Hence, the frequency of replication must be restricted to prevent from a heavy network traffic and server load. In the case of EDRS, placement of replicas occurs with the average of 14 per 100 data accesses.

Because EDRS places the replicas in the best site by considering the number of requests for the replica and the last time access of the replica, Data replication takes time and consumes network bandwidth. However, performing no replication has been demonstrated to be ineffective compared to even the simplest replication strategy. Therefore, a good balance must be discovered, where any replication is for the sake of reducing future network traffic. ENU is the ratio of files transferred to files requested, thus, a lower ENU indicates that the associated strategy is better at putting files in the right places.

The effective network usage for the Random Zipf Access Pattern Generator is shown in Fig. 11. ENU of EDRS is about 60% lower than that of LRU strategy. The main reason is that required files of Grid sites are present at the time of need, hence, the total number of replications will decrease and the total number of local accesses increases. In effective network usage, EDRS reduces 16% and 25% compared to the EDHR and MDHRA, respectively. The EDRS is optimized to minimize the bandwidth consumption and, thus, decrease the network traffic. No Replication strategy does not make any replication, thus the remote access will increase ENU.

Figure 12 depicts the storage resource usage. The storage resource usage of No Replication strategy is best because in this case the data is stored only in one location where the files are produced initially. But EDRS algorithm creates replicas dynamically in advance. Instead of storing files in many sites, they can be stored in a particular site so that the storage usage can be reduced. The number of replicas in the Data Grid with LFU and LRU is higher than that for EDRS algorithm. EDRS successfully decreases storage usage while considering other performance objectives. Also, it avoids unnecessary communication by selecting appropriate replicas based on the latency, self-protection capability, bandwidth, and elapsed time.

Figure 13 shows the replication frequency for various replication strategies. An increased number of replicas imply a higher replication frequency which is the value of how many replications occur per data access. When replication takes place, not only is the network bandwidth resource consumed but also the replica server load grows because of the disk I/O and CPU utilization. Therefore, the frequency of replication operation must be limited to avoid heavy network and server load. In the case of EDRS, placement of replicas occurs with the average of 14 per 100 data accesses.

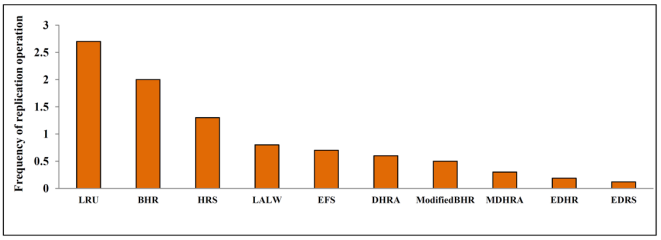

Fig. 13. Replication frequency of the dynamic replication algorithms.

The frequency of LRU strategy is higher than 2.7. It means that at least 2.7 replicas are necessary for a data access. The replication frequency of LRU strategy is too high which

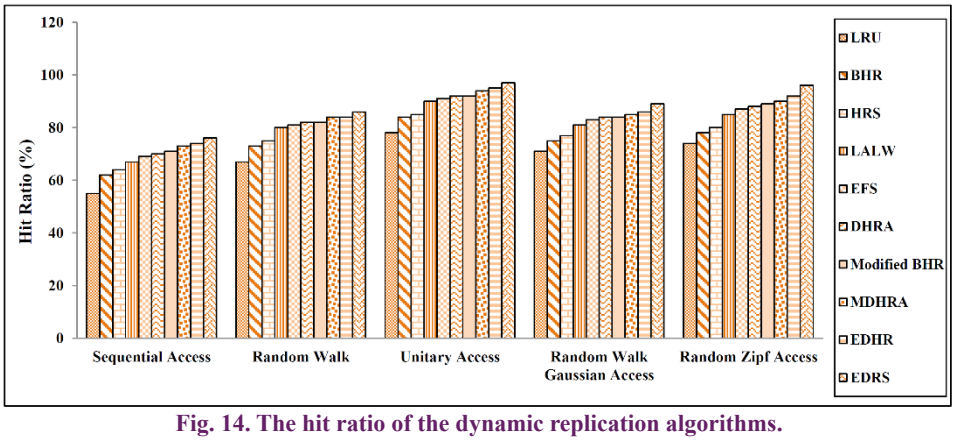

Fig. 14. The hit ratio of the dynamic replication algorithms.

renders it infeasible in the real world. Since in EDRS a small number of replicas is necessary to be transfered, the network communication cost is not very affected.

Comparison of Hit ratio in Fig. 14 reveals that EDhas RS the maximum Hit ratio. The proposed algorithm is able to increase the total number of local accesses by placing the replica in the most appropriate site and avoiding unnecessary replication. Therefore, a total number of replications and remote accesses have been decreased and consequently the hit ratio has been increased. As shown in Fig. 14, EDRS effectively outperforms the others in the total objective function value under the extended domain of system loads, including heavy system load. This can be related to EDRS ability to consider the differences between intra-LAN and inter-LAN bandwidth in all steps of the replication process.

5- Conclusion

One of the main challenge in data grid is selecting the optimal replica site for data retrieval based on job constraints. Most of previous replica selection algorithms only used response time as a metric in their decisions. In this study, we proposed a dynamic replica management strategy. Our replica management strategy consists of the dynamic replica creation method that can automatically increase replicas according to the frequency of the file access. The replica selection method is based on the response time and security, and the replica replacement method is based on the availability of the file, the last time at which the replica was requested, the number of accesses, and the sze of replica. The proposed strategy was evaluated with OptorSim that is provided by European Data Grid projects. We compared EDRS with eleven current algorithms, namely, No replication, LRU, LFU, HRS, LALW, BHR, EFS, DHRA, Modified BHR, MDHRA and EDHR. Average job execution time, ENU, SE usage, frequency of replication, and hit ratio were used as the performance evaluation criteria. We compared simulation results under different file access patterns. The results demonstrated that EDRS has the best performance compared with others in all the access patterns, especially in the different Random file access patterns. However, there remains two interesting problems that might be investigated in the future studies. First, the proposed replication strategy should be evaluated in real data grid infrastructures. Second, replica consistency may benefit from our strategy because it can improve the QoS; hence, further investigation is necessary to study the effect of conflict among copies in a distributed system.

References

[1] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, The data grid: towards an architecture for the distributed management and analysis of large scientific datasets, Journal of Network and Computer Applications, 23 (2001) 187-200.

[2] N. Rathore, I. Chana, Variable threshold-based hierarchical load balancing technique in Grid, Engineering with Computers, 31(3) (2014) 597-615.

[3] A.S. Saleh, An efficient system-oriented grid scheduler based on a fuzzy matchmaking approach, Engineering with Computers, 29 (2013) 185-206.

[4] T. Hamrouni, S. Slimani, F. Ben Charrada, A survey of dynamic replication and replica selection strategies based on data mining techniques in data grids, Engineering Applications of Artificial Intelligence, 48 (2016) 140-158.

[5] E. Gallicchio, J. Xia, W.F. Flynn, B. Zhang, S. Samlalsingh, A. Mentes, R.M. Levy, Asynchronous replica exchange software for grid and heterogeneous computing, Computer Physics Communications, 196 (2015) 236-246.

[6] S. Warhade, P. Dahiwale, M.M. Raghuwanshi, A dynamic data replication in grid system, in: 1st International Conference on Information Security & Privacy, 2016, 537-543.

[7] T. Hamrouni, S. Slimani, Faouzi Ben Charrada, A data mining correlated patterns-based periodic decentralized replication strategy for data grids, Journal of Systems and Software, 110 (2015) 10-27.

[8] E.U. Munir, J. Li, S. Shi, QoS suffrage heuristic for independent task scheduling in grid, Journal of Information Technology, 6 (2007)1166-1170.

[9] OptorSim–A Replica Optimizer Simulation: http://edg-wp2.web.cern.ch/edgwp2/ optimization/optorsim.html

[10] S. Goel, R. Buyya, Data replication strategies in wide-area distributed systems, Enterprise Service Computing: From Concept to Deployment, Idea Group Inc., Hershey, (2006) 211-241.

[11] Y. Saito, M. Shapiro, Optimistic replication, ACM Computing Surveys, 37(1) (2005) 42-81.

[12] I. Foster, K. Ranganathan, Design and evaluation of dynamic replication strategies for high performance data grids, in: Proceedings of International Conference on Computing in High Energy and Nuclear Physics, 2001.

[13] I. Foster, K. Ranganathan, Identifying dynamic replication strategies for high performance data grids, in: Proceedings of 3rd IEEE/ACM International Workshop on Grid Computing, 2002, pp. 75-86.

[14] I. Foster, K. Ranganathan, Decoupling computation and data scheduling in distributed data-intensive applications, in: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, HPDC-11, IEEE, CS Press, Edinburgh, UK, 2002, pp. 352-358.

[15] M. Bsoul, A. Al-Khasawneh, E.E. Abdallah, Y. Kilani, Enhanced fast spread replication strategy for data grid, Journal of Network and Computer Applications, 34 (2011) 575-580.

[16] K. Sashi, A.S. Thanamani, Dynamic replica management for data grid, IACSIT International Journal of Engineering and Technology, 2 (2010) 329-333.

[17] R.S. Chang, H.P. Chang, A Dynamic data replication strategy using access-weight in data grids, The Journal of Supercomputing, 45 (2008) 277-295.

[18] S.M. Park, J.H. Kim, Y.B. Ko, W.S. Yoon, Dynamic grid replication strategy based on internet hierarchy, in: International Workshop on Grid and Cooperative Computing, 1001 (2003) 1324-1331.

[19] K. Sashi, A.S. Thanamani, Dynamic replication in a data grid using a Modified BHR region based algorithm, Future Generation Computer Systems, 27 (2011) 202-210.

[20] A. Horri, R. Sepahvand, G.H. Dastghaibyfard, A hierarchical scheduling and replication strategy,

International Journal of Computer Science and Network Security, 8 (2008).

[21] N. Mansouri, G.H. Dastghaibyfard, Job scheduling and dynamic data replication in data grid environment, The Journal of Supercomputing, 64 (2013) 204-225.

[22] R. Chang, J. Chang, S. Lin, Job scheduling and data replication on data grids, Future Generation Computer Systems, 23 (2007) 846-860.

[23] N. Mansouri, G.H. Dastghaibyfard, A dynamic replica management strategy in data grid, Journal of Network and Computer Applications, 35 (2012) 1297-1303.

[24] N. Mansouri, G.H. Dastghaibyfard, E. Mansouri, Combination of data replication and scheduling algorithm for improving data availability in Data Grids, Journal of Network and Computer Applications, 36 (2013) 711-722.

[25] N. Mansouri, G.H. Dastghaibyfard, Enhanced dynamic hierarchical replication and weighted scheduling strategy in data grid, Journal of Parallel and Distributed Computing, 73 (2013) 534-543.

[26] C. Wang, C. Hsu, P. Liu, H. Chen, J. Wu, Optimizing server placement in hierarchical grid environments, The Journal of Supercomputing, 42 (2007) 267-282.

[27] C. Yang, C. Fu, C. Hsu, File replication, maintenance, and consistency management services in data grids, The Journal of Supercomputing, 53 (2010) 411-439.

[28] R.M. Rahman, R. Alhajj, K. Barker, Replica selection strategies in data grid, Journal of Parallel and Distributed Computing, 68 (2008) 1561-1574.

[29] R. Vingralek, Y. Breitbart, M. Sayal, P. Scheuermann, Web++: a system for fast and reliable web service, in: Proceedings of the USENIX Annual Technical Conference, 1999.

[30] M. Sayal, Y. Breitbart, P. Scheuermann, R. Vingralek, Selection algorithms for replicated web servers, in: Proceedings of the Workshop on Internet Server Performance, 1998.

[31] Load Balancing System, Chapter 6 in Intel Solutions Manual, Intel Corporation, 49-67.

[32] R. M. Almuttairi, R. Wankar, A. Negi, R. Rao Chillarige, M.S. Almahna, New replica selection technique for binding replica sites in data grids, in: 1st International Conference on Energy, Power and Control (EPC-IQ), 2010, pp. 187-194.

[33] S. Lewontin, E. Martin, Client side load balancing for the web, in: Proceedings of 6th International World Wide Web Conference, 1997, pp. 7-11.

[34] Z. Fei, S. Bhattacharjee, E. Zegura, M. Ammar, A novel server selection technique for improving response time of a replicated service, in: Proceedings IEEE INFOCOM, 1998, pp. 783-791.

[35] G. Bingxiang ,Y. Kui, a global dynamic scheduling with replica selection algorithm using GridFTP, in: International Conference on Challenges in Environmental Science and Computer Engineering, 2010, pp. 106-109.

[36] M. Sayal, Y. Breitbart, P. Scheuermann, R. Vingralek, Selection algorithms for replicated web servers, in: Proceedings of the Workshop on Internet Server Performance, Wisconsin, 1998.

[37] T. Ceryen, M. Kevin, Performance characterization of decentralized algorithms for replica selection in distributed object systems, in: Proceedings of the 5th International Workshop on Software Performance, 2005, pp. 257-262.

[38] B. Kusý, P. Dutta, P. Levis, Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services, Int. J. Ad Hoc and Ubiquitous Computing, 1 (2006) 1-14.

[39] H. Hamad, E. AL-Mistarihi, C. Huah Yong, Response time optimization for replica selection service in data grids, Journal of Computer Science, 4 (2008) 487-493.

[40] D.G. Cameron, R. Carvajal-schiaffino, A. Paul Millar, C. Nicholson, K. Stockinger, F. Zini, UK Grid Simulation with OptorSim, UK e-Science All Hands Meeting, (2003).

Please cite this article using:

N. Mansouri, M. M. Javidi, An Efficient Data Replication Strategy in Large-Scale Data Grid Environments Based on Availability and Popularity, *AUT J. Model. Simul. Eng.*, 50(1)(2018) 39-50.

DOI: 10.22060/miscj.2017.12236.5020