Improving Data Grids Performance by using Modified Dynamic Hierarchical Replication Strategy

N. Mansouri* and Gh. Dastghaibyfard** (C.A.)

Abstract: A Data Grid connects a collection of geographically distributed computational and storage resources that enables users to share data and other resources. Data replication, a technique much discussed by Data Grid researchers in recent years creates multiple copies of file and places them in various locations to shorten file access times. In this paper, a dynamic data replication strategy, called Modified Dynamic Hierarchical Replication (MDHR) is proposed. This strategy is an enhanced version of Dynamic Hierarchical Replication (DHR). However, replication should be used wisely because the storage capacity of each Grid site is limited. Thus, it is important to design an effective strategy for the replication replacement task. MDHR replaces replicas based on the last time the replica was requested, number of access, and size of replica. It selects the best replica location from 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 that waiting in the storage queue, the distance between nodes and CPU process capability. Simulation results utilizing the OptorSim show MDHR achieves better performance overall than other strategies in terms of job execution time, effective network usage and storage usage.

Keywords: Access Pattern, Data Grid, Data Replication, Simulation.

1 Introduction

Millions of files will be generated from scientific applications and researchers around the world need access to the large data [1-6]. It is difficult and inefficient to store such huge amounts of data using a centralized storage. Grid technology is the best solution to this kind of problem. The main objective of the Grid Project is to provide sharing of computing and storage resources by users located in different part of the world. 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 Grids deals with the applications that require studying and analyzing massive data sets [7-10]. Replication technique is one of the major factors affecting the performance of Data Grids by replicating data in geographically distributed data stores. 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.

Meanwhile, even though the memory and storage size of new computers are ever increasing, they are still not keeping up with the request of storing large number of data. The major challenge is a decision problem i.e. how many replicas should be created and where replicas should be stored. Hence methods needed to create replicas that increase availability without using unnecessary storage and bandwidth. In this work a novel dynamic data replication strategy, called Modified Dynamic Hierarchical Replication (MDHR) algorithm is proposed. This strategy is an enhanced version of Dynamic Hierarchical Replication strategy. 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. But MDHR algorithm

Iranian Journal of Electrical & Electronic Engineering, 2014.
Paper first received 20 June 2012 and in revised form 6 Oct. 2013.
* The Author is with the Department of Computer Science, Shahid Bahonar University of Kerman, 22 Bahman Bolvar, Kerman, Iran.
** The Author is with the Department of Computer Science and Engineering, College of Electrical & Computer Engineering, Shiraz University, Molla Sadra Avenue, Shiraz, Iran.
E-mails: najme.mansouri@gmail.com, dstghaib@gmail.com and dstghaib@shirazu.ac.ir.

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. The number of requests and the last time the replica was requested define an indication of the probability of requesting the replica again. 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 the many replicas based on response time that can be determined by considering the data transfer time, the storage access latency, the replica requests that waiting in the storage queue, the distance between nodes and CPU process capability. The proposed algorithm is implemented by using a Data Grid simulator, OptorSim developed by European Data Grid project. The simulation results show that our proposed algorithm has better performance in comparison with other algorithms in terms of job execution time, effective network usage and storage resource usage.

The rest of the paper is organized as follows: Section 2 presents some examples motivating Data Grid use. In section 3 the classification of data replication strategies is briefly explained. Section 4 explains some advantages of data replication strategies. Section 5 gives an overview of pervious work on data replication in Data Grid. Section 6 presents the novel data replication strategy. We show and analyze the simulation results in section 7. Finally, section 8 concludes the paper and suggests some directions for future work.

2 Motivation

Nowadays large volume data are generated in many fields like scientific experiments and engineering applications. The distributed analysis of these amount data and their dissemination among researchers located over a wide geographical area needs important techniques such as Data Grid.

For example, the high-energy physics experiment requires a lot of analyses on huge amounts of data sets. CERN is the world's largest particle physics center [11]. The LHC (Large Hadron Collider) at CERN will start to work in 2007. The volume of data sets produced by the LHC is about 10 PB a year. CERN has used the Grid technique to solve this challenging huge data storage and computing problem.

Climate models are important to find climate changes and they are improved after today's models are completely analyzed [12]. Climate models require large computing capability and there are only a few sites world-wide that are appropriate for executing these models. Climate scientists are distributed all over the world and they test the model data. Now, model analysis is done by transferring the data of interest from the computer modeling site to the climate scientist's organization for different post-simulation analysis tasks. The effective data distribution method is necessary to the climate science when the data volume is large.

The Biomedical application field [13] wants to use the Grid to help the archiving of biological objects and medical images in distributed databases, communications between hospitals and medical organization and to present distributed access to the data. Hence, data movement is essential. The Earth observation application area investigates the nature of the planet's surface and atmosphere. One application of Grid technique is for the analyzing and validation of ozone profiles. The processed data sets is scattered to other places worldwide. Use cases for Earth observation science applications are explained in more detail in Ref. [14].

The Human Genome Project [15] produces detailed genetic and physical maps of the human genome. The project requires advanced means of making novel scientific information widely available to researchers so that the results may be used for the public good. Data Grid project is funded by the European Union [16]. A Data Grid consists of a group of geographically distributed computational and storage resources placed in various locations, and enables users to share data and other resources [17-20].

3 Classification of Data Replication Strategies

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 [21, 22]. On the other hand, dynamic strategies create and delete replicas according to the changes in Grid environments, i.e. users' file access pattern [23-27]. As Data Grids are dynamic environments and the requirements of users are variable during the time, dynamic replication is more appropriate for these systems [28]. 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.

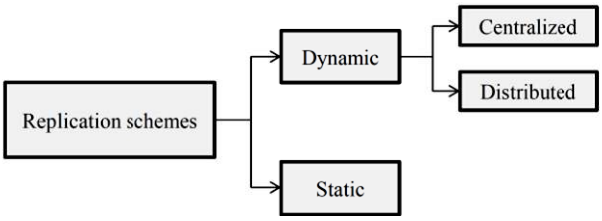

Fig. 1 Types of replication schemes.

4 Advantages of Data Replication Strategies

Availability: All the replication strategies want to improve availability. When a fail happen in any site, a system can resort to replicated data that is stored at other site. In this manner the availability increases.

Reliability: The more the number of replicas more is the probability that user's request will be done completely, and hence a system is more reliable.

Scalability: This is a key parameter which must be provided by a replication strategy. Scalability is more dependent on architecture model selected for the Data Grid than replication strategy.

Adaptability: The nature of the Grid is very changeable and users can enter and quit a Grid at any time. The data replication strategy must be adaptive to the information of the Grid environment in order to provide better results.

Performance: One way to speed up access in data Grid systems is to store replicas at multiple locations, so a user can access the data from the nearest site.

5 Related Works

Foster and Ranganathan [28], 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.
- 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.

Rahman et al. [29] 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 and decreased the prediction error as reported by using Neural Network techniques. Accordingly, one site can request the replica from a site which has minimum transfer time.

In [30] the authors presented a data replication strategy that has a provable theoretical performance guarantee and can be implemented in a distributed and practical manner. They also proposed a distributed caching strategy, which can be easily adopted in a distributed system such as Data Grids. The key point of their distributed strategy is that when there are several replicas, each Grid site keeps track of its closest replica site. This can dramatically enhance Data Grid performance because transferring large-sized files is time and bandwidth consuming [31]. The results of simulation demonstrated that the distributed replication algorithm significantly outperforms a popular existing replication strategy under various network parameters.

Tang et al. [32] presented Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU) strategies to improve the average data access response time for a multi-tier data grid. The main idea of the two strategies is to store a replica to nodes close to its requesting clients when the file's access rate is higher than a pre-defined threshold. SBU uses the file access history for each node, but ABU aggregates the file access history for a system. With ABU, a node transmits aggregated historical access records to its top tiers, and the top tiers do the same until these records reach the root. The results show that ABU improves job response time and bandwidth consumption better than those of SBU because its aggregation capability.

Andronikou et al. [33] proposed a set of interoperable new data replication strategies that take into account the infrastructural constraints as well as the 'importance' of the data. The presented system is scalable and the strategies can be easily implemented on a Grid environment to provide fast execution. The proposed QoS-aware dynamic replication strategy determines the number of replicas required based on data request, content importance and requested QoS. It also places of the new replicas within the Grid environment according to the network bandwidth and the overhead that the replication technique presents. It can handle the dynamicity of the Grid system by increasing or decreasing the set of data replicas based on the number and the geography of the data demands.

Lee et al. [34] presented an adaptive data replication strategy for a star-topology Data Grid, called the Popular File Replicate First algorithm (PFRF). It periodically computes file access popularity to track the changes of users' access behaviors, and then replicates popular files to suitable clusters/sites to adapt to the variation. They considered several types of file access behaviors, including Zipf-like, geometric, and uniform distributions, to evaluate PFRF. The simulation results

demonstrate that PFRF can reduce average job turnaround time and bandwidth consumption.

Saadat et al. [35] presented a new dynamic data replication strategy which is called Pre-fetching based Dynamic Data Replication Algorithm in Data Grids (PDDRA). PDDRA predicts future requires of Grid sites and pre-replicates them before needs are requested. This prediction is done based on the past file access history of the Grid sites. So when a Grid site requests a set of files, it will get them locally. The simulation results show that this strategy improves in terms of job execution time, effective network usage, number of replications, hit ratio and percentage of storage filled.

Taheri et al. [36] proposed a new Bee Colony based optimization strategy, called Job Data Scheduling using Bee Colony (JDS-BC). JDS-BC has two collaborating operations to efficiently schedule jobs onto computational elements and replicate data sets on storage elements in a system so that the two independent, and in many cases conflicting, objectives (i.e., makespan and transfer time of all datafiles) of such heterogeneous systems are concurrently decreased. Three tailor-made test Grids varying from small to large are applied to evaluate the performance of JDS-BC and compare it with other strategies. Results showed that JDS-BC's superiority under different operating scenarios. JDS-BC also presented a balanced decision making behavior, where it occasionally relaxes one of its objectives (e.g., transfer time) to obtain more from optimizing the other one (e.g., makespan).

Mansouri and Dastghaibyfard [37] 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.

Park et al. [38] 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. Modified BHR [39] is an extension of BHR [38] strategy which replicates a file that has been accessed most and it may also be used in near future.

A replication algorithm for a 3-level hierarchy structure and a scheduling algorithm are proposed in [40]. 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 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.

6 Proposed Dynamic Replication Algorithm

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

6.1 Network Structure

The Grid topology of simulated platform is given in Fig. 2. The hierarchical network structure has three levels. First level are Regions that are connected through internet i.e. have low bandwidth. Second level comprises LAN's (local area network) within each region that has moderately higher bandwidth comparing to the first level. Finally the third level are the nodes (sites) within each the LAN's, that are connected to each other with a high bandwidth.

6.2 Dynamic Hierarchical Replication Algorithm

In previous work, we proposed DHR algorithm [37]. DHR strategy first checks replica feasibility. If the requested file size is greater than SE (Storage Element) size, file will be accessed remotely. It among the candidate replicas selects the one that has the highest bandwidth to the requester node and has least number of requests. It also places replicas in best sites i.e. best site that has the highest number of access for that particular replica.

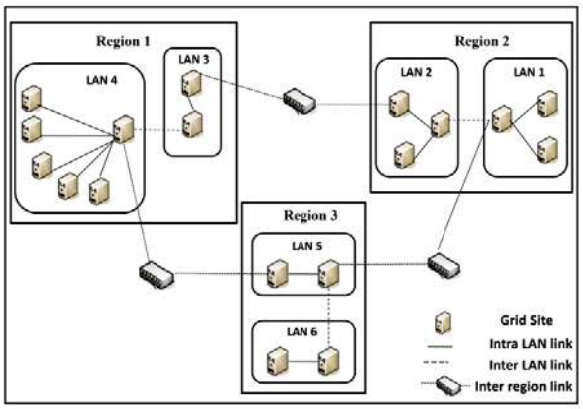

Fig. 2 Grid topology in the simulation.

If the available space in best SE is greater or equal to requested file size, it replicates the file. If not, some files should be deleted. It deletes those files with minimum time for transferring (i.e. only files that are exist in local LAN and local region).

6.3 Modified Dynamic Hierarchical Replication Algorithm

MDHR algorithm has been described in three parts: *1. Replica Selection*: When different sites have replicas of file, there is a significant benefit realized by selecting the best replica. Replica selection decides which replica location is the best for the Grid users. Four important factors are used to choose a best replica:

- *Storage access latency*

The storage media speed and size of requests queue have a key role in the average response time. T_1 can be calculated by the following equation:

$$T_1 = \frac{FileSize(MB)}{StorageSpeed(MB/Sec)} \quad (1)$$

StorageSpeed shows the storage media speed and *FileSize* represents size of file.

- *Waiting time in the storage queue*

Each storage media has some requests at the same time and the storage can perform only one request at a time. So, one has to wait for all the previous requests in the storage queue. T_2 can be defined by the following equation:

$$T_2 = \sum_{i=0}^{n} T_i \quad (2)$$

where n is number of requests waiting in the queue.

- *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.

- *CPU process capability*

CPU process capability is defined by equation

$$C = F \times U_{CPU} \quad (3)$$

where F is the CPU frequency, U_{CPU} is CPU usage.

$$T = T_1 + T_2 \quad (4)$$

$$F(x, y) = w_1 \times T + w_2 \times D(x, y) + w_3 \times C \quad (5)$$

This function can be tailored, because it is defined as a weighted combination of the three former metrics.
The proper weights (w_1, w_2, w_3) have been obtained empirically. We examined different weights in equations and selected the best values.The interrelationships between the above parameters are not very simple. In future work, we plan to use Taguchi technique for determining weights.

The goal of replica manager (RM) in each site is to reduce file transfer time for each job. RM for locating unavailable requested files of each job controls the existence of the file in local LAN. If the file duplicated in the same LAN, then it will create a list of candidate replicas and selects a replica with the minimum *F*. If the file doesn't exist in the same LAN, then MDHR searches within the local region. If the file duplicated in the same region, then it will create a list of candidate replicas and selects a replica with the minimum *F*. Otherwise it generates a list of replicas that are available in other regions, and from this list it selects the replica that has minimum *F*.

2. Replica placement: When a requested replica is not available in the local storage, replication should take place. According to the temporal and geographical locality the replica is placed in the best site (BSE). To select the BSE, DHR creates a sorted list (by number of replica access) of all SE's that requested that particular file in the region. Now the replica will be placed in the first storage element (SE) of the above sorted list, i.e. BSE. If more than one SE is candidate one can be selected randomly. Therefore, replica is not placed in all the requested sites. Hence, storage cost as well as mean job execution time can decrease.

Assume MDHR wants to find the best site for storing replica *R*. Now *list1* shows the sorted list created for replica *R*, then MDHR selects site *S7* from *LAN3* because *S7* has the highest number of requests for *R* which is shown in Fig. 3.

3. Replica Management: If enough storage space exists in the BSE, the selected file is replicated. Otherwise if the file is available in the local LAN, then the file is accessed remotely. Now, if enough space for replication does not exist and requested file is not available in the same LAN, one or more files should be deleted using the following rules:

- Generate a LRU sorted list of replicas that are both available in the site as well as the local LAN. Now start deleting files from the above list till space is available for replica.

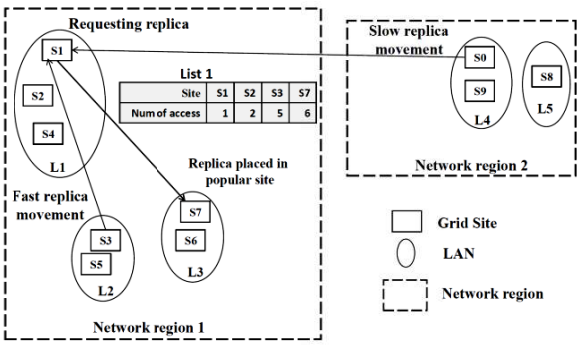

Fig. 3 Replica placement strategy.

- If space is still insufficient, then repeat previous step for each LAN in current region randomly. In other words, generate a LRU sorted list of replicas that are both available in the site as well as the local region.
- If space is still insufficient, a group of replicas (that contains one or more replicas) need to be deleted. In this step using LRU 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. In this step three valuable factors (the last time the replica was requested, number of access, and file size of replica) will be considered instead of LRU. The number of access and the last time the replica was requested define an indication of the probability of requesting the replica again. Obviously, it is more important to replace files with large size, because it can reduce the number of replica replacement. The replica value (RV) is calculated by equation

$$RV = w_1 \times S + w_2 \times NA + w_3 \times \frac{1}{CT - LA} \quad (6)$$

where the value of w_1, w_2 and w_3 can be assigned by the users, S is file size, NA is the number of access to the replica, CT is the current time and LA is the last request time of replica. The replica with least value of RV would be replaced with the new replica. Figure 4 describes DHRS strategy.

7 Experiments

In this section, simulation tool, simulation input, configuration, experiment results and discussion are described respectively.

7.1 Simulation Tool

OptorSim was developed to simulate the structure of a real Data Grid for evaluating various replication strategies, as shown in Fig. 5 [41, 42].The OptorSim has several parts:

- Computing Element (CE): represents computational resource in Data Grid.
- Storage Element (SE): represents data resource in Data Grid.
- Resource Broker (RB): gets the user's job and allocates each job to proper site based on the selected scheduling policy.
- Replica Manager (RM): at each site controls data transferring and provides a mechanism for accessing the Replica Catalog.
- Replica Optimizer (RO): within RM implements the replication algorithm.

7.2 Simulation Input

The OptorSim tool works based on several 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.

```
// locate unavailable requested files (URF)
1. foreach (URF fi in the local site) {
If (fi available in local LAN){
Create list L of fj's that are available in local LAN.
Select fi from L that has minimum value of F.
Continue; }// end if
If (fi available in local region){
Create list L of fj's that are available in local region.
Select fi from L that has minimum value of F.
Continue; }// end if
Create list L of fi's that are available in other regions.
Select a fi from L that has minimum value of F.
} //end foreach step 1

// Now all requested files are available in the local site
2. Execute the job
// Now replicate each unavailable requested files (URF)
// in the best SE (BSE)
3. foreach (Ri  ϵ URF)
    Select best SE (BSE) for storing Ri.
// in this step local LAN means the LAN that holds BSE
// also local region means the region that holds BSE
SRi ← size of Ri;
    SBSE ← storage size of best SE;
// Do not replicate if size of Ri ≥ storage size of best SE;
    If (SRi ≥ SBSE) ; continue; // not enough space
    If (enough space exist for Ri in BSE) {Store Ri;   continue;}
// Do not replicate if Ri is available in the local LAN
    If (Ri 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 Ri)
        delete first file from list L in BSE;
    If (enough space exist for Ri in BSE) {Store Ri;   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 Ri)
        delete first file from list L in BSE;
    If (enough space exist for Ri in BSE) {Store Ri;   continue;}

// Now delete from remaining files in BSE
    Create list L from remaining files in BSE.
    Sort list L using RV in ascending order.
    While (L is not empty && not enough space for Ri)
    Delete first file from list L in BSE;
    Store Ri;  }//end foreach step 3
```

Fig. 4 MDHR algorithm.

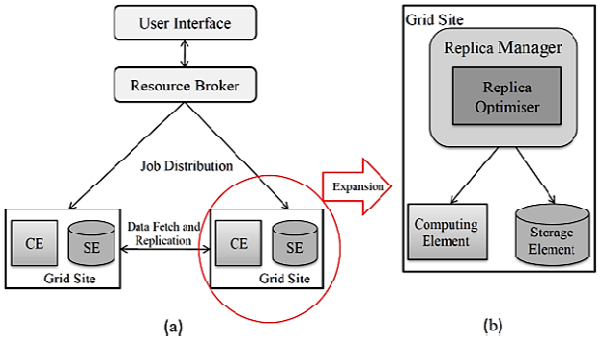

Fig. 5 OptorSim architecture.

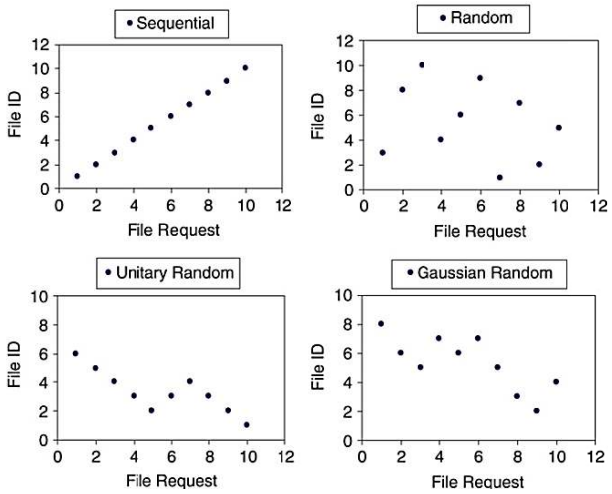

Fig. 6 Various file access patterns.

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. 6.

7.3 Configuration

There are three regions in our configuration and each region has an average two LANs. Initially all master files are distributed to CERN. A master file consists of the original file and cannot be deleted. The topology of our simulated platform includes 10 CEs and 11 SEs. The storage capacity of the master site is 300 GB and the storage capacity of all other sites is 40 GB. Bandwidth in each level is given in Table 1. There are 6 job types, and each job type on average requires 16 files (each is 2 GB) for execution.

Table 1 Bandwidth configuration.

Parameter	Value (Mbps)
Inter LAN bandwidth (level 3)	1000
Intra LAN bandwidth (level 2)	100
Intra Region bandwidth (level 1)	10

Table 2 General job configuration.

Parameter	Value
Number of jobs	2000
Number of jobs types	6
Number of file access per jobs	16
Size of single file (GB)	2
Job delay (ms)	2500

Table 2 specifies the simulation parameters and their values used in our study. To simplify the requirements, we assumed that the data is read-only.

7.4 Simulation Results

We evaluated the performance of MDHR and the six 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 Frequently Used (LFU) 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 Recently Used (LRU) 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 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 strategy (DHR) 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.

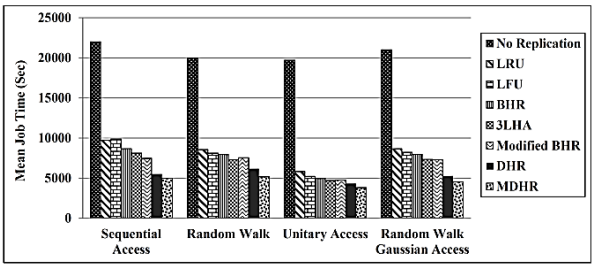

Fig. 7 Mean job execution time for various replication algorithms.

The comparison result is shown in Fig. 7. The experiment results show that MDHR 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. DHR mean job execution time is faster than Modified BHR. MDHR strategy has the best performance since it will not delete those file 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 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, MDHR strategy and also all the other strategies have more improvement for random file access patterns.

Figure 8 displays the mean job time based on changing number of jobs for seven algorithms. It is clear that as the job number increases, MDHR 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.

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. Effective Network Usage (ENU) is used to estimate the efficiency the network resource usage. ENU (E_{enu}) is given from [43]:

$$E_{enu} = \frac{N_{rfa} + N_{fa}}{N_{lfa}} \quad (8)$$

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. Figure 9 shows the comparison of the Effective Network Usage of the seven replication strategies for the sequential access pattern. The ENU of MDHR is lower about 58% 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 MDHR is optimized to minimize the bandwidth consumption and thus decrease the network traffic.

Figure 10 shows the mean job time of replication algorithms for varying inter region bandwidth. When we set narrow bandwidth on the inter-region link, our strategy outperforms other strategy considerably.

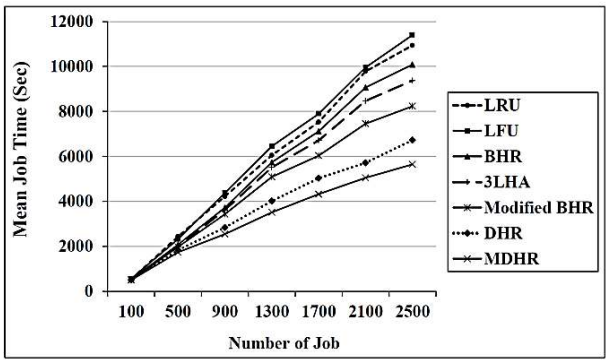

Fig. 8 Mean job time based on varying number of jobs.

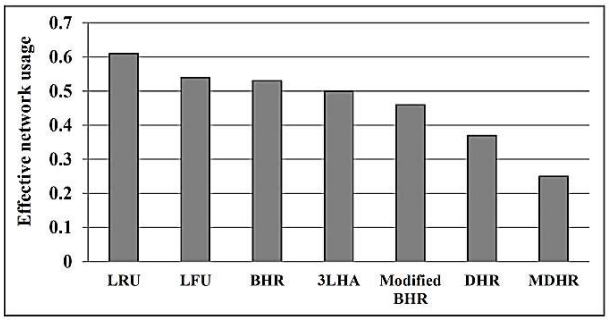

Fig. 9 Effective network usage with sequential access pattern generator.

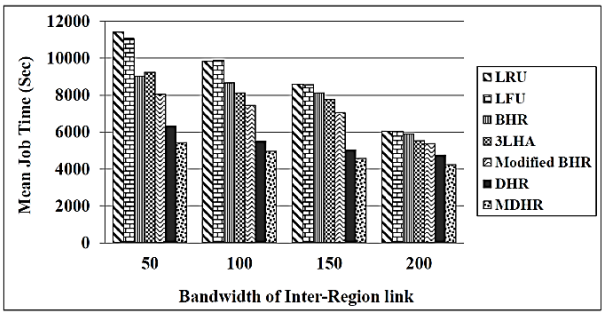

Fig. 10 Mean job time with varying bandwidth.

8 Conclusion

Data replication is a key method used to improve the performance of data access in distributed systems. In this paper, we propose a novel data replication algorithm for a three level hierarchical structure with limited storage space to improve system performance. MDHR replaces replicas based on the last time the replica was requested, number of access, and file size of replica. 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. 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 the many replicas based on response time that can be determined by considering the data transfer time, the storage access latency, the replica requests that waiting in the storage queue, the distance between nodes and CPU process capability. To evaluate the efficiency of our policy, we used the Grid simulator OptorSim that is configured to represent a real world Data Grid test bed. We compared MDHR to seven of existing algorithms, No replication, LRU, LFU, BHR, Modified BHR, 3LHA and MDHR for different file access patterns. The evaluation shows that MDHR outperforms the other algorithms and improves Mean Job Time and Effective Network Usage under all of the access patterns, especially under the different random file access patterns. Also, we concluded that MDHR can be effectively utilized when hierarchy of bandwidth appears apparently.

In the future, we plan to test our simulation results on real Data Grid. We also try to investigate dynamic replica maintenance issues such as replica consistency. In the longer-term, we would like to consider the set of QoS factors taken into account for dynamic replication, including both service provider and client-related requirements.

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, pp. 1304-1316, 2010.

[2] O. Sonmez, H. Mohamed and D. Epema, "On the benefit of processor co-allocation in multi cluster Grid systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, pp. 778-789, 2010.

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

[4] M. Sedghi and M. Aliakbar-Golkar, "Distribution Network Expansion Using Hybrid SA/TS Algorithm", *Iranian Journal of Electrical & Electronic Engineering*, Vol. 5, No. 2, pp. 122-130, 2009.

[5] M. Sharma and K. P. Vittal, "A Heuristic Approach to distributed Generation Source Allocation for Electrical Power Distribution Systems", *Iranian Journal of Electrical & Electronic Engineering*, Vol. 6, No. 4, pp. 224-231, 2010.

[6] M. R. Aghamohammadi, "Static Security Constrained Generation scheduling Using Sensitivity Characteristics of Neural Network", *Iranian Journal of Electrical & Electronic Engineering*, Vol. 4, No. 3, pp. 104-114, 2008.

[7] GriPhyN: The Grid physics network project, 12 July 2010. http://www.griphyn.org, [Accessed: January 2014].

[8] R. S. Chang and M. S. Hu, "A resource discovery tree using bitmap for Grids", *Future Generation Computer Systems*, Vol. 26, pp. 29-37, 2010.

[9] 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, pp. 430-439, 2011.

[10] 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, pp. 836-842, 2011.

[11] CERN. http://public.web.cern.ch/public/, [Accessed: January 2014].

[12] N. T. Thuy, T. T. Anh, D. D. Thanh, D. T. Tung, N. T. Kien and T. T. Giang, "Construction of a Data Grid for meteorology in Vietnam", *Proceedings of the 2007 Int. Conf. on Grid Computing & Applications*, pp. 186-191, 2007.

[13] J. Montagnat and H. Duque, "Medical data storage and retrieval on the Data Grid", *Technical Report DataGrid-10-TED-345148-0-1, the European Data Grid Project*, 2002.

[14] Work Package 9, WP9.4 Use Case, *Technical Report DataGrid-09-TED-0101-1-1, the European Data Grid Project*, 2002.

[15] Human Genome Project: http:// www.nhgri.nih.gov/, [Accessed July 2011].

[16] The Data Grid project. http://eu-datagrid.web.cern.ch/eu-datagrid/, [Accessed January 2014].

[17] S. Figueira and T. Trieu, "Data replication and the storage capacity of Data Grids", *Berlin, Heidelberg: Springer-Verlag*, pp. 567-75, 2008.

[18] D. G. Cameron, A. P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger and F. Zini, "Analysis of scheduling and replica optimization strategies for Data Grids using Optorsim", *Journal of Grid Computing*, Vol. 2, No. 1, pp. 57-69, 2004.

[19] H. Zhong, Z. Zhang and X. Zhang, "A Dynamic replica management strategy based on Data

Grid", *in 2010 Ninth Int. Conf. on Grid and Cloud Computing*, pp.18-22, 2010.

[20] H. Lamehamedi, B. Szymanski, Z. Shentu and E. Deelman, "Data replication strategies in Grid environments", *in Proceedings of the fifth int. conf. on Algorithms and Architectures for Parallel Processing*, pp. 378-383, 2002.

[21] O. Tatebe, Y. Morita, S. Matsuoka, N. Soda and S. Sekiguchi, "Grid datafarm architecture for petascale data intensive computing", *Proc. of the 2nd IEEE/ACM Int. Symp. on Cluster Computing and the Grid* (*CCGrid*), pp. 102-110, 2002.

[22] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger and B. Tierney, "A framework for constructing scalable replica location services", *in Proc. of Super Computing, ACM/IEEE Conf.*, pp. 1-17, 2002.

[23] I. Foster and K. Ranganathan, "Design and evaluation of dynamic replication strategies a high performance Data Grid", *in Proceedings of Int. Conf. on Computing in High Energy and Nuclear Physics*, China, 2001.

[24] U. Cibej, B. Slivnik and B. Robic, "The complexity of static data replication in Data Grids", *Parallel Computing*, Vol. 31, No. 8, pp. 900-912, 2005.

[25] Y. Yuan, Y. Wu, G. Yang and F. Yu, "Dynamic data replication based on local optimization principle in Data Grid", *in Proceedings of GCC*, pp. 815-822, 2007.

[26] H. Lamehamedi, Z. Shentu, B. Szymanski and E. Deelman, "Simulation of dynamic data replication strategies in Data Grids*", in Parallel and Distributed Processing Symposium*, 2003.

[27] B. D. Lee and J. B. Weissman, "Dynamic replica management in the service Grid", *10th IEEE Int. Symposium on High Performance Distributed Computing Proc.*, pp. 433-434, 2001.

[28] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high performance Data Grid", *in Proceedings of the Second International Workshop on Grid Computing*, pp. 75-86, 2001.

[29] R. M. Rahman, K. Barker and R. Alhajj, "Replica selection strategies in Data Grid", *Journal of Parallel and Distributed Computing*, Vol. 68, pp. 1561-1574, 2008.

[30] D. T. Nukarapu, B. Tang, L. Wang and S. Lu, "Data replication in data intensive scientific applications with performance guarantee", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 8, pp. 1299-1306, 2011.

[31] A. Chervenak, R. Schuler, M. Ripeanu, M. A. Amer, S. Bharathi, I. Foster and C. Kesselman, "The Globus replica location service: design and experience", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 20, pp. 1260-1272, 2009.

[32] M. Tang, B. S. Lee, C. K. Yao and X. Y. Tang, "Dynamic replication algorithm for the multi-tier Data Grid", *Future Generation Computer Systems*, Vol. 21, No. 5, pp. 775-790, 2005.

[33] V. Andronikou, K. Mamouras, K. Tserpes, D. Kyriazis and T. Varvarigou, "Dynamic QoS-aware data replication in Grid environments based on data "importance", *Future Generation Computer Systems*, Vol. 28, No. 3, pp. 544-553, 2012.

[34] M. C. Lee, F. Y. Leu, and Y. Chen, "PFRF: An adaptive data replication algorithm based on startopology Data Grids", *Future Generation Computer Systems*, Vol. 28, No. 7, pp. 1045-1057, 2012.

[35] N. Saadat and A. M. Rahmani, "PDDRA: A new pre-fetching based dynamic data replication algorithm in Data Grids", *Future Generation Computer Systems*, Vol. 28, No. 4, pp. 666-681, 2012.

[36] J. Taheri, Y. C. Lee, A. Y. Zomaya and H. J. Siegel, "A Bee Colony based optimization approach for simultaneous job scheduling and data replication in Grid environments", *Computers & Operations Research*, Vol. 40, No. 6, pp. 1564-1578, 2013.

[37] N. Mansouri and G. H. Dastghaibyfard, "A dynamic replica management strategy in Data Grid", *Journal of Network and Computer Applications*, Vol. 35, No. 4, pp. 1297-1303, 2012.

[38] S. -M. Park, J. -H. Kim, Y. -B. Go and W. -S. Yoon, "Dynamic Grid replication strategy based on internet hierarchy", *Int. Workshop on Grid and Cooperative Computing, in: Lecture Note in Computer Science*, Vol. 1001, pp. 1324-1331, 2003.

[39] K. Sashi and A. Thanamani, "Dynamic replication in a Data Grid using a modified BHR region based algorithm", *Future Generation Computer Systems*, Vol. 27, No. 2, pp. 202-210, 2011.

[40] A. Horri, R.Sepahv and and Gh. Dastghaibyfard, "A hierarchical scheduling and replication strategy", *International Journal of Computer Science and Network Security*, Vol. 8, No. 8, pp. 30-35, 2008.

[41] D. G. Cameron, A. P. Millar, C. Nicholson, R. Carvajal-Schiaffino, F. Zini and K. Stockinger, "Optorsim: A simulation tool for scheduling and replica optimization in Data Grids", *in Int. Conf. for Computing in High Energy and Nuclear Physics (CHEP 2004)*, 2004.

[42] OptorSim-A Replica Optimizer Simulation: http://edg-wp2.web.cern.ch/edgwp2/optimization/optorsim.html, [Accessed January 2014].

[43] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar K. Stockinger and F. Zini, "Simulation of Dynamic Grid Replication Strategies in OptorSim", *in Proc. of the ACM/IEEE Workshop on Grid Computing*, 2002.

Najme Mansouri is currently a faculty of Computer Science at Shahid Bahonar University of Kerman. She received her M.Sc. in software engineering at Department of Computer Science & Engineering, College of Electerical& Computer Engineering, Shiraz University (Iran). She received her B.Sc. (Honor Student) in Computer Science from Shahid Bahonar University of Kerman, Iran in 2009. Her research interests include Parallel Processing, Distributed Systems, and Grid Computing.

Gholamhossein Dastghaibyfard received his M.Sc. and Ph.D. in Computer Science from Electrical Engineering and Computer Science Department, College of Engineering, University of Oklahoma, Norman Oklahoma USA in 1979 and 1990, respectively. He is currently an assistant professor of Computer Science at Department of Computer Science and Engineering, College of Electrical & Computer Engineering, Shiraz University. His current research interests include Parallel Algorithms, Grid Computing and Information Technology.