IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-3, Issue-7 E-ISSN: 2347-2693

RDT: A New Data Replication Algorithm for Hierarchical Data Grid

Sheida Dayyani[1], Mohammad Reza Khayyambashi[2*]

[1]*Department of Computer Engineering, Sheikh Bahaei University, Iran*
[2] *Department of Computer Engineering, University of Isfahan, Iran*

www.ijcseonline.org

Received: Jun/23/2015 Revised: July/06/2015 Accepted: July/21/2015 Published: July/30/ 2015

***Abstract*—** Grid computing is a type of distributed computing system that provides access to various computational resources which are shared by different organizations, in order to create an integrated powerful virtual computer. Nowadays, grid is known as an essential technology which is used for different kinds of high performance applications and it is believed that it will be applied more and more in the future as technology progresses. Data replication is a common method used in distributed environments to improve ease of data access and to provide a high level of data availability, increased fault tolerance and data reliability; and that's why this method is used for data management in data grid systems. Since the data files are very large and the Grid storages are limited, managing replicas in storage for the purpose of more effective utilization requires more attention. In this paper, a novel data replication strategy, called Replication with Dynamic Threshold (RDT) is proposed that uses a new threshold for characterizing the number of appropriate sites for replication. Appropriate sites have the higher number of access for that particular replica from other sites. It also minimizes access latency by selecting the best replica when various sites hold replicas. The simulated results with OptorSim, i.e. European Data Grid simulator show that the RDT strategy gives better performance compared to the other algorithms and prevents the unnecessary creation of replicas which leads to efficient storage usage.

***Keywords*—**Distributed systems; Data grid; Data replication; Dynamic Threshold; OptorSim.

I. INTRODUCTION

Computing infrastructure and network application technologies have come a long way over the past years and have become more and more detached from the underlying hardware platform on which they run. At the same time computing technologies have evolved from monolithic to open and then to distributed systems. Grid computing is a wide area distributed computing environment that enables sharing, selection, and aggregation of geographically distributed resources. Also, it is an important mechanism for utilizing distributed computing resources. These resources are distributed in different geographical locations, but are organized to provide an integrated service. The term "grid computing" refers to the emerging computational and networking infrastructure that is designed to provide pervasive and reliable access to data and computational resources over wide area networks, across organizational domains [1], [2], [3], [4].

Nowadays, there is a tendency of storing, retrieving, and managing different types of data such as experimental data that are produced from many projects [1]. This data plays a fundamental role in all kinds of scientific applications such as particle physics, high energy physics, data mining, climate modelling, earthquake engineering and astronomy, to cite a few, manage and generate an important amount of data which can reach terabytes and even petabytes, which need to be shared and analysed [2], [3], [5]. Storing such amount of data in the same location is difficult, even impossible. Moreover, an application may need data produced by another geographically remote application. For this reason, a grid is a large scale resource sharing and problem solving mechanism in virtual organizations and is suitable for the above situation [6], [7], [8]. In addition, users can access important data that is available only in several locations, without the overheads of replicating them locally. These services are provided by an integrated grid service platform so that the user can access the resource transparently and effectively [1], [6].

One class of grid computing is "Data Grids"; that provide geographically distributed storage resources to large computational problems that require evaluating and mining large amounts of data [9], [10]. The Grid resources, including computing facility, data storage and network bandwidth, are consumed by the jobs. For each incoming job, the grid scheduler decides where to run the job based on the job requirements and the system status. In data-intensive applications, the locations of data required by the job impact the Grid scheduling decision and performance greatly. Creating data replicas can reroute the data requests to certain replica servers and offer remarkably higher access speed than a single server. At the same time, the replicas provide broader decision space for the grid scheduler to achieve better performance from the perspective of the job [2], [3], [10]. Managing this data in a centralized location increases the data access time and hence much time is taken to execute the job. So to reduce the data access time,

"Replication" is used [2], [3].

Replication is an important technique to speed up data access for data grid systems that replicate the data in multiple locations; so that a user can access the data from a site in its vicinity. It has been shown that data replication not only reduces access costs, but also increases data availability in many applications [9], [11], [12]. Also, data replication is a practical and effective method to achieve efficient network performance in network bandwidth constrained environment, and it has been applied widely in the areas of distributed database and Internet. New challenges are faced in the data grid, for example, huge data file sizes, system resources belonging to multiple owners, dynamically changing resources and complicated cost model [2], [9].

The experiments on the distributed systems show that the replication promotes higher data availability, lower bandwidth consumption, increase in fault tolerance and improvement in scalability. In other words, the replication is the process of creation and placement of the copies of entities software. The phase of creation consists in reproducing the structure and the state of the replicated entities, whereas the phase of placement consists in choosing the suitable slot of this new duplication, according to the objectives of the replication. So, replication strategy can shorten the time of fetching the files by creating many replicas stored in appropriate locations [10], [13].

Thus, Replication causes three important features in grid systems, such as follow [14]:

- Increased availability
- Increased performance
- Enhanced fault tolerance and reliability

By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find the data close to the site where the request originated, thus increasing the performance of the system. But the benefits of replication, of course, do not come without overheads of creating, maintaining and updating the replicas. If the application has a read-only nature, replication can greatly improve the performance. But, if the application needs to process update requests, the benefits of replication can be neutralised to some extent by the overhead of maintaining consistency among multiple replicas [14].

There is a fair amount of work on data replication in grid environments. Most of the existing work focused on mechanisms for create, decision and delete replicas. The purpose of this document is to present a novel replication technique and compare it with previous techniques which have been presented by other researches.

The rest of this paper is organized as follows. In the second section, we present an overview of grid systems and describe replication scenario, challenges and parameters of evaluating replication techniques. Section three takes a closer look on basic and new existing data replication strategies in grid environment. In section four, a novel replication algorithm is proposed. Section five shows the simulation results. Finally, conclusions and some future research works are presented in Section six.

II. GRID SYSTEMS

A large number of scientific and engineering applications require a huge amount of computing time to carry out their experiments by simulation. Research driven by this has promoted the exploration of a new architecture known as "The Grid" for high performance distributed application and systems [12]. In [13], Foster defines the Grid concept as "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations". There are different types of Grid developed to emphasize special functions that will be defined in the next section.

A. *Types of Grid*

Grid computing can be used in a variety of ways to address various kinds of application requirements and it has three primary types. Of course, there are no hard boundaries between these grid types and often grids may be a combination of two or more of these [14]. Types of grids are summarized below:

- Computational grid: Computational grid is focused on setting aside resources specifically for computing power. Such as most of the machines are high-performance servers [14].
- Scavenging grid: Scavenging grid is most commonly used with large numbers of desktop machines that are scavenged for available CPU cycles and other resources. Owners of the desktop machines are usually given control over when their resources are available to participate in the grid [14].
- Data grid: Data grid is a collection of geographically distributed computer resources that these resources may be located in different parts of a country or even in different countries [10]. For example, you may have two universities doing life science research, each with unique data. A grid connects all these locations and enables them to share their data, manage the data, and manage security issues such as who has access to which data [15], [16].

B. *Data Management in Grid*

An important technique for data management in grid systems is the replication technique. Data replication is characterized as an important optimization technique in Grid for promoting high data availability, low bandwidth

consumption, increased fault tolerance, and improved scalability. The goals of replica optimization is to minimize file access times by pointing access requests to appropriate replicas and pro-actively replicating frequently used files based on access statistics gathered.

Generally, replication mechanism determines which files should be replicated, when the new replicas should be created and where the new replicas should be placed [4], [9], [15].

The main aims of using replication are to reduce access latency and bandwidth consumption. The other advantages of replication are that it helps in load balancing and improves reliability by creating multiple copies of the same data [4], [15].

Replication schemes can be classified as static and dynamic. In **static replication**, a replica persists until it is deleted by users or its duration is expired. The drawback of static replication is evident when client access patterns change greatly in the Data. Static replication can be used to achieve some of the above mentioned goals but the drawback with static replication is that it cannot adapt to changes in user behavior. The replicas have to be manually created and managed if one were to use static replication. But, in **dynamic replication**, replica creation, deletion and management are done automatically. Dynamic strategies have the ability to adapt to changes in user behavior [17].

Various combinations of events and access scenarios of data are possible in a distributed replicated environment. The three fundamental questions any replica placement strategy has to answer are as follow that Depending on the answers, different replication strategies are born [4], [15]:

- When should be the replicas created?
- Which files should be replicated?
- Where should be the replicas placed?

Using replication strategies in grid environment may cause some challenges. The four important challenges in replicated environments are as follow [11]:

- Time of creation of a new replica: If strict data consistency is to be maintained, performance is severely affected if a new replica is to be created. As sites will not be able to fulfill request due to consistency requirements.
- Data Consistency: Maintaining data integrity and consistency in a replicated environment is of prime importance. High precision applications may require strict consistency of the updates made by transactions.
- Lower write performance: Performance of write operations can be dramatically lowered in applications requiring high updates in replicated environment, because the transaction may need to update multiple copies.
- Overhead of maintenance: If the files are replicated at more than one site, it occupies storage space and it has to be administered. Thus, there are overheads in storing multiple files.

Almost all the replications strategies try to reduce the access latency thus reducing the job response time and hence increase the performance of the grids. Similarly almost all the replication strategies try to reduce the bandwidth consumption to improve the availability of data and performance of the system. The target is to keep the data as close to the user as possible, so that data can be accessed efficiently. Some of the replication strategies explicitly target to provide a balanced workload on all the data servers. This helps in increasing the performance of the system and provides better response time. With more number of replicas in a system the cost of maintaining them becomes an overhead for the system. Some of the strategies aim to make only an optimal number of replicas in the data grid. This ensures that the storage is utilized in an optimal way and the maintenance cost of replica is minimized. Some strategies target the strategic placement of the replicas along with an optimal number of replicas. The strategic placement of replicas is a very important factor because it is integrated with few other very important factors. For example, if the replicas are placed on the optimal locations it helps to optimize the workload of different servers. It is also related with the cost of the maintenance. If a strategy goes on replicating a popular file blindly, it will create too many replicas thus increasing the burden for the system as replica maintenance costs will become too high [18].

Job execution time is another very important parameter. Some replication strategies target to minimize the job execution time with optimal replica placement. The idea is to place the replicas closer to the users in order to minimize the response time, and thus the job execution time. This will increase the throughput of the system [18]. Only a few replication strategies have considered replication as an option to provide fault tolerance and quality assurance. All replication strategies use subset of these parameters [18].

III. RELATED WORKS

The role of a replication strategy is to identify when a replica should be created, where to place replicas, when to remove replicas and how to locate the best replica. These strategies are guided by factors such as demand for data, network conditions, cost of transfer and storage cost [19].

Several replication replacement strategies have been proposed in the past and they are the basics of other replication algorithms. Details of some important replication algorithms are as follows:

NO Replication strategy will not create replica and therefore, the files are always accessed remotely. One example of the implemented strategy is the SimpleOptimizer algorithm [20], which never performs replication; rather it reads the required replica remotely. SimpleOptimizer algorithm is simple to implement and performs the best

relative to other algorithms in terms of the storage space usage, but performs the worst in terms of job turnaround time and network usage [17].

Best client creates replica at the client that has generated the most requests for a file, this client is called the best client. At a given time interval, each node checks to see if the number of requests for any of its file has exceeded a threshold, then the best client for that file is identified [17].

Cascading Replication supports tree architectur. The data files generated in the top level and once the number of accesses for the file exceeds the threshold, then a replica is created at the next level, but on the path to the best client, and so on for all levels, until it reaches to the best client itself [17].

Plain Cashing: The client that requests a file stores a copy locally. If these files are large and a client has enough space to store only one file at a time, then files get replaced quickly [17].

Cashing plus Cascading combines cascading and plain cashing strategies. The client caches file locally, and the server periodically identifies the popular files and propagates them down the hierarchy. Note that the clients are always located at the leaves of the tree but any node in the hierarchy can be a server. Specifically, a Client can act as a Server to its siblings. Siblings are nodes that have the same parent [17].

Fast Spread: In this method a replica of the file is stored at each node along its path to the client. When a client requests a file, a copy is stored at each tier on the way. This leads to a faster spread of data. When a node does not have enough space for a new replica it deletes the least popular file that had come in the earliest [17].

Least Frequently Used (LFU) strategy always replicates files to local storage systems. If the local storage space is full, the replica that has been accessed the fewest times is removed and then releases the space for new replica. Thus, LFU deletes the replica which has less demand (less popularity) from the local storage even if the replica is newly stored [21].

Least Recently Used (LRU) strategy always replicates files to local storage system. In LRU strategy, the requested site caches the required replicas, and if the local storage is full, the oldest replica in the local storage is deleted in order to free the storage. However, if the oldest replica size is less than the new replica, the second oldest file is deleted and so on [21].

Proportional Share Replica (PSR) policy is an improvement in Cascading technique. The method is a heuristic one that places replicas on the optimal locations by assuming that the numbers of sites and the total replicas to be distributed are already known. Firstly an ideal load distribution is calculated and then replicas are placed on candidate sites that can service replica requests slightly greater than or equal to that ideal load [22].

Bandwidth Hierarchy Replication (BHR) is a novel dynamic replication strategy which reduces data access time by avoiding network congestions in a data grid network. With BHR strategy, we can take benefits from "network-level locality" which represents that required file is located in the site which has broad bandwidth to the site of job execution.

BHR strategy was evaluated by implementing in Optorsim; a data grid simulator initially developed by European data grid projects. The simulation results show that BHR strategy can outperform other optimization techniques in terms of data access time when hierarchy of bandwidth appears in Internet. BHR extends current site-level replica optimization study to the network-level [23].

Simple Bottom-Up (SBU) and **Aggregate Bottom-Up (ABU)** are two dynamic replication mechanisms that are proposed in the multi-tier architecture for data grids.

The SBU algorithm replicates the data file that exceeds a pre-defined threshold for clients. The main shortcoming of SBU is the lack of consideration to the relationship with historical access records. For the sake of addressing the problem, ABU is designed to aggregate the historical records to the upper tier until it reaches the root.

The performance of algorithms were evaluated and improvements shown against Fast Spread dynamic replication strategy. The values for interval checking and threshold were based on data access arrival rate, data access distribution and capacity of the replica servers [18].

Multi-objective approach is a method exploiting operations research techniques that is proposed for replica placement. In this method, replica placement decision is made considering both the current network status and data request pattern. The problem is formulated in p-median and p-center models to find the p replica placement sites. The p-center problem targets to minimize the max response time between user site and replica server whereas the p-median model focuses on minimizing the total response time between the requesting sites and the replication sites. The dynamic maintainability is achieved by considering the replica relocation cost.

The decision of relocation is made when performance metric degrades significantly in last K time periods. The threshold value is varied proportionally to response time in each time interval [24], [25].

Weight-based dynamic replica replacement strategy calculates the weight of replica based on the access time in the future time window, based on the last access history. After

that, calculate the access cost which embodies the number of replicas and the current bandwidth of the network. The replicas with high weight will be helpful to improve the efficiency of data access, so they should be retained and then the replica with low weight will not make sense to the rise of data access efficiency, and therefore, should be deleted. The access history defines based on the zip-like distribution [26].

Latest Access Largest Weight (LALW) is a dynamic data replication mechanism. LALW selects a popular file for replication and calculates a suitable number of copies and grid sites for replication. By associating a different weight to each historical data access record, the importance of each record is differentiated. A more recent data access record has a larger weight. It indicates that the record is more pertinent to the current situation of data access [27].

Agent-based replica placement algorithm is proposed to determine the candidate site for the placement of replica. For each site that holds the master copies of the shared data files will deploy an agent.

The main objective of an agent is to select a candidate site for the placement of a replica that reduces the access cost, network traffic and aggregated response time for the applications. Furthermore, in creating the replica an agent prioritizes the resources in the grid based on the resource configuration, bandwidth in the network and insists for the replica at their sites and then creates a replica at suitable resource locations.

The agent in this approach is autonomous, self-contained software capable of making independent decisions. There are two important issues that are considered in this strategy, which are:

1. Choosing a replica location is to place a replica at sites that optimize the aggregated response time.
2. Choosing a replica location is to place a replica at sites that optimize the total execution time of the jobs executed in the grid.

Response time is calculated by multiplying the number of requests at site with the transmission time between the nearest replication site to the requester and the sum of the response times for all sites constitutes the aggregated response time. Based on resource factors that influence the data transmission time between the sites is the decision must be made by an agent at each site. The factors include:

- Baud-rate between the sites
- CPU rating and CPU load,
- Site storage capacity
- Local demand of the replicas at each site

In order to evaluate the resource properties and grade with an appropriate rank, the agent uses a multi-dimensional ranking function.

The agent preferences are represented by a set of factor weightings, which allow resource rank to be tailored to the current resource characteristics [7].

Adaptive Popularity Based Replica Placement (APBRP) is a new dynamic replica placement algorithm, for hierarchical data grids which is guided by “file popularity”. The goal of this strategy is to place replicas close to clients to reduce data access time while still using network and storage resources efficiently. The effectiveness of APBRP depends on the selection of a threshold value related to file popularity. APBRP determines this threshold dynamically based on data request arrival rates [28].

Efficient Replication strategy is a new replication strategy for dynamic data grids, which take into account the dynamic of sites. This strategy can increase the file availability, improved the response time and can reduce the bandwidth consumption. Moreover, it exploits the replicas placement and file requests in order to converge towards a global balancing of the grid load. This strategy will focus on read-only-access as most grids have very few dynamic updates because they tend to use a "load" rather than "update" strategy.

There are three steps provided by this algorithm, which are [29]:

1. Selection of the best candidate files for replication; Selected based on requests number and copies number of each files.
2. Determination of the best sites for files placement which are selected in the previous step; Selected based on requests number and utility of each site regarding to the grid.
3. Selection of the best replica; Taking account the bandwidth and the utility of each site [29].

Value-based replication strategy (VBRS) is proposed to decrease the network latency and meanwhile to improve the performance of the whole system.

In VBRS, threshold was made to decide whether to copy the requested file, and then solve the replica replacement problem. VBRS has two steps:

1. The threshold to decide whether a file should be replicated in the local storage device is introduced according to the access history and the storage capacity.
2. A measure based on the values of the local replicas, is devised to choose the replica that should be replaced.

At the first steps, the threshold will be calculated to decide whether the requested file should be copied in the local storage site. Then at the second stage, the replacement algorithm will be triggered when the requested file needs to be copied at the local storage site does not have enough space.

It firstly, calculates the file’s values in the local storage site. The files that have the least value will be deleted by the replacement algorithm. The file’s value mostly concerns with three factors: network bandwidth, file’s size, and the access

history. The files with higher value should be retained, and then the files with lower value will be deleted.

The replica replacement policy is developed by considering the replica's value which is based on the file's access frequency and access time. To evaluate the performance of the VBRS, the grid simulator Optorsim is used that can simulate the real data grid environment. The experiment results show that the effectiveness of VBRS algorithm can reduce network latency [30].

Enhance Fast Spread (EPS) is an enhanced version of Fast Spread for replication strategy in the data grid. This strategy was proposed to improve the total of response time and total bandwidth consumption. Its takes into account some criteria such as the number and frequency of requests, the size of the replica and the last time the replica was requested.

EFS strategy keeps only the important replicas while the other less important replicas are replaced with more important replicas. This is achieved by using a dynamic threshold that determines if the requested replica should be stored at each node along its path to the requester. This strategy takes four factors or criteria that have been stated as above into consideration when calculating the threshold:

1. The number of requests shows the sums or how many times the replica has been requested by its node.
2. The frequency of requests shows how many times the replica has been requested by its node within a specific time interval.
3. The size of replica is also a significant factor in deciding if the replica should be stored.
4. The number and frequency of requests in addition to the last time the replica was requested give a hint of the probability of requesting the replica again [31].

Predictive hierarchical fast spread (PHFS) is a new dynamic replication method in multi-tier data grid environments which is an improve version of common fast spread. The fast spread is a dynamic replication method in the data grid. The multi-tier is a tree-like structure to build data grid. The PHFS tries to forecast future needs and pre-replicates the min hierarchal manner to increase locality in accesses and improve performance that consider spatial locality. This method is able to optimize the usage of storage resources, which not only replicates data objects hierarchically in different layers of the multi-tier data grid for obtaining more localities in accesses. It is a method intended for read intensive data grids.

In PHFS, to predict future needs and pre-replicates them hierarchically in different nodes of different tiers in the multi-tier data grid on the path from the source node to the client by using predictive methods. The nodes in upper layers of multi-tier data grid have more storage capacity and computational power than the lower level nodes. Also, the bandwidth of the links among nodes in upper levels is greater than the links in the lower level nodes. So, the replication method in fast spread can replicate more items in upper level nodes. Otherwise, it can decrease the amount of replicated items on the path to lower levels toward the client. In order to optimize the utilization of resources for replication and provide maximum locality with available resources the hierarchal replication is used.

The PHFS method use priority mechanism and replication configuration change component to adapt the replication configuration dynamically with the obtainable condition. Besides that, it is developed on the basis of the concept that users who work on the same context will request some files with high probability.

Although common fast spread makes some improvements in some metrics of performance like bandwidth consumption, it shows poor results in local accesses patterns. Therefore PHFS is used by predicting user's subsequent file demands and pre-replicate them earlier in hierarchal manner to increase locality in accesses.

The results show that PHFS causes lower latency and better performance compared with common fast spread. Moreover, compared with common fast spread it showed that the most of the accesses in PHFS occur in lower levels. Besides that, it is more suitable for applications wherein the clients work on a context for a period of time and the requests of clients are not random, like scientific applications that researchers work on a project [32].

Dynamic Hierarchical Replication (DHR) is a dynamic replication algorithm for hierarchical structure that places replicas in appropriate sites. Best site has the highest number of access for that particular replica.

The algorithm minimizes access latency by selecting the best replica when various sites hold replicas. The replica selection strategy that proposed in [33], selects the best replica location for the users running jobs by considering the replica requests that waiting in the queue and data transfer time. It stores the replica in the best site where the file has been accessed most, instead of storing files in many sites.

Modified Latest Access Largest Weight (MLALW) is a dynamic data replication strategy. This strategy is an enhanced version of Latest Access Largest Weight strategy. MLALW deletes files by considering three important factors:

1. Least frequently used replicas
2. Least recently used replicas
3. 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 Optorsim data grid simulator. The experiment results show that MLALW strategy gives a better performance compared to the other algorithms and prevents unnecessary creation of replica which leads to efficient storage usage [34].

IV. PROPOSED DATA REPLICATION ALGORITHM

In this section, first network structure of data grid is described, and then the novel RDT algorithm is proposed.

A. *Grid Network Structure*

The grid topology of the simulated platform is given in Figure 1, which is based on European Data Grid CMS test-bed architecture [35] and has three levels similar to what is given by Horri et al. [36].

1) First level are Regions that are connected through internet i.e. have low bandwidth.
2) Second level comprises of LAN's (local area network) within each region that have moderately higher bandwidth compared to the first level.
3) The third level is the sites within each of the LAN's that are connected to each other with a high bandwidth.

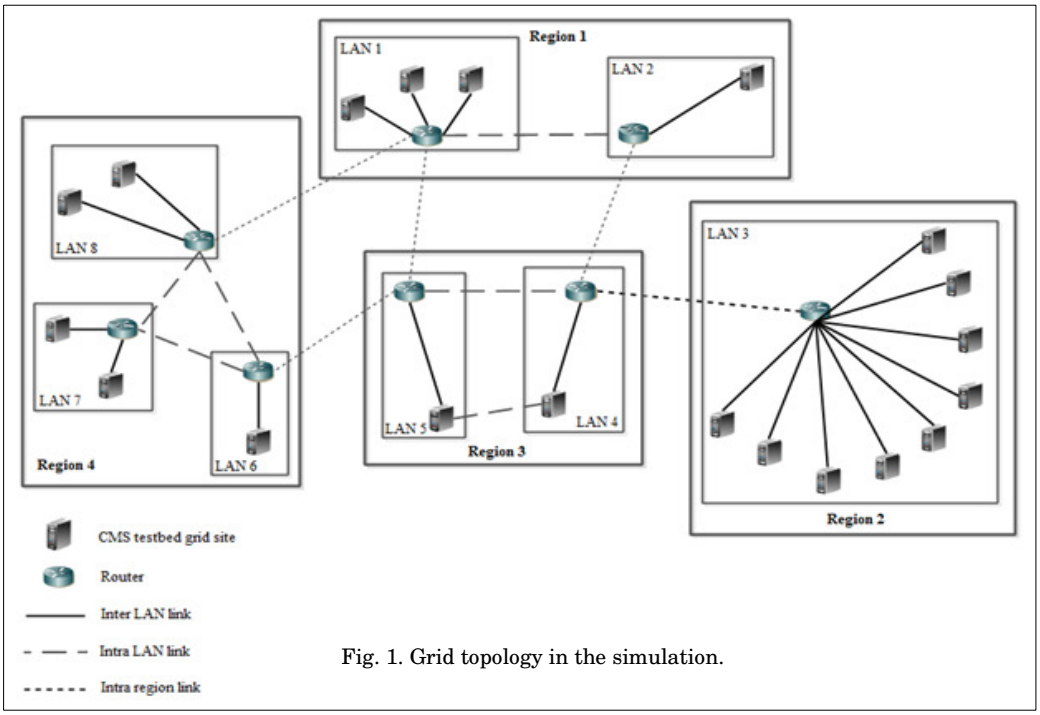

Fig. 1. Grid topology in the simulation.

B. *Replication with Dynamic Threshold*

When a job is allocated to the grid scheduler, before job execution the replica manager should transfer all the required files that are not available in the local site. So, the data replication enhances the job scheduling performance by decreasing job execution time. RDT is a novel dynamic hierarchical replication strategy and has three parts:

1) Replica Selection

Generally when several replicas are available within the local LAN, the local region or other regions, RDT selects the site that has the most storage space. Figure 2 describes RDT's selection algorithm.

Algorithm 1 : Replica Selection

```
   //locate unavailable requested files (URF)
   //replica selection
1  For each (URF f_i in the local site) {
2  If (f_i available in local LAN){
3     Create list L of f_i`s that are available in local LAN.
4     Select f_i from L that has most storage space.
5     Continue; } //end if

6  If (f_i available in local region) {
7     Create list L of f_i`s that are available in local region.
8     Select f_i from L that has most storage space.
9     Continue; } //end if

10 Create list L of f_i`s that are available in other regions.
11 Select f_i from L that has most storage space.
12  } //end for each step 1
```

Fig. 2. Selection algorithm of DHRT strategy.

2) Replica Decision

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 sites (BSEs). To select the BSEs, RDT characterizes the number of appropriate sites for replication by a dynamic threshold. Decision algorithm calculates this threshold from Eq. (1).

Then, RDT creates a sorted list (by number of replica access) of all SEs that requested the particular file in the region. Now the replica will be placed in the storage elements of the above sorted list that threshold shows.

$$Dynamic\ Threshold = \frac{NA(f) \times NS(f)}{NR(f) \times S(f)} \quad (1)$$

The dynamic threshold allows to candidate more than one SE and increase number of replicas. On the other hand, replica is not placed in all the requested sites. Hence, RDT helps to find an appropriate number of replicas based on four factors, such as:

- NA(f): Number of access to that particular file
- NS(f): Number of sites that will require that particular file
- NR(f): Number of existing replicas of that particular file
- S(f): Size of that particular file

Figure 3 describes RDT's decision algorithm.

Algorithm 2 : Replica Decision

```
   //Now all requested files are available in the local site
   //replica placement
1  Execute the job
   //Now replicate each unavailable requested files (URF)
   //in the best SEs (BSE) that dynamic threshold shows
2  For each (Ri ϵ URF) {
3  Run dynamic threshold function to choose best
4        number of replicas (BNR) for storing Ri
5  While (BNR>0){
6     Select a BSE for storing Ri.
      //in this step local LAN means the LAN that holds BSE
      //also local region means the region that holds BSE
7     SRi← size of Ri;
8     SBSE □ storage size of best SE;
      //don't replicate if size of Ri ≥ storage size of BSE;
9     If (Ri ≥ SBSE) Continue;//not enough space
10    If (enough space exist for Ri in BSE) {
11       Store Ri;
12       Continue; }
      //don't replicate if Ri is available in the local LAN
13    If (Ri is available in the local LAN) Continue;
14 }//end of while
```

Fig. 3. Decision algorithm of DHRT strategy.

3) Replica Replacement

If enough storage space exists in the local site, the selected file will be replicated. Otherwise if the file is available in the local LAN, then it will be 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 LFU (least Frequently Used) sorted list of replicas that are both available in the current site as well as the local LAN.
- Start deleting files from the above list till space is available for replica.
- If space is still insufficient, then repeat previous step for each LAN in current region, randomly. In other word, 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, generate a LFU sorted list of the remaining files in the site and start deleting files from the above list till space is available for replica.

Figure 4 describes RDT's replacement algorithm.

Algorithm 3 : Replica Replacement

```
   //Now delete those files from BSE that are also
   //available in the local LAN
1  Create list L of fi`s that are both available in the BSE
      as well as in the local LAN.
2  Sort list L using LFU.
3  While (L is not empty && not enough space for Ri)
4     Delete first file from list L in BSE;
5  If (enough space exists for Ri in BSE){
6     Store Ri;
7  Continue; }
   //Now delete those files from BSE that are also available
   //in the local region
8  Create list L of fi`s that are both available in the BSE
      as well as in the local region.
9  Sort list L using LFU.
10 While (L is not empty && not enough space for Ri)
11       Delete first file from list L in BSE;
12 If (enough space exists for Ri in BSE) {
13       Store Ri;
14       Continue;
15 }
     //Now delete from remaining files in BSE
17 Create list L from remaining files in BSE.
18 Sort list L using LFU.
19 While (L is not empty && not enough space for Ri)
20    Delete first file from list L in BSE;
21 StoreRi;
22 }//end for each step 3
```

Fig. 4. Replacement algorithm of DHRT strategy.

V. EXPERIMENTAL EXPERIENCE

A. Simulation Tool

OptorSim is used to evaluate the performance of RDT algorithm. OptorSim [37] was developed by European Data Grid projects and is written in Java. It provides a framework to simulate the real grid environment. It is

developed to test the dynamic replication strategies. In data grid environment various job execution scenarios are present.

OptorSim has several important components such as computing element (CE), storage element (SE), resource broker (RB), replica manager (RM), and replica optimizer (RO). Computing elements and storage elements are used to execute grid jobs and store the files respectively. OptorSim architecture is shown in Figure 5. Additional details about OptorSim are available in the literatures [21], [37].

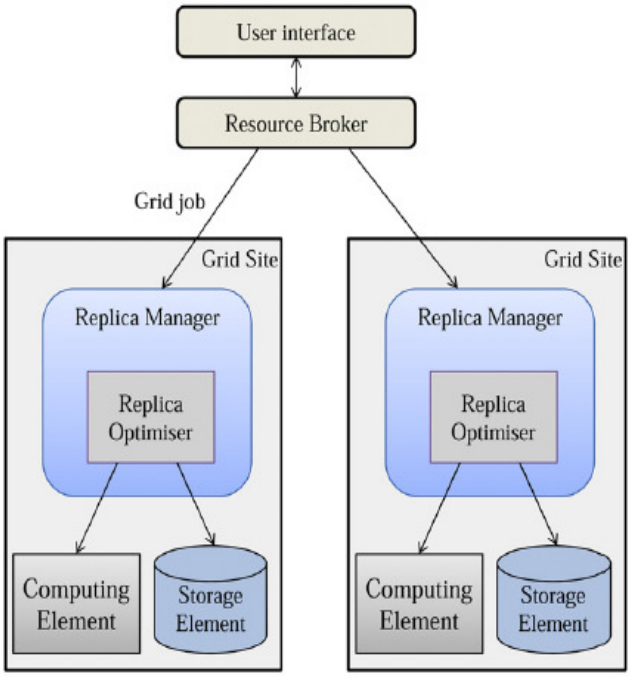

Fig. 5. OptorSim architecture.

B. *Evaluation Parameters*

The architecture used here is the European Data Grid CMS testbed architecture [35]. In this there are twenty sites in which two of them have only storage element and which acts as master node. There are 8 routers which is used to forward request to other sites.

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. It is assumed the network has four regions and on average two LAN's in every region. The average storage capacity is 24.25 GB. Bandwidth in each level is given in Table 1. Also, Table 2 specifies the simulation parameters and their values used in our study. Data replication strategies commonly assume that the data is read only in Data Grid Environments.

Table 1. Bandwidth configuration.

Parameter	Value (Mbps)
Inter LAN bandwidth	1000
Intra LAN bandwidth	100
Intra Region bandwidth	10

Table 2. General configuration parameters.

Parameter	Value
Number of jobs types	6
Job Delay (ms)	2500
Maximum queue size	200
Number of jobs	100
Average size of storage elements (GB)	54.25
Size of each file (GB)	10

C. *Results and Discussion*

The proposed RDT algorithm is compared with four replication algorithms namely, No Replication, Least Frequently Used (LFU), Least Recently Used (LRU), and DHR. In No Replication strategy files are accessed remotely. When storage is full, LRU deletes least recently accessed files and LFU deletes least frequency accessed files. The DHR algorithm 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 me.

Figure 6 shows the mean job execution time for the various replication algorithms. Obviously, the No Replication strategy has the worst performance as all the files requested by jobs have to be transferred from main site. In this simulation LRU and LFU have almost the same execution time. DHR improves data access time by considering the differences between intra-LAN and inter-LAN communication. RDT mean job execution time is faster than other algorithms since it considers a dynamic threshold to distinguish the best number of replicas. If the available storage for replication is not enough, RDT

will not delete those file that have a high transferring time. It also improves the mean job execution time by storing the replica in the most frequently accessed site of the requested region. Valuable information can be gained by monitoring storage resources usage. Since resource cost is proportional to the resource used, so minimizing storage usage is a must.

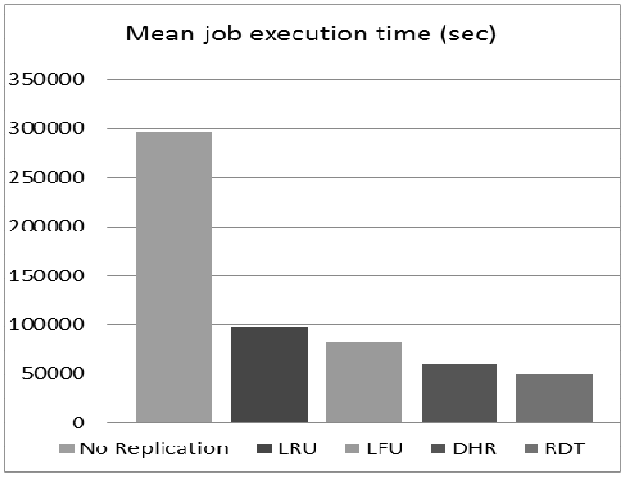

Fig. 6. Job execution time for various replication algorithms.

Figure 7 shows the storage usage which is the percentage of available spaces that are used. No Replication strategy has best storage since it gets files remotely. LFU and LRU are always replicate when a request is made; hence they use a lot of space. DHR strategy performs better than the previous three strategies since it keeps at most one copy of file in the region. The proposed RDT strategy has minimum storage usage among the current algorithms because it place replicas in the appropriate sites so reduces unnecessary replication.

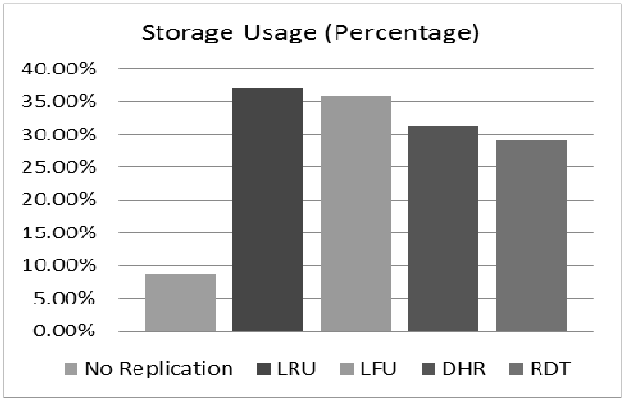

Fig. 7. Mean storage usage for various replication algorithms.

Figure 8 displays the mean job time based on changing number of jobs for LRU, DHR and proposed algorithm. It is clear that at the job number increases, RDT 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.

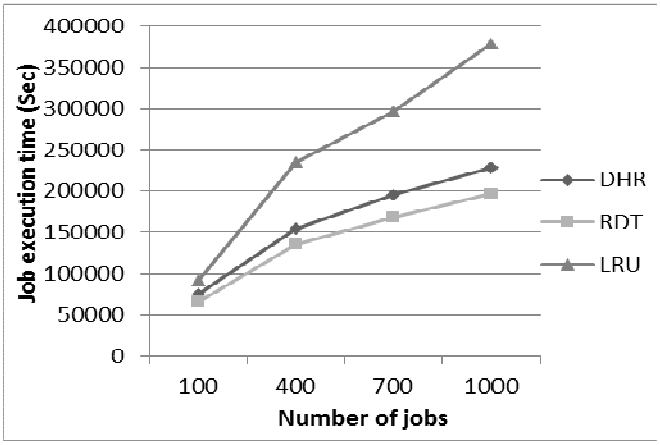

Fig. 8. Mean job execution time based on varying number of jobs

VI. CONCULATION ANDFUTURE WORKS

In Data Grid is the highlight in the development of the Grid technology, which can be treated as a suitable solution for high performance and data-intensive computing applications. Replication is a technique used in grid environments that helps to reduce access latency and network bandwidth utilization. Replication also increases data availability thereby enhancing system reliability. This technique appears clearly applicable to data distribution problems in large scale scientific collaborations, due to their globally distributed user communities and distributed data sites. Data replication strategies depend on the followings: location of data, time and method of data creation, and the method that data is destroyed.

Despite the large amount of research done on the dynamic data replication topic, this system still faces many challenges, such as: long job execution time, file access delay, limited accessible storage resources, and lack of an integrated strategy for replication and scheduling.

The main objective of this paper is to introduce a novel methods used for data replication in a three level hierarchical structure network. Most of the presented algorithms create only one replica at a time, which may reduce the performance and increase the bandwidth consumption in case there are a lot of requests to access the file this happens due to not having enough replicas created. Thus, setting a dynamic threshold which defines number of required file replicas based on the grid

environment conditions is necessary and can minimize the execution time and improve performance. It should be mentioned that using the proposed strategy, file replicas will not be created on all of the sites that requested the file, which results in reducing the storage cost and shorter execution time. Also, in RDT strategy file replicas will be created only in the suitable sites which are dynamically identified.

To evaluate the efficiency of the proposed replication strategy, grid simulator OptorSim is configured to represent a real world data grid testbed. The simulation results shows that proposed algorithm performs better when compared to other traditional algorithm such as LRU, LFU and no replication and a new algorithm such as DHR.

In future work, more realistic scenarios and user access patterns can be investigated and the RDT algorithm can be combined with a proper scheduling to improve performance. We also plan to investigate more replica replacement strategies to further improve the overall system performance. Data transferring between different grid sites is time consuming and consequently scheduling jobs to the appropriate sites is necessary. Replica selection can also be extended by considering additional parameters such as security. Searching for advanced replica replacement methods certainly enhances replication strategies.

REFERENCES

[1] M. Li and M. Baker, "*The grid core technologies*", *John Wiley & Sons*, **2005**.

[2] N. Mohd. Zin, A. Noraziah, A. Che Fauzi, and T. Herawan, "Replication Techniques in Data Grid Environments", in *Intelligent Information and Database Systems*, vol. 7197, Eds. Springer Berlin, Heidelberg, pp. 549–559, **2012**.

[3] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications", in *11th IEEE International Symposium on High Performance Distributed Computing,* pp. 352–358, **2002**.

[4] K. Sashi and A.S. Thanamani, "A new replica creation and placement algorithm for data grid environment", in *International Conference on Data Storage and Data Engineering (DSDE)*, pp. 265–269, **2010**.

[5] S. Naseera and K.V.M. Murthy, "Agent Based Replica Placement in a Data Grid Environement", in *First International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 426–430, **2009**,.

[6] F.B. Charrada, H. Ounelli, and H. Chettaoui, "Dynamic period vs static period in data grid replication", in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pp. 565–568, **2010**.

[7] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data management in an international data grid project", in *Grid Computing*, Springer, pp. 77–90, **2000**.

[8] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets", in *Network Computing Application*, vol. 23, no. 3, pp. 187–200, **2000**.

[9] I. Foster and C. Kesselman, "*The Grid 2: Blueprint for a New Computing Infrastructure*", Morgan Kaufmann, **2003**.

[10] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing", in *Acm Computing Surveys*, vol. 38, no. 1, p. 3, **2006**.

[11] M. Mat Deris, J.H. Abawajy, and A. Mamat, "An efficient replicated data access approach for large-scale distributed systems",in *Future generation computer systems*, vol. 24, no. 1, pp. 1–9, **2008**.

[12] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman, "Data replication strategies in grid environments", in *Fifth International Conference on Algorithms and Architectures for Parallel Processing,* pp. 378 –383, **2002**.

[13] K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities", in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid,* pp. 376–376, **2002**.

[14] S. Goel and R. Buyya, "Data replication strategies in wide area distributed systems", in *Enterprise Service Computing: From Concept to Deployment*, vol. 17, **2006**.

[15] R. Buyya, D. Abramson, and J. Giddy, "An architecture for a resource management and scheduling system in a global computational grid", in *The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region,* vol. 1, pp. 283–289, **2000**.

[16] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", in *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, **2001**.

[17] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid", *Grid Computing,* pp. 75–86, **2001**.

[18] M. Tang, B.S. Lee, C.K. Yeo, and X. Tang, "Dynamic replication algorithms for the multi-tier Data Grid", in *Future Generation Computer Systems*, vol. 21, no. 5, pp. 775–790, **2005**.

[19] K. Sashi and A.S. Thanamani, "Dynamic replication in a data grid using a Modified BHR Region Based Algorithm", in *Future Generation Computer Systems*, vol. 27, no. 2, pp. 202–210, **2011**.

[20] J. Gwertzman, "M. seltzer: The Case for Geographical Push-Cashing," in *5th Conference on Hot Topics in Operating systems,* Orcas Island, USA, **1995**.

[21] D.G. Cameron, R. Carvajal-Schiaffino, A.P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "OptorSim: a grid simulator for replica optimisation", in *UK e-science all hands conference*, vol. 31, **2004**.

[22] J. Abawajy, "Placement of File Replicas in Data Grid Environments", in *Computational Science (ICCS),* vol. 3038, Springer Berlin, Heidelberg, pp. 66–73, **2004**.

[23] S.M. Park, J.H. Kim, Y.B. Ko, and W.S. Yoon, "Dynamic data grid replication strategy based on Internet hierarchy", in *Grid and Cooperative Computing*, pp. 838–846, **2004**.

[24] R.M. Rahman, K. Barker, and R. Alhajj, "Replica placement design with static optimality and dynamic maintainability",

in *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, vol. 1, pp.4, **2006**.

[25] R.M. Rahman, K. Barker, and R. Alhajj, "Replica placement in data grid: a multi-objective approach", in *Grid and Cooperative Computing (GCC)*, Springer, pp. 645–656, **2005**.

[26] W. Zhao, X. Xu, N. Xiong, and Z. Wang, "A weight-based dynamic replica replacement strategy in data grids", in *IEEE Asia-Pacific Services Computing Conference,(APSCC)*, pp. 1544–1549, **2008**.

[27] R.S. Chang and H.P. Chang, "A dynamic data replication strategy using access-weights in data grids", in *The Journal of Supercomputing*, vol. 45, no. 3, pp. 277–295, **2008**.

[28] M. Shorfuzzaman, P. Graham, and R. Eskicioglu, "Adaptive popularity-driven replica placement in hierarchical data grids", in *The Journal of Supercomputing*, vol. 51, no. 3, pp. 374–392, **2010**.

[29] F.B. Charrada, H. Ounelli, and H. Chettaoui, "An efficient replication strategy for dynamic data grids", in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC),* pp. 50–54, **2010**.

[30] W. Zhao, X. Xu, Z. Wang, Y. Zhang, and S. He, "Improve the performance of data grids by value-based replication strategy", in *Sixth International Conference on Semantics Knowledge and Grid (SKG)*, pp. 313–316, **2010**.

[31] M. Bsoul, A. Al-Khasawneh, E.E. Abdallah, and Y. Kilani, "Enhanced fast spread replication strategy for data grid", in *Journal of Network and Computer Applications* , vol. 34, no. 2, pp. 575–580, **2011**.

[32] L.M. Khanli, A. Isazadeh, and T.N. Shishavan, "PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid", in *Future Generation Computer Systems*, vol. 27, no. 3, pp. 233–244, **2011**.

[33] N. Mansouri and G.H. Dastghaibyfard, "A dynamic replica management strategy in data grid", in *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1297–1303, **2012**.

[34] N. Mansouri, "An Effective Weighted Data Replication Strategy for Data Grid", in *Australian Journal of Basic and Applied Sciences*, vol. 6, no. 10, pp. 336–346, **2012**.

[35] W. H. Bell, D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, K. Stockinger, and F. Zini, "Evaluation of an Economy- Based File Replication Strategy for a Data Grid", in *Proc. Of 3rd IEEE Int. Symposium on Cluster Computing and the Grid (CCGrid)*, Tokyo, Japan, IEEE CS-Press, **2003**.

[36] A. Horri, R. Sepahvand, and G. Dastghaibyfard, "A hierarchical scheduling and replication strategy", International Journal of Computer Science and Network Security- IJCSNS, vol. 8, no. 8, pp. 30–35, **2008**.

[37] D. G. Cameron, "OptorSim v2.1 Installation and User Guide", *User Guide*, University of Glasgow, Scotland, **2006**.

AUTHORS PROFILE

Sheida Dayyani was born in Isfahan, Iran 1989. She received the B.Sc. degree in Computer Software Engineering from Sheikh Bahaei University (SHBU), Esfahan, Iran in 2011. She received his M.Sc. in Software Engineering from the same university, Isfahan, Iran in 2013. Her research interest are Grid and Cloud Computing, Scheduling algorithms, Health Information systems and Medical Informatics.

Dr. Mohammad Reza Khayyambashi was born in Isfahan, Iran in 1961. He received the B.Sc. degree in Computer Hardware Engineering from Tehran University, Tehran, Iran in 1987. He received his M.Sc. in Computer Architecture from Sharif University of Technology (SUT), Tehran, Iran in 1990. He got his Ph.D. in Computer Engineering, Distributed Systems from University of Newcastle upon Tyne, Newcastle upon Tyne, England in 2006. He is now working as a lecturer at the Department of Computer, Faculty of Engineering, University of Isfahan, Isfahan, Iran. His research interests include Distributed Systems, Networking, Fault Tolerance and E-Commerce. He has published in these areas extensively.