Journal of Advanced Computer Science & Technology, 4 (1) (2015) 163-174
www.sciencepubco.com/index.php/JACST

doi: 10.14419/jacst.v4i1.4009
Research Paper

A four-phase data replication algorithm for data grid

Alireza Salah [1], Reza Javidan *[2], Mohammad Taghi FatehiKhajeh [3]

[1,3]Department of Computer Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran
[2]Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran
**Corresponding author E-mail: Reza.Javidan@Gmail.Com*

Abstract

Nowadays, scientific applications generate a huge amount of data in terabytes or petabytes. Data grids currently proposed solutions to large scale data management problems including efficient file transfer and replication. Data is typically replicated in a Data Grid to improve the job response time and data availability. A reasonable number and right locations for replicas has become a challenge in the Data Grid. In this paper, a four-phase dynamic data replication algorithm based on Temporal and Geographical locality is proposed. It includes: 1) evaluating and identifying the popular data and triggering a replication operation when the popularity data passes a dynamic threshold; 2) analyzing and modeling the relationship between system availability and the number of replicas, and calculating a suitable number of new replicas; 3) evaluating and identifying the popular data in each site, and placing replicas among them; 4) removing files with least cost of average access time when encountering insufficient space for replication. The algorithm was tested using a grid simulator, OptorSim developed by European Data Grid Projects. The simulation results show that the proposed algorithm has better performance in comparison with other algorithms in terms of job execution time, effective network usage and percentage of storage filled.

Keywords: *Data Grid; Data replication; Geographical Locality; Replica Placement; Temporal Locality.*

1. Introduction

Today, the management of the huge distributed and shared data resources efficiently around the wide area networks becomes a significant topic for both scientific research and commercial application. As a specialization and extension of the Grid [1], the Data Grid is a solution for this problem [2]. A data grid connects a collection of hundreds of geographically distributed computers and storage resources located in different parts of the world to facilitate sharing of data and resources [3], [4]. Grids can be classified into computational grids and data grids [5]. Computational grids are developed for managing computational intensive tasks, and data grids are developed for data sharing and collaboration, which ‘‘involves the complete dynamic life cycle of service deployment, provisioning, management and decommissioning’’.

In order to increase performance of data grids, jobs must be executed as fast as possible. Data intensive application runs in Data Grid and need a huge amount of data that its size reaches from terabytes to petabytes. Managing such huge amounts of data in a centralized manner is almost impossible due to extensively increased data access time. Data replication is a key technique to manage large data in a distributed manner; that is, to create copies of a replica to get faster access to it. Creating replicas can reroute the client requests to certain replica sites and offer remarkably higher access speed than a single server. At the same time, the workload of the original server is distributed to replica servers and decreases significantly. 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 [6] [7] [8]. Data replication is a common technique to manage large data in distributed environment such as data grid. It can be answered to the requirements of many grid applications Thereby, to increase data reliability and availability identical copies of a data file are replicated and dispatched to the diverse grid sites [9]. Generally, the benefits of using replication are to reduce access latency and bandwidth consumption. Replication can

also improve data availability and load balancing. Earlier research on data replication [10], [11] focused on decreasing the data access latency and the network bandwidth assumption. As bandwidth and computing capacity have become relatively cheaper, the data access latency can drop dramatically, and how to improve the data availability and system reliability becomes the new focus. The most important questions any replication strategy has to answer are: Which files should be replicated? When should the replicas be created? How many replicas should be created? Where the replicas should be placed? Which replica should be deleted if there is no enough space in data storage? Depending on the answers, various different replication strategies are born. In a classification, Replication methods are classified into static and dynamic [2]. Replication methods can be classified as static and dynamic. For the static replication, after a replica is created, it will exist in the same place till it is deleted manually by users or its duration is expired. The drawback of static replication is evident, when client access patterns change greatly in the Data Grid, the benefits brought by replica will decrease sharply. On the contrary, dynamic replication takes into consideration the changes of the Grid environments and automatically creates new replicas for popular data files or moves the replicas to other sites when necessary to improve the performance. Data grids are a dynamic environment so dynamic replication is more suitable for these environment [8], [12].

In this Paper we have presented a Four-Phase Data Replication Algorithm named 4PDRA, based on Temporal and Geographical locality is proposed. Temporal locality means files accessed recently are likely to be accessed again and Geographical locality means files recently accessed by a client are likely to be accessed by nearby clients [13]. With the fact of temporal locality, a popular data is determined by analyzing the users' access to the data. When the popularity of the data passes a dynamic threshold, the replication operation will be triggered. The number of replicas will be determined based on the system availability and failure probability. New replica will be created among data nodes in a balanced way based on temporal geography. At last if there is no enough space for new replica, the data file with least cost of average access time will be deleted. Simulation results show that the proposed method decrease job execution time and network bandwidth use.

The rest of the paper is organized as follows, in Section 2 a summary of existing and related works are presented. In Section 3, some definitions and assumptions are defined. In Section 4, A Four-Phase Data Replication Algorithm is proposed. Section 5 describes the experiments and the results achieved followed by conclusion in Section 6.

2. Related works

Several recent studies have taken into account data replication algorithm in the Data Grid. In [14], Ranganathan et al. simulated the six replica strategies (No Replica, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread) for the three user access patterns (random access, small temporal locality, and small geographical and temporal locality). The simulation results show that the best strategy has significant savings in latency and bandwidth consumption if the access patterns contain a moderate amount of geographical locality.

In [2], Khanli et al. proposed PHFS (predictive hierarchal fast spread), which is a replication technique designed to decrease the access latency of data access. This is an extension of fast spread presented by Ranganathan et al. [14]. PHFS uses predictive techniques to predict the future usage of files and then pre-replicates them in hierarchal data grid on a path from source to client. It works in two phases, in phase one it makes the file access log files by collecting the access information from all over the system. In the next phase it applies data mining techniques like clustering and association rule mining to find useful knowledge like clusters of files that are accessed together or most frequent sequential access patterns. In this way PHFS finds the relationship between the files for future predictions. In this way PHFS tries to increase the locality in access by predicting the user's succeeding file demands and pre-replicates them in the hierarchal method in advance and achieves higher availability with optimized usage of storage resources.

In [8] Park et al. proposed an algorithm called Bandwidth Hierarchy Replication (BHR), which reduces data access time by avoiding network congestions in a data grid network. With the BHR strategy, one can take advantage of 'network level locality', which means that the required file is located in a site that has a large amount of bandwidth between it and the job execution site. In a data grid, some sites may be located within a region where sites are linked closely. For instance, a country can be referred to as this network region. Network bandwidth between sites within a region will be broader than those sites across regions. If the required file is located in the same region, less time will be consumed fetching the file. The BHR strategy reduces data access time by maximizing network level locality. In another paper [15], the authors proposed the BHR algorithm by using three level hierarchical structures. They addressed the problems of both scheduling and replication.

In [16] Sashi et al. have presented a modified form of Bandwidth Hierarchy Replication (BHR) to overcome its limitations. In the modified BHR model a network region is defined as a network topological space where sites are located closely. Whenever the required replica is present in the same region, the job completion will be fast. The storage locations for popular files are determined by considering the temporal and geographical localities. If the required file is not present locally then the Replica Optimizer algorithm looks for it in the nearby sites of same region and proceeds to execute the job. Then it sorts files in all storage elements (SE) in Most Frequently Accessed order to find the SE which accesses the file for most of the time. If the selected SE has enough space to hold this file, the file is replicated to it. Otherwise it looks for a duplication of this replica in other sites within the same region; if such duplications are present the replica optimizer will be terminated. If no duplications are present the replica optimizer will find the LRU file and

check whether this file is duplicated on any other site in the same region or not. If it is present within same region and its access frequency is less than the access frequency of new replica, then it is deleted from the selected SE to make room for the replication of the new file. In this way the Modified BHR replicates the file within the region with the condition that the replica is present in the site where it is accessed for most of the time. In this paper OptorSim simulator is used for evolution the purposed algorithm. Result shows that this algorithm enhances the performance by minimizing the data access time and avoiding unnecessary replication.
Another Dynamic Replication Algorithm (DRA) [17] was proposed by Sashi et al. for European Data Grid. It considers a network topology in which different clusters are present. Within a cluster the sites are located closely. DRA improves the availability of a file by replicating it within a cluster. The data is initially produced in cluster master and it is then distributed to all cluster heads. Access frequency of all the files is determined and most popular files are replicated to the site where it is requested for most of the times, considering the geographical and temporal localities. The matrices used for evaluation of performance of DRA are Mean Job Execution time (MJET) and Average Storage Used. During simulation they have compared DRA with No Replication, LRU and LFU and it is observed that performance of DRA is better.

3. Problem definition

In this section, a system model, a series of availability definitions and a mathematical analysis to describe the relationship between system availability and the number of replicas are presented in detail.

3.1. Problem assumption

Let be a file set composed of Fn files, $R = \{r_1, r_2, \dots, r_{Rn}\}$ be a replica set composed Rn replicas, and $R_i = \left\{r_{i_1}, r_{i_2}, \dots, r_{i_{Rn_i}}\right\}$ be a sub-set of replicas of the i-th file f_i, where Rn_i is the number of replica of f_i, so the number of replica of whole system can be calculate as (1).

$$Rn = \sum_{i=1}^{Fn} Rn_i \tag{1}$$

Let $DC = \{dc_1, dc_2, \dots, dc_{DCn}\}$ be a data center set of composed DCn data centers, $U = \{u_1, u_2, \dots, u_{Un}\}$ be a user set composed Un users, $J = \{j_1, j_2, \dots, j_{Jn}\}$ be a job set composed Jn jobs, and $F_{j_k} = \left\{f_{j_{k_1}}, f_{j_{k_2}}, \dots, f_{j_{kFn_k}}\right\}$ be a sub-set of files of the k-th job j_k, where Fn_k is the number of sub-files, and $f_{j_{kl}}$ is the l-th requested file by job j_k and independent of the other jobs. For simplicity, we assume that the jobs are non-preemptable and non-interruptible which mean that a job cannot be broken into smaller sub-jobs and it has to be executed as a whole on a single processor with given resources. In addition, as soon as a job starts its execution on a processor, it cannot be interrupted and it occupies the processor until its execution completes successfully or a failure occurs.

3.2. Availability

One of the most important objectives of distributed system is to provide the highest availability by placing all replicas of data files in a load balanced way on different of data centers, which is similar to that for grid environments [18].

Definition 3.2.1: *(Availability): It is the ability of a system to limit, control, and provide proper service un- der given constraints, defined as the "readiness for correct service" of a system [19]. The lifetime of a grid can be divided into a set of "up states" and a set of "down states".*

Definition 3.2.2: *(Replica Availability): Replica availability is the ability of a data replica to limit, control, and provide proper service under given constraints. The replica availability of a replica r_k is denoted as RA_k . $P(RA_k)$ is the probability of replica r_k in an available state. $P(\overline{RA_k})$ is the probability of replica r_k in an unavailable state, and $P(\overline{RA_k}) = 1 - P(RA_k)$.*

Definition 3.2.3: *(File Availability): File availability is the ability of a data replica to limit, control, and provide proper service under given constraints. The file availability of a replica f_i is denoted as FA_i. $P(FA_i)$ is the probability of replica f_i in an available state. $P(\overline{FA_i})$ is the probability of replica f_i in an unavailable state, and $P(\overline{FA_i}) = 1 - P(FA_i)$.*
The number of replicas of file f_i is Rn_i. It is obvious that file f_i is considered unavailable only if all the replicas of file f_i are not available. So the availability and unavailability of file f_i is shown by (2) and (3):

$$P(FA_i) = 1 - \left(1 - P(RA_i)\right)^{Rn_i} \tag{2}$$

$$P(\overline{FA_i}) = \left(1 - P(RA_i)\right)^{Rn_i} \quad (3)$$

Proof: The available and unavailable probability of each replica of file r_k are $P(RA_k)$ and $P(\overline{RA_k})$, and the available and unavailable probability of file f_i are $P(FA_i)$ and $P(\overline{FA_i})$. As there are Rn_i replicas of file f_i , file f_i is unavailable if and only if all the Rn_i replicas of file f_i are unavailable. Therefore,

$$P(\overline{FA_i}) = P\left(\overline{RA_{i_1}}, \overline{RA_{i_2}}, \dots, \overline{RA_{i_{Rn_i}}}\right)$$

All the Rn_i replicas are distributed in different data centers, and all the Rn_i replicas are independent of each other, thus,

$$P(\overline{FA_i}) = P\left(\overline{RA_{i_1}}\right) \times P\left(\overline{RA_{i_2}}\right) \times \dots \times P\left(\overline{RA_{i_{Rn_i}}}\right) = \prod_{k=1}^{Rn_i} P\left(\overline{RA_{i_k}}\right)$$

Then

$$P(\overline{FA_i}) = \prod_{k=1}^{Rn_i} P\left(\overline{RA_{i_k}}\right) = \prod_{k=1}^{Rn_i} \left(1 - P\left(RA_{i_k}\right)\right) = \prod_{k=1}^{Rn_i} \left(1 - P(RA_i)\right) = \left(1 - P(RA_i)\right)^{Rn_i}$$

We obtain

$$P(FA_i) = 1 - \left(1 - P(RA_i)\right)^{Rn_i}$$

4. Four-phase data replication algorithm

The four-phase data replication algorithm 4PDRA (Four-Phase Data Replication Algorithm) has four important phases:

1) Which data file should be replicated and when to replicate in the data grid to meet users' requirements such as mean job time reduction data access speeding up.
2) How many suitable new replicas should be created in the data grid to meet a given availability requirement.
3) Where the new replicas should be placed to meet the bandwidth consumption requirements.
4) Which replica should be deleted if there is no enough space in data storage?

4.1. Decide which and when to replicate

Given the fact that a more recently accessed data file might be accessed again in the near future according to the current status of data access pattern, which is called temporal locality, a popular data file is determined by analyzing the access to the data from users. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered.

Definition 4.1.1: *(Time-Based Forgetting Function): A time-based forgetting function ω is defined over the domain Time, with values within the interval* [0,1] *[20]. It is used to calculate the popularity degree PD_{f_i} of a file f_i at the present time t_p according to the access frequency at the start time t_s, as shown by (4),*

$$\omega(t_p, t_s) = a^{-(\Delta t)^k} = a^{-(t_p - t_s)^k}, a > 1, k \in \{1,2, \dots\} \quad (4)$$

Where $\Delta t = t_p - t_s$, as usual, parameter a is assigned as e, as shown by (5)

$$\omega(t_p, t_s) = e^{-(\Delta t)^k} = e^{-(t_p - t_s)^k}, k \in \{1,2, \dots\} \quad (5)$$

The value of k determines the rate of decay of the popularity degree with time Δt, and is assigned by the file f_i based on its perception about the change. Fig. 1 shows the nature of the change of $\omega(t_p, t_s)$ with deferent values of k. If $\Delta t = 0$, then $\omega(t_p, t_s) = e^{-0} = 1$. If $\Delta t \to +\infty$, then $\omega(t_p, t_s) = \lim_{\Delta t \to +\infty} e^{-(\Delta t)^k} = 0$. This corroborates the fact that the time-based forgetting weight is asymptotic to zero at infinite time.

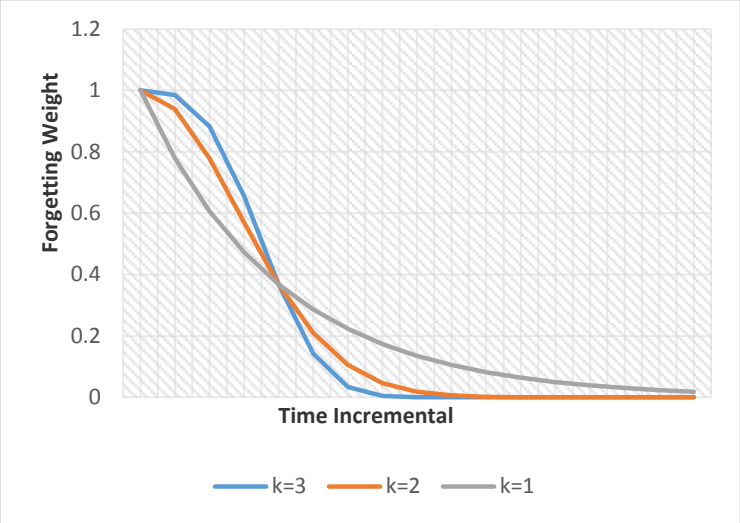

Fig. 1: Time-Based Forgetting Function [20].

Definition 4.1.2: *(Popularity Degree): The popularity degree of a file f_i is defined as the access frequency based on time factor. During the period from the start time t_s to the present time t_p, the popularity degree PD_i of a file f_i can be calculated by (6).*

$$\mathrm{PD_i} = \sum_{t_k=t_s}^{t_p} \left(\mathrm{An_i}(t_k, t_{k+1}) \times \omega(t_k, t_p)\right) \tag{6}$$

Where $An_i(t_k, t_{k+1})$ is the number of accesses during the time interval t_k to t_{k+1}.

Definition 4.1.3: *(Replica Factor): The replica factor is defined as the ratio of the popularity degree and the total number of bytes of data file f_i requested by all tasks under given constraints. It is used to determine whether the data file f_i should be replicated, denoted as RF_i in (7).*

$$RF_i = \frac{PD_i}{Rn_i \times Fs_i} \tag{7}$$

Where PD_i, RN_i, FS_i are the popularity degree, number of replicas and file size of data file f_i in million bytes, respectively. According to (6) RF_i can be calculated by (8).

$$RF_i = \frac{\sum_{t_k=t_s}^{t_p}\left(An_i(t_k,t_{k+1}) \times \omega(t_k,t_p)\right)}{Rn_i \times Fs_i} \tag{8}$$

According to replica factor Definition, we can prove that the system replica factor RF_{sys} can be calculated by (9).

$$RF_{sys} = \frac{\sum_{i=1}^{Fn}\left(\sum_{t_k=t_s}^{t_p}\left(An_i(t_k,t_{k+1}) \times \omega(t_k,t_p)\right)\right)}{\sum_{i=1}^{Fn}(Rn_i \times Fs_i)} \tag{9}$$

In each time interval T, the replication operation of the data file f_i will be triggered if the condition shown in (10) is met.

$$RF_i > \left((1+\alpha) \times RF_{sys}\right), \alpha \in [0,1] \tag{10}$$

Where $\boldsymbol{\alpha}$ is the adjustable parameter according to different system performance. The better the requested system performance, the greater $\boldsymbol{\alpha}$ can be selected. If $\boldsymbol{\alpha}$ is near to zero more files are chosen for replication, and if $\boldsymbol{\alpha}$ is near to 1 less files are chosen for replication.

4.2. Determine the number of new replicas

To meet the data availability requirement, new replicas should be created. With a reasonable increase of file availability, the number of new replicas that need to be created can be calculated according to (11), which determines

the new replicas on the basis of old file availability $P_{old}(FA_i)$ of data file f_i and the replica factor based adjustable parameter β.

$$P_{new}(FA_i) = P_{old}(FA_i) + \beta \times \left(1 - P_{old}(FA_i)\right), \beta \in [0,1] \tag{11}$$

Where $P_{new}(FA_i)$ and $P_{old}(FA_i)$ are the new file availability and the old file availability of data file f_i, respectively. β is the replica factor based adjustable parameter.It can be calculated according to (12).

$$\beta = \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \tag{12}$$

Where $Fn(slc)$ is the number of files selected to be replicated. So the number of new replicas that need to be created can be calculated according to (13).

$$Rn_i(inc) = \left\lfloor \frac{ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \times \left(1 - P_{old}(FA_i)\right)\right)\right)}{ln\left(1 - P(RA_i)\right)} - Rn_i(old) \right\rfloor \tag{13}$$

Where $Rn_i(old)$ and $Rn_i(inc)$ are the old replica number of data file f_i and the number of new replicas to be created respectively.

Proof: As $P_{new}(FA_i)$ and $P_{old}(FA_i)$ are the new file availability and old file availability of data file f_i, respectively, and $P_{new}(FA_i) = 1 - \left(1 - P(RA_i)\right)^{Rn_i(new)}$, according to (11) and (12), we obtain.

$$1 - \left(1 - P(RA_i)\right)^{Rn_i(new)} = P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \times \left(1 - P_{old}(FA_i)\right)$$

So

$$\left(1 - P(RA_i)\right)^{Rn_i(new)} = 1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \times \left(1 - P_{old}(FA_i)\right)\right)$$

And

$$ln\left(1 - P(RA_i)\right)^{Rn_i(new)} = ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \times \left(1 - P_{old}(FA_i)\right)\right)\right)$$

Then

$$Rn_i(new) \times ln\left(1 - P(RA_i)\right) = ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \times \left(1 - P_{old}(FA_i)\right)\right)\right)$$

We obtain

$$Rn_i(new) = \left\lfloor \frac{ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k} \times \left(1 - P_{old}(FA_i)\right)\right)\right)}{ln\left(1 - P(RA_i)\right)} \right\rfloor$$

If the old number of replicas is $Rn_i(old)$, the number of new replicas $Rn_i(inc)$ to be created is,

$$Rn_{\mathrm{i}}(inc) = \left\lfloor \frac{ln\left(1-\left(P_{old}(FA_i)+\frac{RF_i}{\sum_{k=1}^{Fn(slc)} RF_k}\times\left(1-P_{old}(FA_i)\right)\right)\right)}{ln\left(1-P(RA_i)\right)} - Rn_i(old) \right\rfloor$$

4.3. Placement of new replicas

To meet the system task successful execution rate and bandwidth consumption requirement, deferent data centers which have the selected replica data file f will decide the replica placement and the placement of new replicas to be created according to the access information of f_i data centers. As mentioned An_i is the number of accesses to file f_i so An_{i_j} is the number of accesses to file f_i from j-th data center dc_j that should meet (14).

$$An_i = \sum_{j=1}^{DCn} An_{i_j} \tag{14}$$

Definition 4.3.1: *(Popularity Degree per data center): The popularity degree of a file f_i from data center dc_j is defined as the access frequency in data center based on time factor. During the period from the start time t_s to the present time t_p, the popularity degree PD_{i_j} of a file f_i from data center dc_j can be calculated by (15).*

$$PD_{i_j} = \sum_{t_k=t_s}^{t_p}\left(An_{i_j}(t_k, t_{k+1}) \times \omega(t_k, t_p)\right) \tag{15}$$

Where $An_{i_j}(t_k, t_{k+1})$ is the number of accesses of file f_i from data center dc_j during the time interval t_k to t_{k+1}. This parameter used to select data centers set that need replicas to be created in them, and data centers which have the most Popularity Degree per data center will be chosen. So $DC_{need_i} = \left\{dc_{need_{i_1}}, dc_{need_{i_2}}, \dots, dc_{need_{i_{Rn_i(inc)}}}\right\}$ is the data center set that new replica should be created on them and it can calculated by (16).

$$DC_{need_i}=\left\{dc_{need_{i_1}}, dc_{need_{i_2}}, \dots, dc_{need_{\mathrm{i}_{Rn_i(inc)}}}\right\}, \quad DC_{need_i} \subset DC, \forall\, dc_{need_{i_j}} \in DC_{need_i}, dc_j \in DC - DC_{need_i};\ PD_{i_{need_{i_j}}} > PD_{i_j} \tag{16}$$

4.4. Deletion old replica

After selecting the appropriate data centers for the placement of the new replicas, if the replica is not available in the data center and it has enough space for placement, the replica on data center can be created. Otherwise, some files should be deleted from the data center to be replicated if possible. To do this, some definitions are presented and based on that how to select the file for deletion will be described.

Definition 4.4.1: (*The best data center contains a file): is the data center which has the required data for replication, and has the most bandwidth between itself and the destination data center to other data centers contain files.*
If dc_j is the one of the selected data center to replicate the new replica of file f_i, let $DC_{containing_i} = \left\{dc_{containing_{i_1}}, dc_{containing_{i_2}}, \dots, dc_{containing_{i_{Rn_i}}}\right\}$ be set of data center that containing the data file f_i composed Rn_i data center, and $BW_{j_{containing_i}} = \left\{bw_{jcontaining_{i_1}}, bw_{jcontaining_{i_2}}, \dots, bw_{jcontaining_{i_{Rn_i}}}\right\}$ be a set of bandwidth between data center dc_j and data center of set $DC_{containing_i}$, best data center containing file can be calculated by (17).

$$dc_{Best_i} = dc_{containing_{i_k}}, dc_{containing_{i_k}} \in DC_{containing_i}, bw_{jcontaining_{i_k}} = max\left(BW_{j_{containing_i}}\right) \tag{17}$$

Definition 4.4.2: *(Access cost): access cost of data center dc_j to file f_i is size of file f_i to bandwidth between data center dc_j and the best data center dc_{Best_i} contains file f_i. It can be calculated according to (18).*

$$Cost_{jBest_i} = \frac{Fs_i}{bw_{jBest_i}} \tag{18}$$

Where $Cost_{jBest_i}$ is access cost of data center dc_j to file f_i, and bandwidth between data center dc_j and the best data center dc_{Best_i} contains file f_i.

According to access cost definition, we can prove that the average cost of file f_i can be calculated by (19).

$$AvgCost_i = \frac{\sum_{j=0}^{Rn_i(inc)} Cost_{jBest_i}}{Rn_i(inc)} \tag{19}$$

During the placement of replica, if the data center doesn't have enough space to store new replicas, a mechanism should be considered to delete existing files. Deletion takes place in two steps. In the first step, the existing files are arranged based on the least frequently used (LFU) and then as much as Five times bigger than the size of the selected files, some files from data center are chosen as a candidate and introduced to the second step. Let $F_{dc_j} = \left\{ f_{dc_{j_1}}, f_{dc_{j_2}}, \dots, f_{dc_{j_{Fn_j}}} \right\}$ be files set of data center dc_j Candidate files for deletion can be calculated according to (20).

$$F_{candid_{dc_j}} = \left\{ f_{candid_{dc_1}}, f_{candid_{dc_2}}, \dots, f_{candid_{dc_{Fn_j}}} \right\}, \\ F_{candid_{dc_j}} \subset F_{dc_j}, \forall f_{candid_{dc_k}} \in F_{candid_{dc_j}}, f_{dc_{j_k}} \in F_{dc_j} - F_{candid_{dc_j}}, An_{candid_{dc_k}} > An_{dc_{j_k}} \tag{20}$$

In the second step, the files of set $F_{candid_{dc_j}}$ that have the following two conditions are deleted:

1) The average of access time to the selected file for deletion is less than the average of access time to every single existing file in its data center.
2) The average of access time to the selected file for deletion assuming it is deleted is less than the average of access time to the file which is selected for replication.

And this operation continues until enough space for replication of new replicas is created. And if it fails to create enough space, placement doesn't take place. The file $f_{candid_{dc_k}}$ will be deleted if the condition shown in (21) is met.

$$\forall f_{dc_{j_k}} \in F_{dc_j}, AvgCost_{candid_{dc_k}} \leq AvgCost_{dc_{j_k}}, AvgCost_{candid_{dc_k}} < AvgCost_i, if\ f_{candid_{dc_k}}\ deleted \tag{21}$$

4.5. PDRA algorithm

According to the above analysis, the replication decision is based on the theory of temporal locality and temporal geography. A popular data file is determined by the analysis of the access information to the data from users. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. The number of replicas depends on the reasonable increase of file availability; the replica placement is determined by the popularity of a data file per data center and is accomplished in a balanced way, and the deletion of file is based on access cost of data file. The core part of the 4PDRA algorithm is described as follows.

1) 4PDRA Algorithm
2) Begin
3) for each data file f_i in all data centers DC do
4) Calculate the popularity degree PD_i of data file f_i by (6).
5) Calculate replica factor RF_i of data file f_i by (8).
6) end for
7) Calculate replica factor RF_{sys} of system by (9).
8) for each data file f_i at all data centers DC do
9) if condition of (10) is met then
10) The replication operation of the data file f_i will be triggered.
11) end if
12) end for
13) for each data file f_i at all data centers DC do
14) Calculate the old file availability $P_{old}(FA_i)$ of data file f_i by (2).
15) end for
16) for each triggered data file f_i do
17) Calculate the new file availability $P_{new}(FA_i)$of data file f_i by (11).
18) Calculate the number of new replicas needed R$n_i(inc)$ by (13).
19) end for
20) for each triggered data file f_i do
21) for each data center dc_j do
22) Calculate popularity degree per data center PD_{i_j} for file f_i in data center dc_j by (15).
23) Select data center set DC_{need_i} for replica placement of file f_i by (16)

24) end for
25) for each data center dc_{need_i} do
26) Replicate replicas of file f_i in data center dc_{need_i}
27) if the storage space of the target data center dc_{need_i} is not enough then
28) Find the candidate files set $F_{candid_{dc_j}}$ to deletion in data center dc_{need_i} by (20).
29) do
30) Delete data file from data center dc_{need_i} according to (21).
31) while enough space for replication provide
32) end if
33) end for
34) End

5. Experimental results

In order to evaluate the performance of the proposed 4PDRA algorithm, simulation environment and parameter setup are discussed in this section, followed by the precise performance evaluation results.

5.1. Simulation tool and parameter setup

OptorSim is used as the simulator tool to evaluate the performance of our proposed algorithm. OptorSim was developed by the European Data Grid projects [21] and is written in Java. It provides a framework to simulate the real grid environment. It is developed to test the dynamic replication strategies.
The simulation here has been performed with the grid network topology shown in Fig. 2 we used the CMS [22] testbed architecture; this architecture contains twenty sites. The job execution scenario used for this algorithm is shown in Table 1. 4PDRA is compared with No-Replication, Least Frequently Used (LFU), and Least Recently Used (LRU). In No Replication Algorithm the files are taken from the master site where the data are originally produced. LFU always replicates and delete those files least frequently accessed if the storage space is not enough for storing new replica. LRU strategy also always replicates and delete those files least recently accessed.

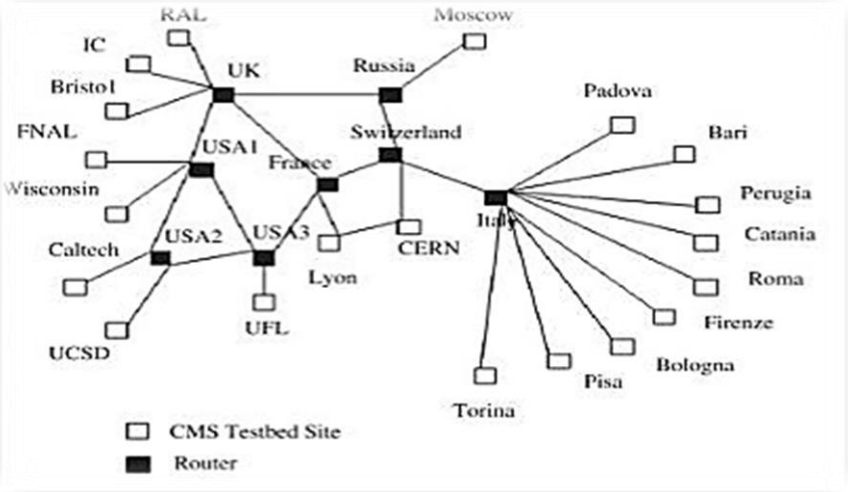

Fig. 2: Network Topology.

Table 1: Simulation Parameters

Parameters	Values
Number of Jobs	1000
Number of Job Types	6
Size of single file	100Mb-1GB
Job Delay	2500ms
Bandwidth	100-10000MB
Volume of storage element	50000MB
α	0.5

5.2. Evaluation metrics

Definition 5.2.1: *(Mean job execution time): The mean job execution time is defined as the total time to execute all the jobs divided by the number of jobs completed. The total time includes the time that elapses from when a job enters the queue in a site to await execution until the time when the job finishes its processing and leaves the site. It is computed as shown in (22).*

$$MJET = \frac{\sum_{i=1}^{Jn}(Jat_i - Jdt_i)}{Jn_{completed}} \quad (22)$$

Where Jn is the number of jobs processed through the system, Jat_i is arrival time of job J_i, Jdt_i is departure time of job J_i, and $Jn_{completed}$ is number of completed job. This metric is considered the most important of the evaluation metrics.

Definition 5.2.2: *(Average storage used): Storage used specifies the amount of space used by files. Monitoring the use of storage resources in the grid sites can also be a valuable source of information. Storage usage can be calculated for each site as a percentage of the capacity reserved by files out of the total capacity of the underlying storage. The average of all the storage elements in the grid can reflect the total system storage cost, which is computed as shown in (23).*

$$ASU = \frac{\sum_{i=1}^{DCn}\left(\frac{DCu_i}{DCs_i}\right)}{DCn} \times 100\% \quad (23)$$

Where DCn is the number of data centers, DCu_i is the storage usage for data center dc_i, and DCs_i is the total capacity of the storage medium.

Definition 5.2.3: *(Effective network usage): Replicating a file takes time and uses network bandwidth. This cost is effectively the ratio of files transferred to files requested, so a low value indicates that the optimization strategy used is better at placing files in the right places. It can be measured by using (24).*

$$\mathrm{ENU} = \frac{\mathrm{An_{remote}} + \mathrm{Fn_{replicate}}}{\mathrm{An_{remote}} + \mathrm{An_{local}}} \quad (24)$$

Where $\mathrm{An_{remote}}$ is the number of times the Computing Element reads a file from a Storage Element on a different data center, $\mathrm{Fn_{replicate}}$ is the total number of file replications that take place during the job execution and $\mathrm{An_{local}}$ is the number of times a Computing Element reads a file from a Storage Element on the same data center?

5.3. Simulation results and discussion

In this section the result of 4PDRA simulation will be described.
The mean job execution time for the Random Zipf Access Pattern Generator is shown in Fig. 3 when compared to all other algorithms, the Modified 4PDRA algorithm has the shortest mean job execution time. The 4PDRA algorithm improves the mean job execution time by locating files based on temporal locality and temporal geography, and stores it in the most frequently accessed data center.

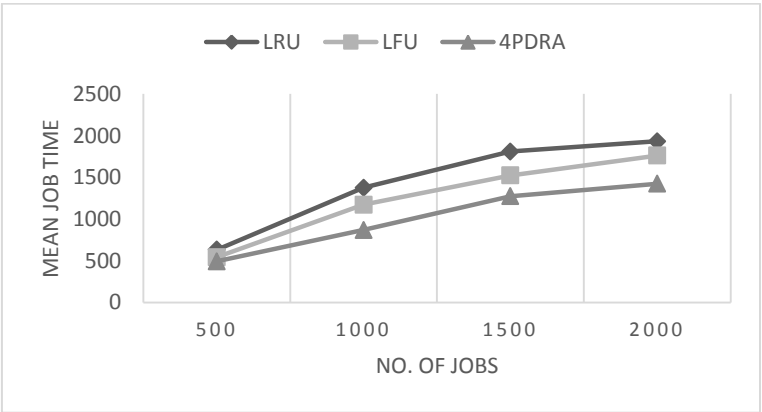

Fig.3: The Mean Execution Time for Different Algorithms.

The storage used for the Random Zipf Access Pattern Generator is depicted in Fig. 4. The storage used is best in the No replication strategy because the file is stored in only one site, which is the master site, and no replication takes place.

When compared to the LRU, the LFU and 4PDRA algorithms, the 4PDRA algorithm performs better because of considering temporal locality and creating a reasonable number of replicas.

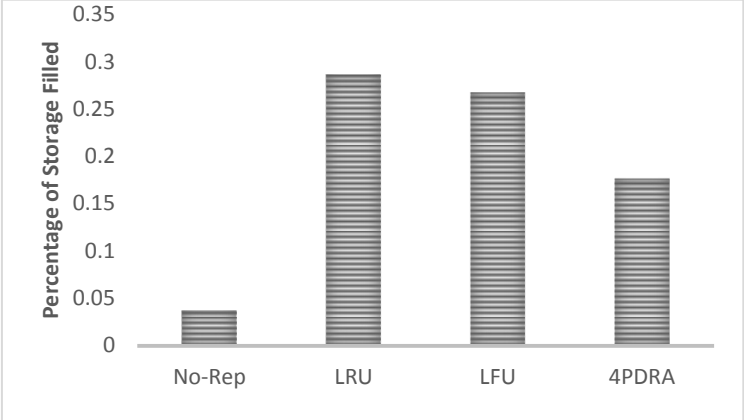

Fig.4: The Percentage of the Used Storage Space.

The effective network usage ranges from 0 to 1. This algorithm is optimized to minimize the bandwidth consumption and thus reduce the network traffic. The No Replication Strategy performs the worst and consumes the maximum network bandwidth available in the network. The effective network usage is better in 4PDRA in comparison the LFU and LRU strategies because of considering access cost in deletion of replicas. The effective network usage for the Random Zipf Access Pattern Generator is shown in Fig. 5.

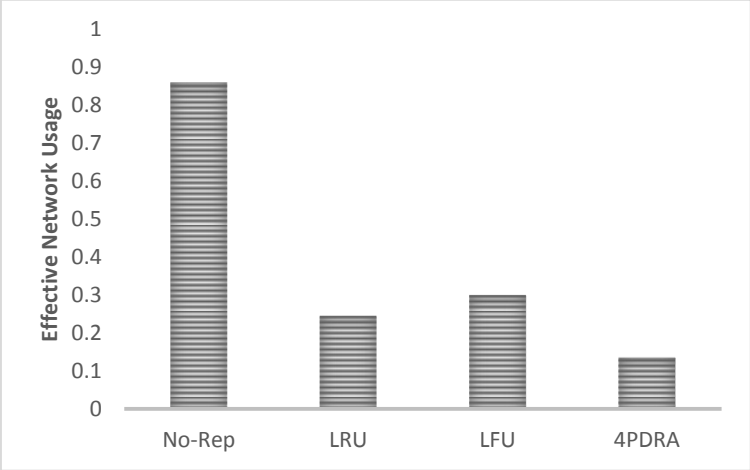

Fig. 5: Effective Network Usage.

6. Conclusion and future works

A data grid provides the ability to access and manage data and data resources on the grid. Replication involves the creation of identical copies of data files and their distribution over various grid sites needing access to these replicas. Replicas must be managed in terms of creation, deletion and placement. In this paper, a Four-Phase Data Replication Algorithm based on Temporal and Geographical locality is proposed. It includes: 1. Evaluating and identifying the popular data and triggering a replication operation when the popularity data passes a dynamic threshold; 2. Analyzing and modeling the relationship between system availability and the number of replicas, and calculating a suitable number of new replicas; 3. Placing replicas among data nodes in a balanced way; 4. It removes files with least cost of average access time when encountering insufficient space for replication. By using this algorithm, mean job execution time can be minimized. The network is used more effectively, and storage space is saved. In future work, more realistic scenarios and user access patterns can be investigated. It can be used another function to calculate the popularity degree such as times series or considering user's access pattern, instead of time-based forgetting function. It can be used some other metric to evaluate this algorithm such as response time, waiting time and etc. Finally, this model can be deployed in a real grid environment.

References

[1] I. Foster, C. Kesselman. "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, (2004).

[2] Khanli L.M., A. Isazadeh, T.N. Shishavanc. "PHFS: A dynamic Replication method, to decrease access latencyin multi-tier data grid", Future GenerationComputer Systems 27, (2011), pp.233-244. http://dx.doi.org/10.1016/j.future.2010.08.013.

[3] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D.Quesnel, S. Tuecke. "Secure, Efficient Data Transportand Replica Management for High-Performance Data-Intensive Computing" IEEE Mass Storage Conference, (2001).

[4] Stevens. R, Woodward. P, DeFanti. T, Catlett. C. "From the I-WAY to the National Technology Grid", Communications of the ACM, Vol.40, No.11, (1997), pp.50-61. http://dx.doi.org/10.1145/265684.265692.

[5] Jose M. Perez, F. Garcia-Carballeira, J. Carretero, A. Calderon, J. Fernandez. "Branch replication scheme: a new model for data replication in large scale data grids", Future Generation Computer Systems, Vol.26, No.1, (2010), pp.12–20. http://dx.doi.org/10.1016/j.future.2009.05.015.

[6] Chang R-S., J-S. Chang, S-Y. Lin. "Job scheduling and data replication on data grids", Future Generation Computer Systems, Vol.23, (2007), pp.846-860. http://dx.doi.org/10.1016/j.future.2007.02.008.

[7] Rabinovich M., I. Rabinovich, R. Rajaraman. "Dynamic Replication on the Internet", Technical Report, (1998).

[8] Park S.M., J.H. Kim, Y.B. Ko, W.S. Yoon. "Dynamic Data Replication Strategy Based on Internet Hierarchy BHR", Lecture notes in Computer Science Publisher, Springer-Verlag, Vol.3033, (2004), pp.838-846.

[9] Tang M., B.S. Lee, X. Tang, C.K. Yeo. "The impact of data replication on job scheduling performance in the data grid", Future Generation Computer Systems, Vol.22, No.3, (2006), pp.254–268. http://dx.doi.org/10.1016/j.future.2005.08.004.

[10] William H. Bell, David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Kurt Stockinger, Floriano Zini. "Evaluation of an economy-based file replication strategy for a Data Grid", International Workshop on Agent Based Cluster and Grid Computing at CCGrid, Tokyo, Japan, IEEE Computer Society Press ,(2003). http://dx.doi.org/10.1109/CCGRID.2003.1199430.

[11] Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko, Won-Sik Yoon. "Dynamic Data Grid replication strategy based on internet hierarchy", Second International Workshop on Grid and CooperativeComputing, GCC', Shanghai, China, (2003).

[12] K. Sashi, A.S. Thanamani. "Dynamic replication in a data grid using a modified BHR region based algorithm", Future Generation Computer Systems, Vol.27, No.2, (2011), pp.202–210. http://dx.doi.org/10.1016/j.future.2010.08.011.

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

[14] A. Horri, R. Sepahvand, Gh. Dastghaibyfard. "A hierarchical scheduling and replication strategy", International Journal of Computer Science and Network Security Vol.8, (2008).

[15] K. Sashi, A.S. Thanamani. "Dynamic replication in a data grid using a modified BHR region based algorithm", Future Generation Computer Systems, Vol.27, No.2, (2011), pp.202–210. http://dx.doi.org/10.1016/j.future.2010.08.011.

[16] K. Sashi, A.S. Thanamani. "A new dynamic replication algorithm for European data grid", Proceedings of the Third Annual ACM Bangalore Conference, (2010), p.17. http://dx.doi.org/10.1145/1754288.1754305.

[17] Rood B, Lewis M J. "Grid resource availability prediction- based scheduling and task replication", Journal of Grid Computing, Vol.7, No.4, (2009), pp.479-500. http://dx.doi.org/10.1007/s10723-009-9135-2.

[18] Al-Kuwaiti M, Kyriakopoulos N, Hussein S. "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability", IEEE Communications Surveys & Tutorials, Vol.11, No.2, (2009), pp.106-124. http://dx.doi.org/10.1109/SURV.2009.090208.

[19] Sun DW, Chang GR, Gao S., "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments", Journal of Computer Science and Technology, Vol. 27, No.2, (2012), pp.256-272. http://dx.doi.org/10.1007/s11390-012-1221-4.

[20] D.G. Cameron, R.C. Schiaffino, J. Ferguson, P. Millar, C. Nicholson, K.Stockinger, F. Zini. "OptorSim v2.0 Installation and User Guide", (2004).

[21] K. Holtman. "CMS Data Grid system overview and requirement", Tech report CERN, (2001).