Australian Journal of Basic and Applied Sciences, 6(10): 336-346, 2012
ISSN 1991-8178

An Effective Weighted Data Replication Strategy for Data Grid

Najme Mansouri

Department of Computer Science & Engineering, Birjand University of Technology, Postal Code 97175-569, Birjand, Iran

Abstract: Data Grid is a good solution to large scale data management problems including efficient file transfer and replication. Dynamic data replication in Data Grid aims to improve data access time and to utilize network and storage resources efficiently. Since the data files are very large and the Grid storages are limited, managing replicas in storages for the purpose of more effective utilization of them require more attention.In this paper, a dynamic data replication strategy, called Modified Latest Access Largest Weight (MLALW) is proposed. This strategy is an enhanced version of Latest Access Largest Weight strategy.MLALW deletes files by considering three important factors: least frequently used replicas, least recently used replicas and the size of the replica. MLALW stores each replica in an appropriate site i.e. appropriate site in the region that has the highest number of access in future for that particular replica. The algorithm is simulated using a Data Grid simulator, OptorSim, developed by European Data Grid projects. The experiment results show that MLALW strategy gives better performance compared to the other algorithms and prevents unnecessary creation of replica which leads to efficient storage usage.

Key words:

INTRODUCTION

Today's scientific and industrial world requires processing of huge amount of distributed data (Chakrabarti, A., S. Sengupta, 2008). Kesselman and Foster in 1998 defined a Grid as follows "A Computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities"(Foster, I.,C. Kesselman,2004).Later in 2002 they improved the previous definition in (Foster *et al*., 2004; Andronikou *et al*., 2012) as follows "A system that coordinates resources that are not subject to centralized control, using standard, open, general purpose protocols and interfaces to deliver non-trivial qualities of services". Grid can be divided as two parts, Computational Grid and Data Grid. Computational Grids are used for computationally intensive applications that require small amounts of data. But, Data Grid deals with the applications that require studying and analyzing large amount of data (terabytes or petabytes) (Chang, R.S. and M.S.Hu,2010; Wu*et al*., 2011; Ebadi, S.,L.M. Khanli,2011;Mansouri *et al*., 2011; Nemati *et al*., 2012). Managing this large amount of data in a centralized way is very difficult or even impossible (Nukarapu *et al*., 2011). Hence, such huge dataset must be separated and stored in several physical locations.

The main problems that degrade the performance of Data Grid are wide area networks latency and the internet bandwidth. Therefore, network latency and bandwidth between storage sites and processing sites has important role in Data Grid. In order to improve performance of Data Grids one needs to consider the followings:

- Scheduling optimization: It determines where to submit the job with respect to location of replicas and computational capabilities of the nodes.
- Short-term optimization: It decides from where to select replicas by considering available network bandwidth between nodes.
- Long-term optimization or dynamic replication strategy: It determines what replica files to store or remove when there is shortage of storage in a node.

Data replication is a practical and effective approach to achieve efficient and fault-tolerant data access in Grids. There are three key issues in all the data replication algorithms as follows:

- Replica selection: process of selecting replica among other copies that are spread across the Grid.
- Replica placement: process of selecting a Grid site to place the replica.
- Replica management: the process of creating or deleting replicas in Data Grid.

In Grid environment, where large number of users is sharing limited computing and storage resources, the optimisation of resource usage is very critical in order to reach reasonable execution time. Even though the memory and storage size of computers are ever increasing, they are still not keeping up with the request of storing large number of data. Hence methods needed to create replicas that increase availability without using unnecessary storage and bandwidth.In this paper, a ModifiedLatest Access Largest Weight (MLALW) is

Corresponding Author: Najme Mansouri, Department of Computer Science & Engineering, Birjand University of Technology, Postal Code 97175-569, Birjand, Iran
Tel: +98-915-3624299, E-mail: najme.mansouri@gmail.com, Fax: +98-561-2252001.

proposed to overcome the limitations of the LALW algorithm (Chang, R.S., H.P. Chang, 2008). According to our previous works, although LALW makes some improvements in some metrics of performance like mean job time, it shows two deficiencies:

(1) The LALW determines in which region the replica has to be placed and how many replica has to be placed. But LALW doesn't determine in which site within the region the file has to be placed. MLALW places replicas in two stages. In the first stage MLALW like LALW determines how many replicas have to be placed in each region. The second stage is to place the replica in the Best Site(BS) within the region. MLALW uses the concept of exponential decay to estimate the next number of access for the file. It stores replica in a best site that has the highest number of access for that particular replica in coming interval.

(2) In replica replacement step using LFU strategy may delete some valuable files that may not be available in local region and may be needed in future. Therefore, such deletions will result in a high cost of transfer.

MLALW considers three important factors into replacement decision: least frequently used replicas (LFU), least recently used replicas (LRU) and the size of the replica. The number of access in addition to the last time the replica was requested give an indication of the probability of requesting the replica again. Since the storage space is an important issue in the Data Grid, the size of replica is also a valuable parameter in deciding if the replica should be stored.

To evaluate various replica optimization strategies, various Grid simulators have been developed, such as MicroGrid (Song *et al*., 2000) Bricks (Takefusa *et al*., 1999) Simgrid (Casanova, 2001), GridSim (Buyya, R.,M. Murshed,2002) and Optorsim (Bell *et al*., 2003). Optorsim, based on the architecture of the EU DataGrid project, simulates a real DataGrid environment for researchers to investigate the effectiveness of replication strategy. The proposed strategy is simulated in OptorSim and compared with various replica strategies. The Mean Job Execution Time and Effective Network Usage (ENU) are the performance evaluation metrics in our simulation. The experiment results show the better performance of our strategy than former ones. The rest of the paper is organized as follows: In section 2 the classification of data replication strategies is briefly explained. Section 3 introduces related work of this study. Section 4 presents the proposed replication strategy. We show and analyze the simulation results in section 5. Finally, section 6 concludes the paper and suggests some directions for future work.

2. Classification of Data Replication Strategies:

The motivation for replication is how to enhance data availability, accessibility, reliability, and scalability. Generally, replication algorithms are either static or dynamic as shown in Fig. 1. In static approaches the created replica will exist in the same place till user deletes it manually or its duration is expired. The disadvantages of the static replication strategies are that they cannot adapt to changes in user behavior and they are not appropriate for huge amount of data and large number of users. Of course static replication methods have some advantages like: they do not have the overhead of dynamic algorithms and job scheduling is done quickly (Tatebe *et al*., 2002; Chervenak *et al*., 2002). On the other hand, dynamic strategies create and delete replicas according to the changes in Grid environments, i.e. users' file access pattern (Foster, I., K. Ranganathan, 2001;Cibej*et al*., 2005; Yuan*et al*., 2007; Lamehamedi *et al*., 2003; Lee, B.D., J.B. Weissman, 2001; Amjad *et al*., 2012). As Data Grids are dynamic environments and the requirements of users are variable during the time, dynamic replication is more appropriate for these systems (Ranganathan, K., I. Foster, 2001; Foster *et al*., 2001). But many transfers of huge amount of data that are a consequence of dynamic algorithm can lead to a strain on the network's resources. So, inessential replication should be avoided. A dynamic replication scheme may be implemented either in a centralized or in a distributed approach. These methods also have some drawbacks such as; the overload of central decision center further grows if the nodes in a Data Grid enter and leave frequently. In case of the decentralized manner, further synchronization is involved making the task hard.

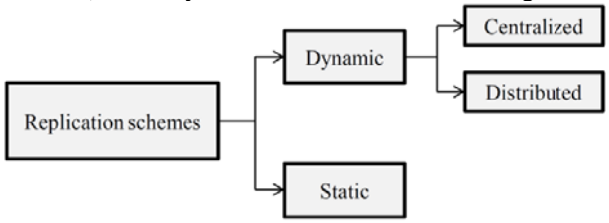

Fig. 1: Types of replication schemes.

3. Related Work:

Replication has been an interesting topic in World Wide Web, peer-to-peer networks, ad hoc and sensor networking, mesh networks and distributed databases (Wolfson *et al*., 1997; Rabinovich *et al*., 1998; Perez *et al*., 2010; Vrable *et al*., 2009; Shen H., 2010).

Foster and Ranganathan (Foster, I., K. Ranganathan, 2001), proposed six distinct replica strategies No Replica, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread) for multi-tier data. 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.
- 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.

Park *et al*. (2003) 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. Also, BHR strategy has good performance only when the capacity of storage element is small. Modified BHR (Sashi, K., A.S. Thanamani,2011) is an extension of BHR (Park *et al*., 2003) strategy which replicates a file that has been accessed most and it may also be used in near future. Searching all sites to find the best one is the main weakness of Modified BHR algorithm.

A replication algorithm for a 3-level hierarchy structure and a scheduling algorithm are proposed in (Horri *et al*., 2008) They considered a hierarchical network structure that has three levels. In their replication method among the candidate replicas they select the one that has the highest bandwidth to the requested file. Similarly, it uses the same technique for file deletion. This leads to a better performance comparing 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 Dastghaibyfard (2012) 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.

Sepahvand *et al*. (2011) presented a cosine similarity predictor function that uses the path similarity between the previous and current file transfer paths. They proposed new data replication and job scheduling algorithms using this predictor function. For scheduling algorithm two key factors are considered: estimated file transfer time and queue length. The simulation results show that the proposed replication and scheduling strategies outperform multi regression and neural network methods as the number of jobs increases.

Dan-wei *et al*. (2010) presented a node selection model based on the degree of distribution of complex networks, without considering the local environment. They determined two candidate replica nodes: a degree-based candidate pool and a frequency-based candidate pool. Considering the environment, it replicates data in the node with minimum local cost. Further, they defined and proved a replica creation theorem. Experimental results demonstrate that the proposed strategies can simultaneously reduce makespan and data storage consumption.

Jaradat *et al*. (2011) presented a new replica selection strategy called Balanced QoS Replica Selection Strategy (BQSS), that based on a sound mathematical model and supporting metric to measure balanced QoS time, availability and security (BTAS). They answered two important questions: How to find the best replica location from among many replicas distributed across the Grid sites in a high- quality, timely, and consistent balanced rates of QoS parameters. How to predict the value of the QoS parameters. The simulation performance results show that the BQSS performs best when compared to the competitor's D-system.

4. Proposed Replication Algorithm:

In this section, first network structure is described and then the LALW and MLALW algorithms are presented.

4.1. Network Structure:

Figure 2 shows the hierarchical architecture which has two levels similar to what is given in (Chang *et al*., 2008). We locate group of the Grid sites on the same network region. A network region is a network topologicalspace where sites are placed closely. There is a region master responsible for replica management.

There is aregion header used to manage the site information in a region. The region master gets the information of accessed files from all headers. Each site hasthe history of files accessed in that site. The access historyprovides the details of FileId, RegionId and the Time, whichshows that the file has been accessed by a site placed inthe region (RegionId) at a particular time. Each site sends its access history to its region header regularly.All information in the same region will be aggregated and summarized by the region header. The region master provides theglobal information based on the file and the number of timeit has been accessed.

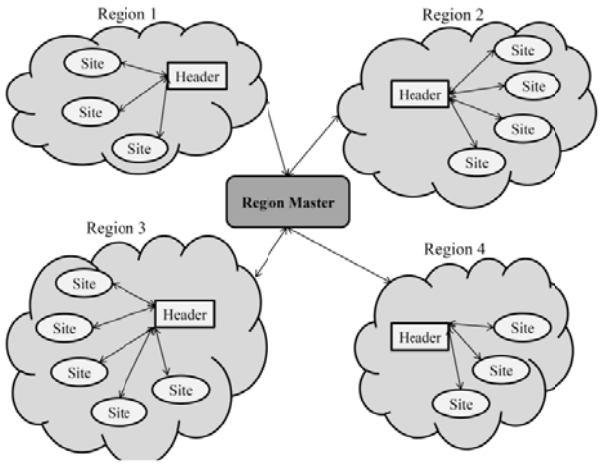

Fig. 2: The Hierarchical architecture.

4.2. LALW Strategy:

In (Chang *et al*., 2008)LALW strategy is proposed where the replicas are created dynamically based on the weights and placed in the regions. LALW strategy is briefly explained and more information about LALW algorithm can be found in (Chang *et al*., 2008). Replica creation in LALW strategy has two steps. First step determines which file to be replicated and the second step identifies the number of replicas. In first step region master collects the information from the region header at a constant time interval. Information collected at different time interval has different weights. The rule of setting weight employs the concept of Half-life which is mentioned in many fields, such as physics, chemistry. Here the weight indicates the quantity and the time interval. It assigns larger weight to the recently accessed files and less weight to the older files. Access Frequency (*AF*) represents the importance for access history in various time intervals. The Access Frequency for file *F* is defined by the equation:

$$AF(f)=\sum_{t=1}^{N_T}(a_f^t \times 2^{-(N_T-t)}), \forall f \in F \tag{1}$$

Where *NT* is the number of time interval passed, *F* is the set of files that have been requested and a_i^j represents the number of accesses for the file *i* at time interval *j*. The file with maximum *AF* is a popular file. The second step is to find the number of replicas needed. The number of replicas can be calculated by comparing the average access frequency of the popular file with all other files. Assuming *p* is the popular file. Then the average access frequency of the popular file *p* is given by

$$AF_{avg}(p)=\frac{AF(p)}{N_T} \tag{2}$$

The average Access Frequency of all other files is given by

$$AF_{avg}(f)=\frac{[AF(f)]_{sum}}{N_F \times N_T}, \forall f \in F \tag{3}$$

$AF(f)_{sum}$ indicates the sum of *AF* for all files requested and $NF=|F|$ is the number of different files that have been requested. The number of replicas needed for the popular file is calculated as

$$Num_{system}(p)=\left\lceil \frac{AF_{avg}(p)}{AF_{avg}(f)} \right\rceil \tag{4}$$

The region master requests to every region header to obtain the information about the popular file. The information involves the region details and the number field. These files still have various weights based on various time intervals. The Region master determines *AF(p)* in different regions after collecting the information for popular file from region headers. Then it sorts regions in descending order according to the *AF(p)*. The first region in the sorted list has the highest priority to have the replicas. Number of replicas to be placed at region *c* is

$$Num_c(p) = \left\lceil n \times \frac{AF_c(p)}{\left[AF_c(p)\right]_{sum}} \right\rceil, c = 1,2,...N \tag{5}$$

Where *n* is the number of replicas needed to be replicated. AF_c(p) represents the *AF* for the popular file *p* in region *c*, $[AF_c(p)]$sum is the sum of AF_c(p) for all regions.
Finally, if there is no space to store the replica Least Frequently Used (LFU) file is deleted from the site.

4.3. MLALW Strategy:

We describe ELALW strategy in three sections.

Replica Creation: At intervals, the proposed algorithm like LALW collects the information about accessed files from all headers. By calculating the product of weight and the number of accesses for a file, it considers a more precise metric to determine a popular file for replication. Then it calculates the number of replicas needed from Eq. (4).

Replica Placement: Replica placement has two stages. In the first stage MLALW like LALW determines how many replicas have to be placed in each region by using Eq. (5). The second stage is to place the replica in the Best Site (BS) within the region. To select the BSE, MLALW finds SE with maximum number of replica access in future.

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:

$$n(t) = n_0 \times e^{-rt} \tag{6}$$

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

$$\alpha = \frac{\sum_{i=0}^{T-1} \alpha_i}{T} \tag{7}$$

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

$$\alpha^{T+1}{}_f = \alpha^T{}_f e^{\alpha} \tag{8}$$

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:

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

Replica Replacement: If enough space for replication does not exist, a list of replicas needs to be removed from the storage. But what if that list of replicas that are to be deleted are more valuable than the new replica? MLALW strategy stores only the important replicas while the other less necessary replicas are replaced with more important replicas.In this step three valuable factors (LFU, LRU, and Replica Size) will be considered. Obviously, it is more important to replace files with large size, because it can reduce the number of replica replacement. The *Replica_Value* is calculated by equation

$$\text{Re}\,plica\_Value = S + LRU - LFU \tag{9}$$

$$S = \left(\frac{SR}{\sum_{i=1}^{n} SR} \times 100\% \right)$$

$$LFU = \left(\frac{NR}{\sum_{i=1}^{n} NR} \times 100\% \right) \tag{10}$$

$$LRU = \left(\frac{CT - LT}{\sum_{i=1}^{n} CT - LT} \times 100\% \right) \tag{11}$$

Where *SR* is replica size, *NR* is the number of access to the replica, *CT* is the current time and *LT* is the last request time of replica. Sort all remaining replicas according to *Replica*_Value in descending order and start deleting replicas from the above list till space is available for replica. In this replacement strategy, the underlying two main goals are achieved simultaneously, namely, data availability is increased by keeping the most important replicas and storage resource cost is reduced by removing unnecessary largest replicas.

5. Experiments:

5.1 SimulationTool:

OptorSim was developed to simulate the structure of a real Data Grid for evaluating various replication strategies. OptorSim is the project of EDG, a Java-based simulation language. Figure 3 shows the architecture of the Data Grids simulated by the OptorSim simulator (Cameron*et al*., 2004). OptorSim assumes that a Grid consists of several sites, each of which contains zero or more Computing Elements (CEs) which execute jobs and zero or more Storage Elements (SEs) which store files. Resource Broker (RB) accepts job submission from users and dispatches each job to proper site according to the scheduling algorithm, which collect some information to make an optimal decision. Replica Manager (RM) at each site controls data transferring and provides a mechanism for accessing the Replica Catalog. The Replica Optimiser (RO) within the RM is responsible for the selection and dynamic creation and deletion of file replicas.

5.2 Simulation Input:

Our program gets input from four configuration files.

- Parameter configuration file: The basic simulation parameters are set in the parameter configuration file such as total number of jobs to be run, delays between each job submission, maximum queue size, the choice of replication strategies, access patterns for the job, etc.
- Grid configuration file: contains the network topology, i.e., the links between grid sites, the available network bandwidth between sites, and number of CEs and SEs, as well as their sizes.
- Job configuration file: describes information about simulated jobs, the files needed by each job, the probability each job runs, etc.
- Bandwidth configuration file: specifies the background network traffic.

As mentioned above, jobs requires to access files during execution. The order in which those files are requested is determined by the access pattern. Four important access patterns are as follow: Sequential (files are selected in the order stated in the job configuration file), random (files are accessed using a random distribution), random walk unitary (files are selected in one direction away from the previous file request and the direction will be random) and random walk Gaussian (files are requested in a Gaussian distribution). These file access patterns are shown in Fig. 4.

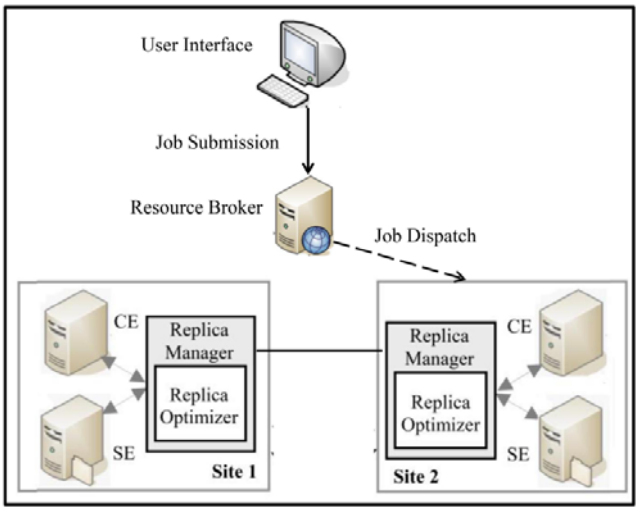

Fig. 3: OptorSim architecture.

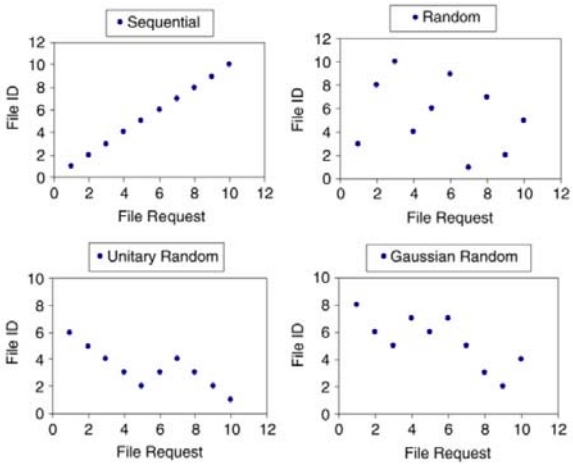

Fig. 4: Various file access patterns.

5.3 Configuration:

With OptorSim, it is possible to simulate any Grid topology and replication strategy. So OptorSim code has been modified to implement the hierarchical structure, since it uses a flat network structure. The Grid topology of simulated platform is given in Fig. 5. It is assumed the network has four regions and each one has three sites. Node 8 has the most capacity to store all the master files at the beginning of the simulation. The storage capacity of all other sites is 50GB. The connection bandwidth is 100 Mbps. We ran the simulation with 1500 jobs. The number of file accessed per job on average is 16 and job delay is 2500 ms. Each data file to be accessed is 2 Gbyte. To simplify the requirements, we assumed that the data is read-only.

5.4 Simulation Results:

We evaluated the performance of MLALW and the seven strategies in four types of access patterns: Sequential, Random Access, Random Walk Unitary Access and Random Walk Gaussian Access. Six replication algorithms have been used for evaluation, namely:

- 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.
- In Least Frequently Used (LFU)strategy always replication takes place 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.
- Bandwidth Hierarchy based Replication (BHR) 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.

- The Modified BHR algorithm (MBHR) replicates the files within the region in a site where file has the highest access.
- 3-Level Hierarchical Algorithm (3LHA) considers a hierarchical network structure that has three levels. Bandwidth is an important factor for replica selection and deletion.
- Dynamic Hierarchical Replication (DHR) algorithm that places replicas in appropriate sites i.e. best site that has the highest number of access for that particular replica. It also minimizes access latency by selecting the best replica by considering the replica requests that waiting in the storage and data transfer time.

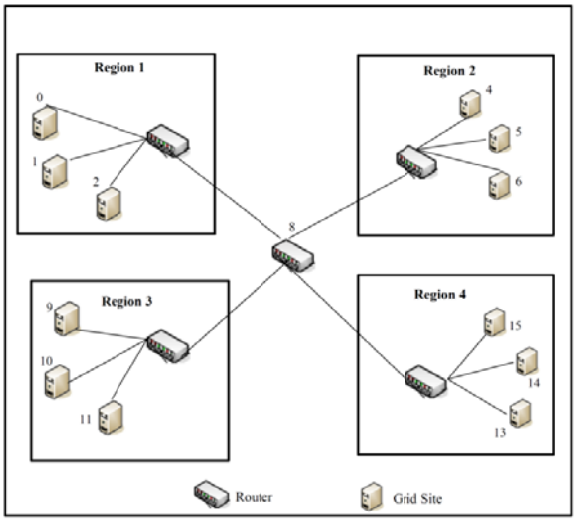

Fig. 5: Grid topology in the simulation.

The comparison result is shown in Fig. 6. The experiment results show that MLALW has the lowest value of mean job execution time in all the experiments and all of file access patterns. Obviously, the No Replication strategy has the worst performance as all the files requested by jobs have to be transferred from CERN. In this simulation LRU and LFU have almost the same execution time. BHR algorithm improves data access time by avoiding network congestions. The 3LHA performs better than BHR because it considers the differences between intra-LAN and inter-LAN communication. MLALW has the best performance since it will not delete those files that have a high transferring time. One of the important parameters that reduce the Grid site's job execution time is having their needed files locally stored on their storage element. It estimates the next number of access for the file. So, considering number of file accesses in future and last access time, in replacement proposed algorithm, made MLALW better than the others. It replicates files wisely and does not delete valuable files which results in preserving the valuable replicas. As in Random access patterns comprising Random, Unitary random walk and Gaussian random walk, 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, MLALW strategy and also all the other strategies have more improvement for random file access patterns.

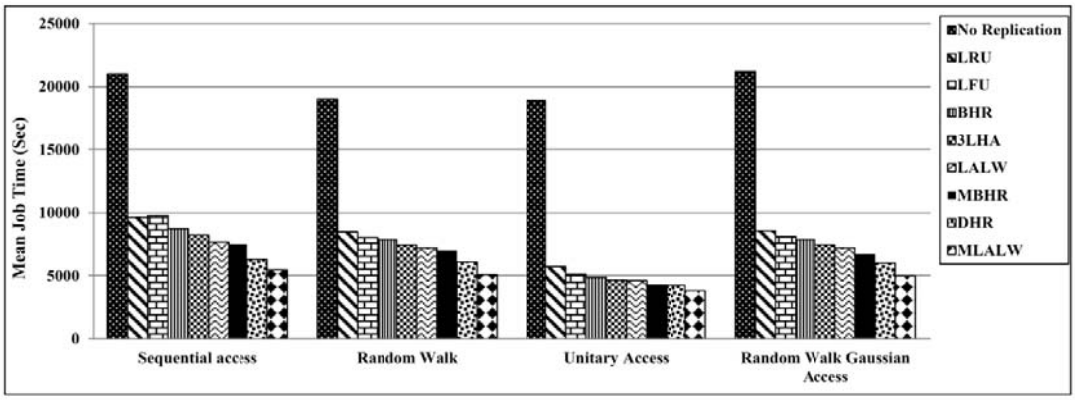

Fig. 6: Mean job execution time for various replication algorithms.

Figure 7 displays the mean job time based on changing number of jobs for seven algorithms. It is clear that as the job number increases, MLALW is able to process the jobs in the lowest mean time in comparison with other methods. It is similar to a real Grid environment where a lot of jobs should be executed.

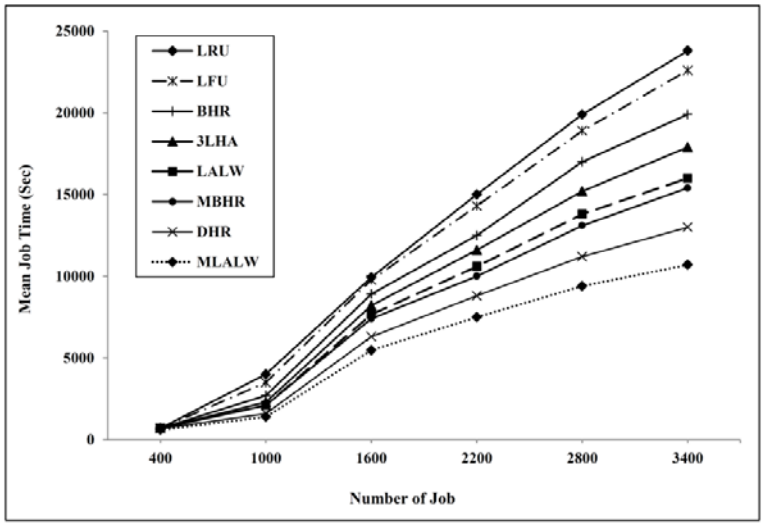

Fig. 7: Mean job time based on varying number of jobs with Sequential access pattern.

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 used to estimate the efficiency the network resource usage. Effective Network Usage (E_{enu})is given from (Bell*et al*., 2003):

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

Where N_{rfa} is the number of access times that CE reads a file from a remote site, N_{fr} 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.

Figure 8 shows the comparison of the Effective Network Usage of the nine replication strategies for the sequential access pattern. The ENU of MLALW is lower about 45% 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 MLALW is optimized to minimize the bandwidth consumption and thus decrease the network traffic. The No Replication strategy operates the worst and consumes the maximum network bandwidth available in the network.

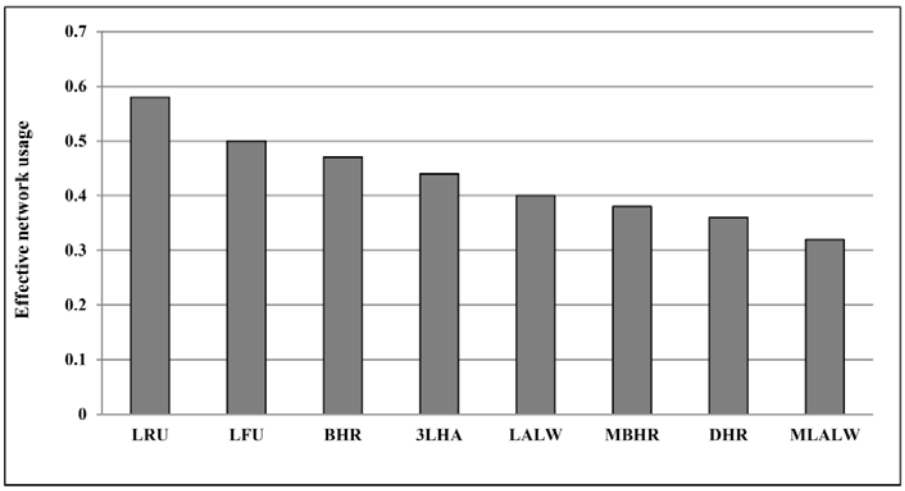

Fig. 8: Effective network usage with Sequential access pattern.

7. Conclusion and Future Work:

Data Replication is an effective technique to reduce file transfer time and bandwidth consumption in Data Grids. A dynamic data replication strategy, called Modified Latest Access Largest Weight (MLALW) for hierarchical structure is proposed. MLALW considers the last time the replica was requested, number of access, and file size of replica in replacement step. 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. We also estimate the next number of access for the file using the concept of exponential decay/gorwth. MLALW stores each replica in an appropriate site i.e. appropriate site in the region that has the highest number of access in future for that particular replica. 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 MLALW to 8 of existing algorithms, No replication, LRU, LFU, BHR, LALW, MBHR, 3LHA and DHR for different file access patterns. The evaluation shows that MLALW outperforms the other algorithms and improves Mean Job Time and Effective Network under all of the access patterns, especially under the different random file access patterns. For future works, MLALW can be combined with a proper scheduling to improve performance. We aim to predict the future needs of Grid sites by using suitable techniques such as data mining. Employing replica consistency management strategies is also our future work plans.Finally, we will focus in integrating users' preferences in the replica selection step.

REFERENCES

Amjad, T., M. Sher and A. Daud, 2012. A Survey of Dynamic Replication Strategies for Improving Data Availability in Data Grids, Future Generation Computer Systems, 28: 337-349.

Andronikou, V., K. Mamouras, K. Tserpes, D. Kyriazis, T. Varvarigou, 2012. Dynamic QoS-aware data replication in grid environments based on data''importance''. Future Generation Computer Systems, 28: 544-553.

Bell, W.H., D.G. Cameron, L. Capozza, A.P. Millar, K. Stockinger and F. Zini, 2003. Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies. International Journal of High Performance Computing Applications, 17(4): 1-20.

Bell, W.H., D.G. Cameron, R. Carvajal-Schiaffino, A.P. Millar, K. Stockinger and F. Zini, 2003. Evaluating Scheduling and Replica Optimization Strategies in Data Grid. IEEE.

Buyya, R. and M. Murshed, 2002. GridSim: a Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. The Journal of Concurrency and Computation: Practice and Experience, 14: 1175-1200.

Cameron, D.G., A.P. Millar, C. Nicholson, R. Carvajal-Schiaffino, F. Zini and K. Stockinger, 2004. Optorsim: A Simulation Tool for Scheduling and Replica Optimization in Data Grids. In: International Conference for Computing in High Energy and Nuclear Physics (CHEP 2004).

Casanova, H., 2001. Simgrid: A Toolkit for the Simulation of Application Scheduling. Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'01), 18-21.

Chakrabarti, A. and S. Sengupta, 2008. Scalable and Distributed Mechanisms for Integrated Scheduling and Replication in Data Grids. ICDCN, 227-238.

Chang, R.S. and H.P. Chang, 2008. A Dynamic Data Replication Strategy Using Access Weights in Data Grids. Journal of Supercomputing, 45(3): 277-295.

Chang, R.S. and M.S. Hu, 2010. A Resource Discovery Tree Using Bitmap for Grids. Future Generation Computer Systems, 26: 29-37.

Chen, D.W., S.T. Zhou, X.Y. Ren and Q. Kong, 2010. Method for Replica Creation in Data Grids based on Complex Networks, The Journal of China Universities of Posts and Telecommunications, 17(4): 110-115.

Chervenak, A., E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, 2002. A Framework for Constructing Scalable Replica Location Services. In: Supercomputing, ACM/IEEE Conference, pp: 58.

Cibej, U., B. Slivnik and B. Robic, 2005. The Complexity of Static Data Replication in Data Grids. Parallel Computing, 31(8): 900-912.

Ebadi, S. and L.M. Khanli, 2011. A new distributed and hierarchical mechanism for service discovery in a Grid environment. Future Generation Computer Systems, 27: 836-842.

Foster, I. and C. Kesselman, 2004. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann.

Foster, I. and K. Ranganathan, 2001. 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.

Foster, I., C. Kesselman, S. Tuecke, 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 200-222.

Horri, A., R. Sepahvand and Gh. Dastghaibyfard, 2008. A Hierarchical Scheduling and Replication Strategy. International Journal of Computer Science and Network Security, 8.

Jaradat, A., A.H. Muhamad Amin and M., Nordin Zakaria, 2011. Balanced QoS Replica Selection Strategy to Enhance Data Grid, 2nd International Conference on Networking and Information Technology, 17.

Lamehamedi, H., Z. Shentu, B. Szymanski and E. Deelman, 2003. Simulation of Dynamic Data Replication Strategies in Data Grids.

Lee, B.D. and J.B. Weissman, 2001. Dynamic Replica Management in the Service Grid. In: High Performance Distributed Computing, Proceedings, 10th IEEE International Symposium on, 433-434.

Mansouri, N. and Gh. Dastghaibyfard, 2012. A Dynamic Replica Management Strategy in Data Grid. Journal of Network and Computer Applications, 35(4): 1297-1303.

Mansouri, N., Gh. Dastghaibyfard and A. Horri, 2011. A Novel Job Scheduling Algorithm for Improving Data Grid's Performance. International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 142-147.

Nemati, Y., F. Samsami and M. Nikhkhah, 2012. A Novel Data Replication Policy in Data Grid. Australian Journal of Basic and Applied Sciences, 6(7): 339-344.

Nukarapu, D.T., B. Tang, L. Wang and S. Lu, 2011. Data Replication in Data Intensive Scientific Applications with Performance Guarantee. IEEE Transactions on Parallel and Distributed Systems, 22(8).

Park, S.M., J.H. Kim, Y.B. Ko and W.S. Yoon, 2003. Dynamic Grid Replication Strategy based on Internet Hierarchy. In: International Workshop on Grid and Cooperative Computing, in: Lecture Note in Computer Science, 1001: 1324-1331.

Perez, J.M., F. Garcia-Carballeira, J. Carretero, A. Calderon, and J. Fernandez, 2010. Branch Replication Scheme: a New Model for Data Replication in Large Scale Data Grids. Future Generation Computer Systems, 26(1): 12-20.

Rabinovich, M., I. Rabinovich and R. Rajaraman, 1998. Dynamic Replication on the Internet, Technical Report, HA6177000–980305-01-TM, AT&T Labs.

Ranganathan, K. and I. Foster, 2001. Identifying Dynamic Replication Strategies for a High Performance Data Grid. In: Proceedings of the Second International Workshop on Grid Computing, 75-86.

Sashi, K. and A.S. Thanamani, 2011. Dynamic Replication in a Data Grid Using a Modified BHR Region based Algorithm. Future Generation Computer Systems, 27(2): 202-210.

Sepahvand, R., A. Horri and Gh. Dastghaibyfard, 2011. Replication and Scheduling Methods Based on Prediction in Data Grid, Australian Journal of Basic and Applied Sciences, 5(11): 1485-1496.

Shen, H., 2010. An Efficient and Adaptive Decentralized File Replication Algorithm in P2P File Sharing Systems. IEEE Transactions on Parallel and Distributed Systems, 21(6): 827-840.

Song, H.J., J. Liu, D. Jakobsen, X. Zhang, K. Taura and A. Chien, 2000. The MicroGrid: a Scientific Tool for Modeling Computational Grids. Scientifics Programming, 8(3): 127-141.

Takefusa, A., S. Matsuoka, H. Nakada, K. Aida, U. Nagashima, 1999. Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms. Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing.

Tatebe, O., Y. Morita, S. Matsuoka, N. Soda, S. Sekiguchi, 2002. Grid Datafarm Architecture for Petascale Data Intensive Computing. In: CCGrid, pp: 102.

Vrable, M., S. Savage and G.M. Voelker, 2009. Cumulus: File System Backup to the Cloud. ACM Transactions on Storage, 5(4).

Wolfson, O., S. Jajodia and Y. Huang, 1997. An Adaptive Data Replication Algorithm. ACM Transactions on Database Systems, 22 (2): 255-314.

Wu, J., X. Xu, P. Zhang and C. Liu, 2011. A Novel Multi-Agent Reinforcement Learning Approach for Job Scheduling in Grid Computing. Future Generation Computer Systems, 27: 430-439.

Yuan, Y., Y. Wu, G. Yang and F. Yu, 2007. Dynamic Data Replication based on Local Optimization Principle in Data Grid.