Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 7, No. 4, November 2016), Pages: 79-93
www.jacr.iausari.ac.ir

A Utility-Based Data Replication Algorithm in Large Scale Data Grids

Najme Mansouri
Young Researchers and Elite Club, Sirjan Branch, Islamic Azad University, Sirjan, Iran
najme.mansouri@gmail.com

Received: 2016/03/14; Accepted: 2016/06/19

Abstract

Data grids support access to widely distributed storage for large numbers of users accessing potentially many files. To enhance access time, replication at nearby sites may be used. Data replication, a technique much investigated by data grid researchers in past years creates multiple replicas of file and places them in conventional locations to shorten file access times. One of the problems in data replication is creation of replicas, replica placement and replica selection. Dynamic creation of replicas in an appropriate site by data replication strategy can increase the systems performance. In this paper, we propose a data replication algorithm, called the Utility-base Data Replication (UDR) algorithm that improves file access time. Each grid site has its own capabilities and characteristics; therefore, choosing one specific site from many sites that have the needed data is a key and significant decision. The replica selection problem has been studied by many researchers who only considered response time as a criterion for the selection process. Therefore, in this study, we addressed the problem of how to select the best replica for the users' jobs. Our approach is simulated using a data grid simulator, OptorSim, developed by European Data Grid projects. Comparing to the previous work the experimentation shows the improvement in the overall performance.

Keywords: *Data Grid, Data Replication, Simulation, Replica Selection*

1. Introduction

In recent years, many large-scale scientific systems [1–5] and commercial applications [6] such as the Large Hadron Collider (LHC) [7], the Data Grid Project (EDG) [8], image processing, physics Data Grids [9, 10], and data mining deal with large volume of data in the terabyte and even petabyte range and require increasing amounts of computing power, network bandwidth, and storage capacity for optimum performance [11-13]. A large number of storage elements and computational resources are integrated to generate a grid which provides us shared access to computing and storage resources such as CPUs, memory, and hard disks [14]. In particular, Data Grid is a type of grid which gives services and infrastructure that facilitate discovery, transfer and manipulation of huge amounts of data, as well as creation and management of copies of these data sets [15-17]. According to the Pareto principle (also known as the 80/20 rule) [18], some particular of Data Grid files are regularly accessed and transferred. In addition storing data on a central server causes problems such as single point of failure and bottleneck. Hence, this huge amount of data should be replicated and stored in various locations of distributed system to avoid such problems. Data

replication is a practical and effective approach to achieve efficient and fault-tolerant data access in grids.

In this paper, a Utility-base Data Replication (UDR) algorithm is proposed to decrease the job execution time and increase the data availability in the system. Restricted by the storage capacity, it is essential to present an effective strategy for the replication replacement task. We propose a new replacement strategy which considers four key factors: the number of requests in the future times, the size of the replica, availability, and the last time the replica was requested.

The main concept is: some files that were accessed more frequently in past will be accessed in future more than others. The strategy assigns an availability factor for each storage element in grid. This factor is symptom of probability of file existence. It is supposed that all files in a storage element have the same availability and in addition, each file has only one copy in storage element. Also, we are fallowing users behavior of requesting a file, and record the change to this request, whether growth or decay change. Then the average decay/growth rate is computed, and according to this average, the next number of access of this file is predicted. The proposed strategy is simulated in OptorSim and compared with various replica strategies. The simulation results show the better performance of our algorithm than former ones.

The rest of the paper is organized as follows: Section 2 introduces the Data Grid elements. In Section3 the replica management system is briefly explained. Section 4 gives an overview of pervious work on data replication algorithms. Section 5 presents the proposed replication strategy. We show and analyze the simulation results in section 6. Finally, section 7 concludes the paper and suggests some directions for future work.

2. Data Grid Elements

Grids can primarily be divided [2-4] into different types, depending on the nature of their emphasis: computation, data, application service, interaction, knowledge, and utility.

- Computational Grids are used for computationally intensive applications that require small amounts of data (e.g., TeraGrid, ChinaGrid, and APACGrid).
- Data Grids deal with the applications that require studying and analyzing massive data sets (e.g., LHCGrid, Gri-PhyN).
- Application service grids concentrate on providing access to remote applications; libraries hosted on data centers or computational grids (e.g., NetSolve and Grid- Solve).
- Interaction grids concentrate on interaction and collaborative visualization between participants (e.g., AccessGrid).
- Knowledge grids target toward knowledge acquisition, analyzing, and management, and provide business analytics services driven by integrated data mining services.
- Utility grids concentrate on presenting all the grid services, comprising compute power, data, and service to end users as IT utilities on a subscription basis, and offer the infrastructure essential for negotiation of needed quality of service, establishment and control of contracts, and allocation of resources to meet competing requests.

Data Grids consist of four main elements as shown in Fig. 1. The first element is from the point of view of Data Grid organization. This classifies ongoing scientific Data Grid efforts worldwide. The next element deals with the transport technologies applied

within Data Grids. This not only consists of well-known file transfer protocols but also includes other techniques of managing data transportation.

A scalable, robust, and intelligent replication strategy is a key operation of a Data Grid and the subtaxonomy presented next takes into account concerns of Grid system such as metadata and the nature of data transfer strategy used. The last element categorizes resource allocation and scheduling strategy and looks into issues such as locality of data. While each of the felids of data transport, replica management, and resource management, are independent are as of study and merit detailed investigations on their own, in this paper, replica management is presented from the point of view of the specific requirements of Data Grid environments.

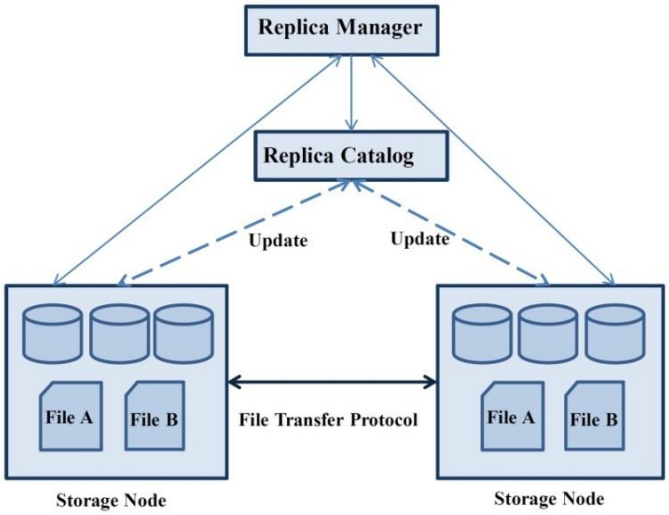

Figure. 1. A replica management architecture [4].

3. Replica Management System

Replication of the data is therefore an important method to guarantee scalability of the collaboration, reliability of data access and to preserve bandwidth. Replication is limited by the size of storage available at various sites within the Data Grid and the bandwidth between these locations. A replica management system therefore ensures access to the needed data while managing the underlying storage.

A replica management system, presented in Fig. 1, includes storage nodes which are connected to each other using high-performance data transport protocols.

The replica manager directs the generation and management of replicas according to the requests of the users and the availability of storage, and a catalog or a directory maintains track of the replicas and their locations. The catalog can be requested by applications to find the number and the locations of available replicas of a particular dataset. In some environment, the manager and the catalog are combined into one entity. Client-side software mainly consists of a library that can be merged into applications and a set of commands or GUI utilities that are built on top of the libraries. The client libraries allow querying of the catalog to find data files and to request replication of a specific data file.

4. Related Works

Foster and Ranganathan [19], 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, too) and Spatial locality (The related files to recently accessed file are likely to be requested in the near future). These strategies 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 pattern needs different replica strategies. Cascading and Fast Spread performed the best in the simulations. Also, the authors combined different scheduling and replication strategies.

Sashi and Thanamani [20] have extended Latest Access Largest Weight (LALW) strategy [21] where the replicas are created based on their weights. LALW algorithm gives higher weighs to recently requested files and replica placement were done only in cluster levels and not in the site levels. But Sashi et al. algorithm minimizes mean job execution time by placing the replicas in best site within the cluster by considering number of file requests and response time.

Park et al. [22] 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. So 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 replicated files are placed in all the requested sites not the appropriate sites. BHR strategy has good performance only when the capacity of storage element is small.

Mansouri and Dastghaibyfard [23] presented a Dynamic Hierarchical Replication (DHR) strategy that store replica in suitable sites where the particular file has been accessed most, instead of storing 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 that waiting in the storage and data transfer time. The simulation results show, 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 works, although DHR makes 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. We proposed in [24] Modified Dynamic Hierarchical Replication Algorithm (MDHRA) that improves DHR strategy. MDHRA deletes files in two steps when free space is not enough for the new replica: First, it deletes those files with minimum time for transferring (i.e. only files that are exist in local LAN and local region). Second, if space is still insufficient then it uses three important factors into replacement decision: the last time the replica was requested, number of access, and file size of replica. It also improves access latency by selecting the best replica when various sites hold replicas. The proposed replica selection selects the best replica location from among many

replicas based on response time that can be determined by considering the data transfer time, the storage access latency, the replica request waiting in the storage queue and the distance between nodes. Also a novel job scheduling algorithm called Combined Scheduling Strategy (CSS) is proposed in [24] that uses 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 job and the computing capacity of sites.

Khanli et al. [25] proposed a new dynamic replication strategy in a multi-tier data grid called predictive hierarchical fast spread (PHFS) which is an extended version of fast spread (a dynamic replication method in the data grid). Considering spatial locality, PHFS tries to predict future needs and pre-replicates them in hierarchal manner to increase locality in accesses and consequently improves performance. PHFS not only replicates data objects hierarchically in different layers of the multi-tier data grid for obtaining more localities in accesses but also optimized the usage of storage resources. But the authors just compared PHFS and CFS (common fast spread) with an example from the perspective of access latency. Therefore we implemented their strategy using OptorSim and compared it with other data replication strategies.

5. Utility-Based Dynamic Replication Strategy

When a job is allocated to local scheduler, before job execution the replica manager should transfer all the required files that are not available. Some of data files, are more desirable that those are called, popular files, whereas some other files will be rarely used. If the popular files can be find and copied into the required sites, then a great stage are taken to decrease the data access and therefore decrease the job execution time. We explained proposed replication strategy in three sections:

Replica selection: Typically Data Grid sites vary in their capabilities and their abilities to provide different levels of QoS. Two important factors are used to choose a best replica:

- Storage access latency

The storage media speed has a key role in the average response time. *T* can be calculated by the following equation:

$$T = \frac{FileSize(MB)}{StorageSpeed(MB/Sec)} \quad (1)$$

- Distance between nodes

D(x,y) represents network distance between nodes *x* and y. Computed using the number of hops with a trace route command. To reduce the cost, distance information can be stored when a replica is checked for the first time.

- Network performance

N(x,y) shows network and protocol performance between nodes *x* and *y*. Computed using network bandwidth (in Mbits/s) and latency among nodes *x* and *y*, as shown below.

$$\boldsymbol{N}\left(\boldsymbol{x},\boldsymbol{y}\right)=\frac{\boldsymbol{Latency}\left(\boldsymbol{x},\boldsymbol{y}\right)}{\boldsymbol{Bandwidth}\left(\boldsymbol{x},\boldsymbol{y}\right)} \tag{2}$$

$$\boldsymbol{Score}=\boldsymbol{w}_1\times\boldsymbol{T}+\boldsymbol{w}_2\times\boldsymbol{D}\left(\boldsymbol{x},\boldsymbol{y}\right)+\boldsymbol{N}\left(\boldsymbol{x},\boldsymbol{y}\right) \tag{3}$$

This function can be tailored, because it is defined as a weighted combination of the two former metrics. If several sites have the replica of f, it selects one that has minimum Score.

Replica placement: To improve the system reliability and performance, each file can has some copy in grids that in this case each one of those replicas must be saved in different storage elements. Because saving some replicas of one file in one storage element, not only don't help to increase file's availability but also will consumed huge amount of storage space.

If the requested file exists in the storage element, there isn't any need to replicate and copy it. Since as mentioned before, various replications of the file in storage element don't enhance the availability. In contrast it can cause to waste the storage space. But, if the requested file doesn't exist in the storage element, the file replication will be done. Now UDR places the replica in the Best Storage Element (BSE). To select the BSE, UDR finds SE with minimum Value-SE (*VSE*). In the calculation of *VSE* the frequency of requests of the replica and the last time the replica was requested are considered. These parameters are important because they give an indication of the probability of requesting the replica again.

$$VSE=(CT-LT_i)+\frac{1}{FR_i} \tag{4}$$

Where *CT* is the current time, LT_i is the last request time of replica *i*, and FR_i is the frequency of requests of the replica *i*.

Replica replacement: If enough space for replication does not exist, one or more files should be candidate for replacement stage using the following formula:

$$CM=\frac{N}{S}+\frac{1}{CT-LT}+\frac{1}{P} \tag{5}$$

Where *N* is number of access for the file in future time based on exponential growth/decay. S is the size of particular replica. *CT* is the current time, *LT* is the last request time of particular replica, *P* is data availability. Replicas that are available in BSE sorted based *CM* value in ascending order for deletion.

Now some of these important parameters are explained:

Availability (*P*): Each storage element has the data availability that is indicator of possibility of existing one file in it. Also it is assumed that all the saved files in storage element have the same availability. The file availability in the storage element *j* is shown by PSE_j. Since it is possible that there were more than one copy of file, so the availability of each fi file that is shown by P_i will be obtained in this equation:

$$P_i=1-\prod_{j=1}^{N}(1-p_{SEi}) \tag{6}$$

N shows the number of fi files copies. It is obvious that for each operation of accessing to a file, the possibility of unavailability is obtained form (1-P_i) junction, of

course with this supposition that the access operation of files will done separate from each other [26].

Number of access in future (*N*): We use the concept of exponential decay to predict the next number of access for the file. Many real world phenomena such as bacteria, radioactive isotopes, and credit payments can be modeled by functions that explain how things grow or decay as time passes. Exponential growth/decay is a growth in which the rate of growth is proportional to the current size. This model can be used in access history as well, since each file has number of access that increases by the increase of access rate and vice versa. We explain an exponential growth/decay principle for an access number of files in access history. The process of accessing files in Data Grid environment obeys an exponential model. If n_0 is the number of access for the file *f* at time *t*, and n(t) is the number of access for the same file at time *t+1* (just after the first access). The exponential decay/growth model is defined by the equation:

$$n(t) = n_0 \times e^{-rt} \tag{7}$$

Suppose *T* is the number of intervals passed, *F* is the set of files that have been demanded and ${n^t}_f$ represents the number of access for the file *f* at time interval *t*, and then we acquire the sequence of the access numbers:

${n^0}_f \;\; {n^1}_f \;\; {n^2}_f \ldots\ldots\ldots\ldots {n^{T-1}}_f \;\; {n^T}_f$

Therefore, according to the exponential decay/growth model we have:

${n^T}_f = {n^{T-1}}_f * e^{\alpha_{T-1}}$ This implies that $\alpha_{T-1} = \ln \frac{{n^T}_f}{{n^{T-1}}_f}$

So, the average rate for all intervals is

$$\alpha = \frac{\sum_{i=0}^{T-1} \alpha_i}{T} \tag{8}$$

We can predict the number of access for next time interval:

$${\alpha^{T+1}}_f = {\alpha^T}_f e^{\alpha} \tag{9}$$

For example, we use exponential model to find the next number of access for file *A*. If 23, 20, 12, 10 are number of access for file *A* during four intervals respectively then first we have to compute the average decay/growth rate for file *A*.

$$\alpha = \frac{\ln \frac{20}{23} + \ln \frac{12}{20} + \frac{10}{12}}{3} = -0.27$$

Finally estimation of next number of access for file *A* is:

$$a_A^5 = 10 \times e^{-0.27} = 7.6 \approx 8$$

The replacement will be done just in situations that the value of saving new copy be more than the expense of deleting the existed files. Now, the value of replicating fi file is obtained by

$$\Delta p_i = (p_i' - p_i) \times N_i \quad (10)$$

That P_i is the present availability of f_i file and P_i' is the availability of f_i file after replication.

Also the expense of deleting the candidate files will be obtained by the equation

$$\sum_{j \in Candidates} (P_j' - P_j) \times N_j \quad (11)$$

That P_j is the present availability of candidate file and P_j' is the availability after deleting the candidate file.

Therefore, the requested file fi replicated in BSE if the value gained by replicating f_i is greater than the accumulative value loss by deleting the candidates file from the BSE. Where

$$\Delta p_i \times N_i > \sum_{j \in Candidates} (P_j' - P_j) \times N_j \quad (12)$$

Figure 2 describes the replacement strategy.

Create list L from files in BSE and for each file compute *CM* (equation 4);
Sort list L in ascending order of *CM*;
Sum=0;
While (L is not empty && not enough space for new replica) {
    Select file f_i from list L;
    Sum = Sum + $(p_i' - p_i) \times N_i$;
}
If $(\Delta p_F \times N_F > Sum)$ { delete candidates files from list L in BSE; }
If (enough space exist for new replica of F in BSE) { store new replica of F; }

Figure 2. Replacement strategy.

6. Experiments

In this section, elements of Grid simulation, network configuration and the simulation results are described.

6.1. Elements of Grid Simulation

We have implemented the proposed strategy using OptorSim, a simulator for Data Grids. It provides users with the Data Grids simulated architecture and programming interfaces to analysis and validate their strategies. In order to obtain a realistic simulated environment, there are a number of components which are included in OptorSim. These include Computing Elements (CEs), Storage Elements (SEs), Resource Broker (RB), Replica Manager (RM), and Replica Optimiser (RO). Each site consists of zero or more CEs and zero or more SEs (as shown in Fig 3.)

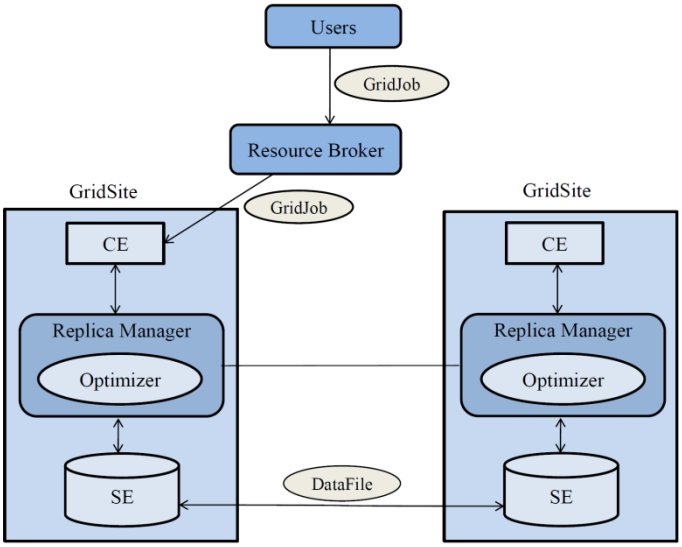

Figure 3. OptorSim architecture [27].

6.2 Configuration

The study of our scheduling algorithm is carried out using a model of the EU Data Grid Testbed [27] sites and their associated network geometry as shown in Fig. 4. Initially all jobs are placed on CERN (European Organization for Nuclear Research) storage element. CERN contains original copy of some data sample files that cannot be removed.

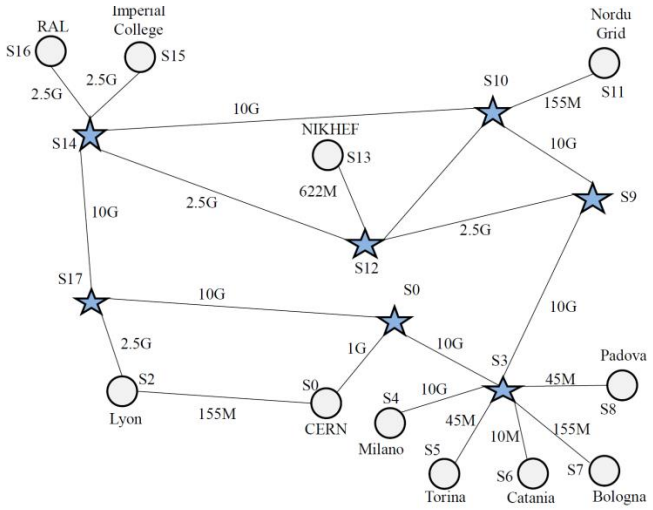

Figure 4. The gird topology of EDG [27].

Since all files are available in Site 0, so any job sent to this site does not require any file transfer. Therefore in our simulation we only consider all CE sites except site 0. Each file is set to be 1 GB. To record file transfer time and path, we changed OptorSim code. The simulation parameter values appear in Table 1. A job will typically request a set of logical filename(s) for data access. The order in which the files are requested is specified by the access pattern. We considered three different access patterns: sequential (files are accessed in the order stated in the job configuration file), unitary random (file

requests are one element away from previous file request but the direction is random), and Random Zipf access (given by $P_i = K/ i^s$, where P_i is the frequency of the ith ranked item, K is the popularity of the most frequently accessed data item and S determines the shape of the distribution). Data replication strategies commonly assume that the data is read-only in Data Grid environments.

Table 1. Simulation parameter values.

Description	Value
Number of files	200
File size	1 G
Storage available at an SE	30 G–100000 G
Number of jobs	10000
Number of files accessed by a job	3–20

6.3 Simulation Results and Discussion

In order to evaluate the effectiveness of the different replication strategies implemented in OptorSim, we used the following metrics:

- Total job execution time;
- Effective Network Usage;
- System File Missing Rate

Total Job Execution Time: The total job time consists of the time of data transferring and job execution. This is a typical Grid user would probably evaluate it to be the most significant metric of how the algorithm is working.

Effective Network Usage: ENU is used to estimate the efficiency the network resource usage. Effective Network Usage (E_{enu}) is given from [27]:

$$E_{enu} = \frac{N_{rfa} + N_{fa}}{N_{lfa}} \tag{13}$$

Where N_{rfa} is the number of access times 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. The effective network usage ranges from 0 to 1. A lower value represents that the network bandwidth is used more efficiently.

System File Missing Rate: *SFMR*— indicates the ratio of the number of files potentially unavailable and the number of all the files requested by all the jobs. System File Missing Rate and it is defined as follows [34] to measure the data availability:

$$SFMR = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m_j}(1-P_i)}{\sum_{j=1}^{n} m_j} \tag{14}$$

Where n indicates the total number of jobs. m_j indicates the number of file access operation of each job. Pi shows the probability of availability of file f_i as defined in equation (5). It substantiated that smaller value for *SFMR* show better system data availability [34].

Figure 5 shows the mean job time of the eight dynamic replication strategies with three different access patterns: Unitary Random, Sequential and Random Zipf distribution. The Least Frequently Used (LFU) strategy always replicates files in the site

where the job is executing. If there is not enough space for new replica, least accessed file in the storage element is deleted. In Least Recently Used (LRU) strategy always replication takes place in the site where the job is executing. If there is not enough space for the new replica, the oldest file in the storage element is deleted. Bandwidth Hierarchy based Replication (BHR) strategy stores the replicas in a site that has a high bandwidth and replicates those files that are likely to be requested soon within the region. Since size of SE at each site, is not enough to store large portion of all data, we cannot have much performance improvements with site-level replacement policy.

Therefore, BHR strategy takes benefit from network-level locality by storing several files in a region as many as possible. The results for LFU and LRU are similar, and we only include those for LRU in the remaining figures. The mean job time in LALW is about 9% faster than that of BHR. LALW selects a popular file for replication. By associating a different weight to each historical data access record, the importance of each record is differentiated. The mean job time of DHR is lower by 12% compared to BHR algorithm with Sequential access pattern. Since it selects the best replica location for execution jobs with considering number of requests that waiting in the storage and data transfer time. The mean job time is about 27% faster using MDHRA than using LRU, and 11% faster than DHR with Zipf distribution. UDR has the lowest value of mean job execution time in all the experiments and all of file access patterns. As in Random access patterns comprising Unitary random and walk Random Zipf , a certain set of files is more likely to be requested by Grid sites, so a large percentage of requested files have been replicated before. Therefore, UDR strategy and also all the other strategies have more improvement for random file access patterns.

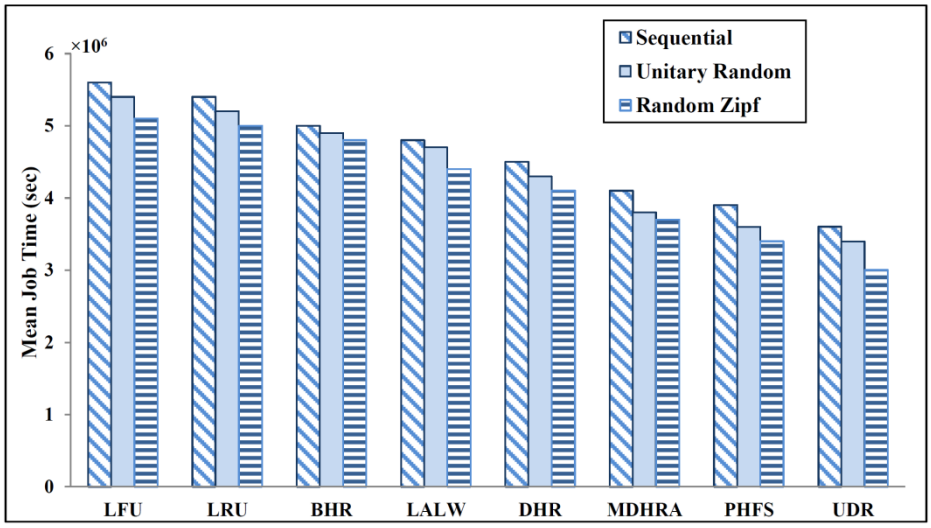

Figure 5. Mean job Time for different access patterns.

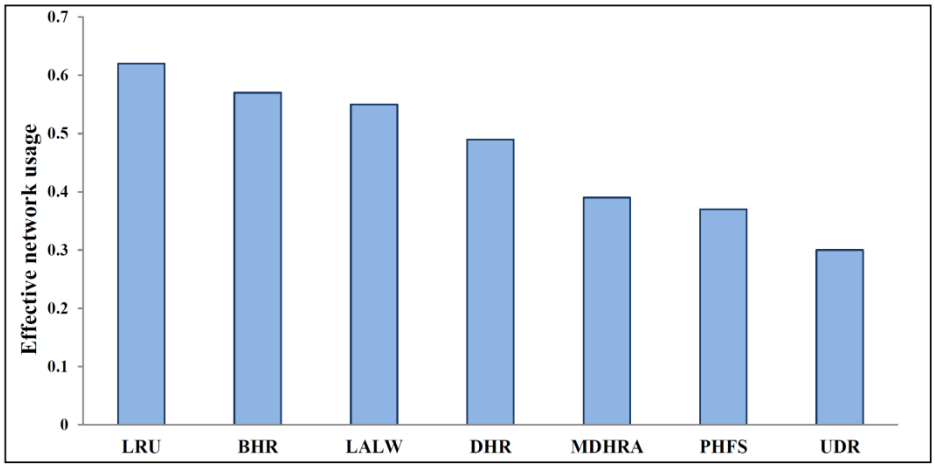

Figure 6. Effective network usage.

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. So, a good balance must be discovered, where any replication is in the interest of reducing future network traffic. ENU is effectively the ratio of files transferred to files requested, so a low value indicates that the strategy used is better at putting files in the right places. The effective network usage for the Random Zipf Access Pattern Generator is shown in Fig. 6. The ENU of PHFS is lower about 42% compared to the LRU strategy. The main reason is that Grid sites will have their needed files present at the time of need, hence the total number of replications will decrease and total number of local accesses increase. The UDR is optimized to minimize the bandwidth consumption and thus decrease the network traffic.

Figure 7 shows the amount of SFMR for each 8 increasing schemes and three access patterns. In this evaluation the random scheduler used to schedule the jobs. The LRU performs slightly better than LFU. It is obvious that UDR strategy for all of the access patterns performs better than others. This is because our UDR replica managers decide to make the replica only when the gain of the value from the replicated file is greater than the loss of the value of the replaced file.

The PHFS does not have a lower missing rate when compared to UDR. Because the prediction function is not as accurate, so it brings some inaccuracy into the calculation of the file weight. This, in turn, will cause the replica scheme to fail to work as well as.

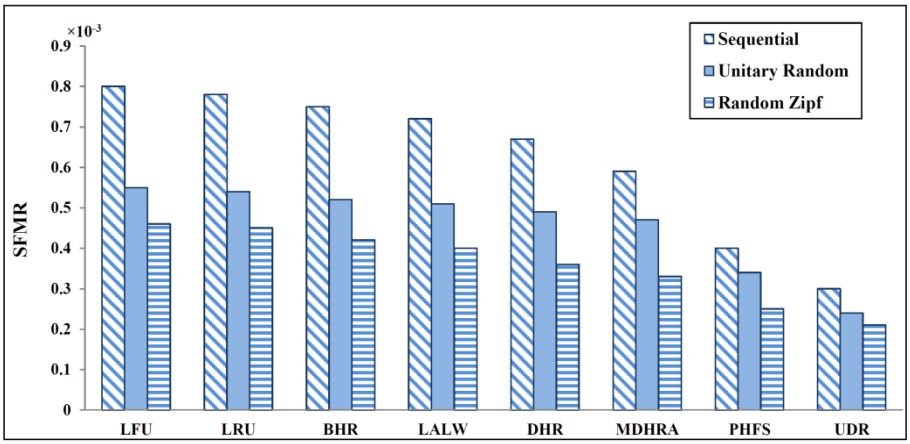

Figure7. SFMR with varying access patterns.

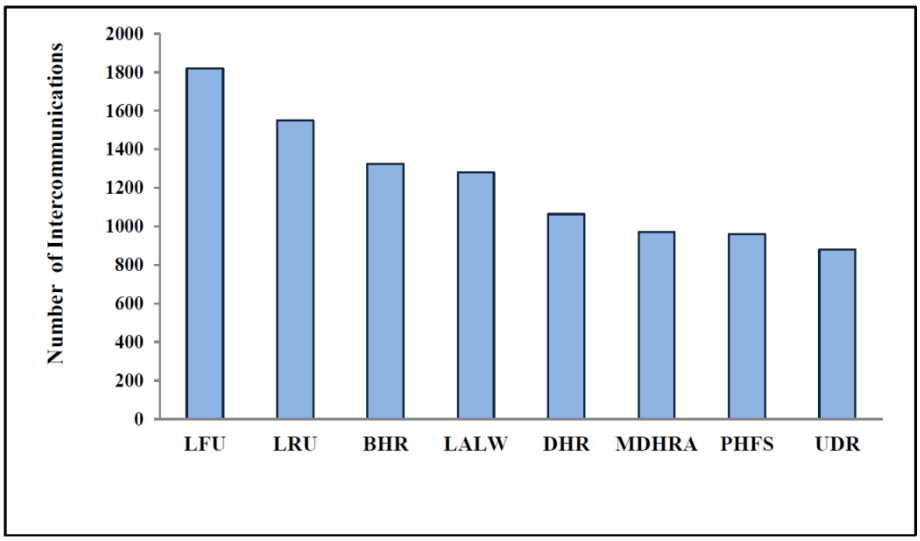

Figure. 8. The number of inter-communications.

Figure 8 shows the number of inter-communications for different replication strategies. According to the temporal and geographical locality, UDR stores the replica in the best site. It considers the frequency of requests of the replica and the last time the replica was requested. Therefore it can reduce the inter-communications between different sites.

7. Conclusion and Future Work

Replication not only reduces access time, but also improves data availability. A dynamic data replication strategy, called Utility-base Data Replication (UDR) is proposed. UDR selects the best replica location for execution jobs with considering three important factors: CPU, I/O and Bandwidth. It also stores the replicas in the best site where the file has been accessed for the most time instead of storing files in many sites. Therefore, sites will have their required files locally at the time of need and this will decrease response time, access latency, bandwidth consumption and increase system performance considerably.

Availability of files in distributed system obtains with a limited copy space condition. It minimize the data miss rate and maximize the availability of files, meanwhile with limited storage space and low "time to data access". The data files are sorted based on the weight factor and if the value of replicating a file is more than the loss of deleting the candidate files, the replication work will be done.

To evaluate the efficiency of policy, we use the Grid simulator OptorSim that is configured to represent a real world Data Grid testbed. We compared UDR algorithm to 7 of existing algorithms, LRU, LFU, BHR, DHR, LALW, MDHRA and PHFS for different file access patterns. The evaluation shows that UDR algorithm outperforms the other algorithms and improves Total Job Time and Effective Network Usage under different the access patterns, especially under the different random file access patterns. In our future work we will investigate in integrating users' preferences in the replica selection process.

References

[1] A. Folling, C. Grimme, J. Lepping, and A. Papaspyrou, "Robust load delegation in service grid environments," IEEE Transactions on Parallel and Distributed Systems, Vol. 21 (9), pp. 1304–1316, 2010.

[2] O. Sonmez, H. Mohamed, and D. Epema, "On the benefit of processor coallocation in multicluster grid systems," IEEE Transactions on Parallel and Distributed Systems, Vol. 21 (6), pp. 778–789, 2010.

[3] H. Li, "Realistic workload modeling and its performance impacts in large-scale science grids," IEEE Transactions on Parallel and Distributed Systems, Vol. 21 (4), pp. 480–493, 2010.

[4] S. Venugopal, R. Buyya, K. Ramamohanarao, "a taxonomy of data grids for distributed data sharing, management and processing," ACM Computing Surveys (CSUR), Vol. 38(1), 2006.

[5] N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, and E. Varvarigos, "Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing," Computer Communications, Vol. 30, pp.499–515, 2007.

[6] B. Tierney, W. Johnston, J. Lee, and M. Thompson, "A data intensive distributed computing architecture for grid applications," Future Generation Computer Systems, Vol. 16 (5), pp. 473–481, 2000.

[7] LHC accelerator project. http://www-td.fnal.gov/LHC/USLHC.html.

[8] European DataGrid Project (EDG). http://www.eu-egee.org.

[9] GriPhyN: The Grid physics network project, 12 July 2010. http://www.griphyn.org.

[10] PPDG. http://www.ppdg.net.

[11] R.S. Chang and M.S. Hu, "A resource discovery tree using bitmap for grids," Future Generation Computer Systems, Vol. 26 (1), pp. 29–37, 2010.

[12] J. Wu, X. Xu, P. Zhang, and C. Liu, "A novel multi-agent reinforcement learning approach for job scheduling in grid computing," Future Generation Computer Systems, Vol. 27 (5), pp. 430–439, 2011.

[13] S. Ebadi and L.M. Khanli, A new distributed and hierarchical mechanism for service discovery in a grid environment," Future Generation Computer Systems, Vol. 27 (6), pp. 836–842, 2011.

[14] K. Christodoulopoulos, V. Sourlas, I. Mpakolas, and E. Varvarigos, "A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in Grid networks," Computer Communications, Vol. 32, pp. 1172–1184, 2009.

[15] N. Mansouri, "Improve the Performance of Data Grids by Cost-Based Job Scheduling Strategy," Computer Engineering and Applications Journal, Vol. 3, pp. 101-111, 2014.

[16] J. Zhang, B.S. Lee, X. Tang, and C.K. Yeo, "A model to predict the optimal performance of the hierarchical data grid," Future Generation Computer Systems, Vol. 26 (1), pp. 1–11, 2010.

[17] N. Mansouri, "A Hybrid Approach for Scheduling based on Multi-criteria Decision Method in Data Grid," Computer Engineering and Applications Journal, Vol. 3, pp. 1-11, 2014.

[18] Z.W. Xu, H.J. Zhou, G.J. Li, "Usability Issues of Grid System Software," Journal of Computer Science and Technology, Vol. 21(5), pp. 641-647, 2006.

[19] I. Foster and K. Ranganathan, "Design and evaluation of dynamic replication strategies a high performance data grid," in Proceedings of International Conference on Computing in High Energy and Nuclear Physics, China 2001.

[20] K. Sashi and A.S. Thanamani, "Dynamic replica management for data grid," IACSIT International Journal of Engineering and Technology, Vol. 2, pp. 329-333, 2010.

[21] R.S. Chang and H.P. Chang, "A Dynamic data replication strategy using access-weight in data grids," Journal of Supercomputing, Vol. 45, pp. 277-295, 2008.

[22] S.M. Park, J.M. Kim, Y.B. Go, and W.S. Yoon, "Dynamic grid replication strategy based on internet hierarchy," in International Workshop on Grid and Cooperative Computing, Vol. 1001, pp. 1324-1331, 2003.

[23] N. Mansouri and G.H. Dastghaibyfard, "A dynamic replica management strategy in data grid," Journal of Network and Computer Application, 2012.

[24] N. Mansouri, G.H. Dastghaibyfard, and E. Mansouri, "Combination of data replication and scheduling algorithm for improving data availability in data grids," Journal of Network and Computer Applications, Vol. 36, pp. 711-722, 2013.

[25] L. Mohammad Khanli, A. Isazadeh, and T.N. Shishavan, "PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid," Future Generation Computer Systems, Vol. 27, pp. 233–244, 2011.

[26] M. Lei, S.V. Vrbsky, and X. Hong, "An on-line replication strategy to increase availability in data grid," Future Generation Computer Systems, Vol. 28, pp. 85-98, 2008.

[27] D.G. Cameron, A.P. Millar, C.C. Nicholson, R. Carvajal-Schiaffino, F. Zini, and K. Stockinger, "Optorsim: a simulation tool for scheduling and replica optimization in data grids," in International conference for computing in high energy and nuclear physics (CHEP'04), 2004.