# An Adaptive Data Replication Algorithm based on Star-based Data Grids

**Ming-Chang Lee**
**Fang-Yie Leu**
**Ying-ping Chen**

NCLab Report No. NCL-TR-2010009
November 2010

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
http://nclab.tw/

# An Adaptive Data Replication Algorithm based on Star-based Data Grids

Ming-Chang Lee, Fang-Yie Leu, Ying-ping Chen

Abstract

In recent Grid research and development, data replication has been used to duplicate frequently accessed data from its current location to appropriate sites so as to improve the whole system's data access performance and reduce bandwidth consumption for data delivery. Several data replication algorithms have been proposed. Some were designed based on unlimited storage. However, not all Data Grids are with unlimited storage space. Others were implemented on limited storage environments. However, none of the algorithms developed on limited storage environments has considered file popularity, defined as how often a file is accessed by users. In fact, file popularity and data access patterns of a system vary with time since users sometimes change their interests where data access patterns, defined as the distribution of access counts on files of a system, may influence on the data access performance of the system. In other words, the file replication model of the system might not be able to adapt to the change of users' data access behaviors. Therefore, in this study, we proposed an adaptive data replication algorithm, called Popular File Replication First algorithm (PFRF for short), which is developed on a star-based Data Grid with limited storage space. With aggregated data access information of previously job execution and user behaviors, PFRF can predict future file popularity, and replicate potential popular files/replicas to appropriate cluster/sites to adapt the change. We employ several types of file access behaviors, including Uniform, Geometric, and Zipf-like distributions, to evaluate PFRF. The simulation results show that PFRF can effectively shorten average job response time, reduce bandwidth consumption for data delivery, and increase data availability as compared with the tested algorithms.

*Keywords*: star-based Data Grid, data replication algorithm, Zipf-like distribution, Geometric distribution

## 1. Introduction

Generally, a Data Grid, a specific Grid system that very usual provides a huge amount of storage space, often maintains a high volume of distributed data to serve users. Many recent scientific studies [1], engineering applications [2], and commercial applications [3], e.g., Biomedical Informatics Research Network (BIRN) [4], the Large Hadron Collider (LHC) [5], the DataGrid Project (EDG) [6], and physics Data Grids [7][8], have collected a huge number of data files and performed their complex experiments and analyses on Data Grids.

In a Data Grid, according to 80/20 rules a part of files is frequently accessed and transferred. If a system does not allow the existence of replicas, a file that a job frequently accesses is possibly located at a remote site. The data access efficiency of the job will be then poor. Long distance data transfer always occupies a lot of bandwidth and conducts long transmission delays. So how to decrease data access latency, lower bandwidth consumption for data transmission, and increase data availability have been the key research issues of Data Grids [9]. Data replication, a general and simple approach to achieve these goals, has been widely used in many areas, such as in the Internet and distributed databases [10][11]. A well-defined data replication method should meet the following requirements [9][12][13][14], including being able to determine an appropriate time to replicate files, determining which files should be replicated, and storing these replicas in appropriate locations.

On the other hand, data access pattern analyses have been the critical steps in designing efficient dynamic data replication schemes [15][16][17]. Several distributions have been used to model data access patterns defined as the distribution of access counts on files of a system, and file popularity defined as how often a file is accessed by users, i.e., how popular a file is [18][19]. Breslau et al. [18] claimed that using a Zipf-like distribution can more accurately model the distribution of webpage accesses. Cameron et al. [19] showed that the distribution of file accesses in Data Grids follows the Zipf-like distribution. Ranganthan and Foster [12][21] claimed that the Geometric distribution can properly model property of temporal/geographical locality and file

access behaviors, and they in [22] derived file popularity by using both Zipf and Geometric distributions under the assumption that Grid storage is unlimited or the storage is sufficient to keep all files and their replicas. Tang et al. [13] used Zipf-like and Geometric distributions to simulate users' file access behaviors on a multi-tier Data Grid. Furthermore, [14][26] proposed two data replication strategies on limited storage. But, [26] did not deal with file popularity and data access pattern. [14] did not consider the fact that file popularity is changed with time. In fact, the change influences the performance of the system employing the replication strategies.

Therefore, in this study, we propose an adaptive data replication algorithm, called Popular File Replication First algorithm (PFRF for short), which is developed on a star-based Data Grid with limited storage space. A star-based Data Grid, a cluster Data Grid with a center cluster that connects all other clusters, is a hierarchical architecture that can significantly reduce workload of user requests [20]. We simulate several cases in which file popularity follows Zipf-like distribution, Geometric distribution, and Uniform distribution under the assumption that user behaviors vary with the change of user interests. To adapt the change, PFRF aggregates file access information and replicates popular files to suitable clusters/sites. Three metrics including response time, data availability, and bandwidth cost ratio were employed to evaluate the tested algorithms, where bandwidth cost ratio as a new metric will be defined later. The simulation results show that PFRF provide users with a system that has higher data availabilities, lower data transmission delays, and less bandwidth consumption for data access.

The rest of the paper is organized as follows. In Section 2, we introduce related work and background of this study. Section 3 introduces architecture of a star-based Data Grid and the PFRF. Simulation results are described and discussed in Section 4. Section 5 concludes this article and addresses our future research.

## 2. Background and Related Work

In this section, we describe the architectures of Data Grids and their replication strategies and

algorithms.

## 2.1 A Data Grid Architecture

Data Grids can be classified into multi-tier Data Grids, first proposed by MONARC project [27][13], and cluster Data Grids, initially proposed by Chang et al. [26]. Fig. 1 illustrates the multi-tier Data Grid architecture in which a leaf node represents a user or a computation node, and internal nodes are resource sites keeping sharable files. In this architecture, a file, held by a site, e.g., by site *n*-*i* shown in Fig. 1, will be also held by all the ancestor sites, i.e., sites *n*-*i*-1, *n*-*i*-2, …, 1 and 0, $i$=0, 1, 2, 3,…, $n$-1. Therefore, the root (i.e., site 0) will hold all files that the system has. When an end user, e.g., node $n$, requires a file which does not exist in $n$, $n$ then requests the file from its immediate ancestor, i.e., node $n$-1. If node $n$-1 does not have the file, it requests its immediate ancestor, i.e., node $n$-2, to give it the file. The process repeats until a node, e.g., node $j$, which holds the file duplicates the file to node $j+1$. The file will be then delivered to node $n$ following the reverse direction of the requests. By using the multi-tier Data Grid, file access latency can be reduced, but files should be redundantly stored. Its maintenance cost is high.

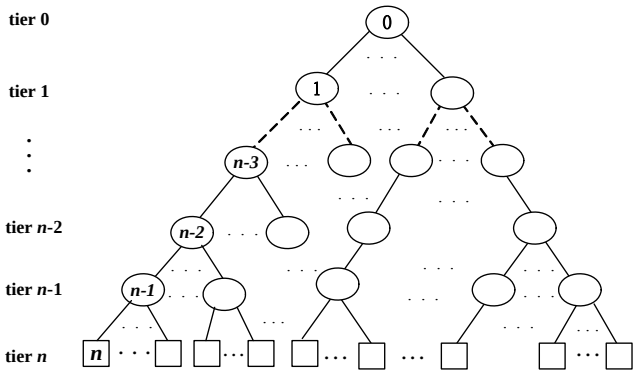



Fig. 1. A multi-tier Data Grid architecture.

As illustrated in Fig. 2, a cluster Data Grid consists of $n$ clusters connected by the Internet. Files are distributed to and stored in these clusters. Each cluster has a header node (a header for short) responsible for managing site information and exchanging file access information with other cluster headers. A header periodically, every T time units, determines which file should be

replicated to which cluster. After that, the header replicates the file with the largest weight to all clusters that need the file. Assume that in a specific T, a node, e.g., node A in cluster 1, frequently accesses a remote file, e.g., file F, with the largest weight. F will be then replicated to cluster 1 so that A can locally and quickly retrieve F in the next T. Comparing the two types of Data Grids, the cluster Grid consumes less storage to hold files.

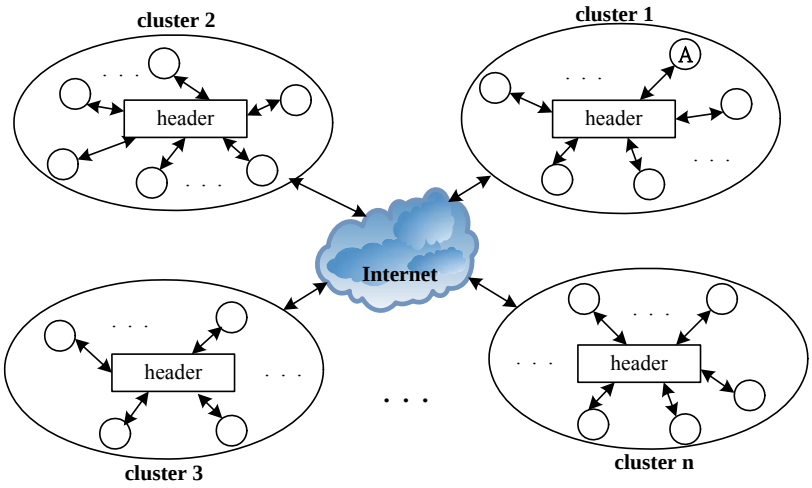



Fig. 2. A cluster Data Grid architecture.

## 2.2 Data Replication Algorithms/Strategies

Least Frequently Used (LFU) [28] and Most Frequently Used (MFU) [28] are two simple dynamic replication strategies widely used in many areas, such as disk and cache memory duplication. If a storage device has insufficient space to hold a new file, LFU (MFU) will be invoked to choose the files that have been least (most) frequently used as the victims to make room for the new one. However, MFU's characteristic contradicts our replication strategy. So, only LFU is involved in the following experiments.

On a multi-tier Data Grid, Ranganathan and Foster [12] presented six replication/caching strategies: No Replication or Caching, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replication and Fast Spread, among which the experimental results showed that the Fast Spread's and Cascading Replication's performance is better and file access latency is shorter

than those of the other four. However, the six strategies cannot avoid the multi-tier Data Grid's disadvantages stated above (recall a file is often redundantly stored in tiers). In fact, the storage space and access latency is a trade-off [26].

Tang et al. [13] introduced Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU) algorithms to reduce the average data access response time for a Data Grid. However, SBU does not consider historical access records for files, whereas a node in ABU sends the aggregated historical records to the upper tiers. The upper tiers do the same until these records reach the root. Due to the aggregation capability, ABU's job response time is shorter and its bandwidth consumption is less than those of the SBU. However, ABU and SBU were developed on multi-tier Data Grids.

Schintke et al. [24] and Ranganathan et al. [23] individually proposed one data replication algorithms to improve data availability. Files are arbitrarily replicated as needs, even though users only access these files for just once, thus wasting too much storage space to keep useless replicas. Kunszt et al. [25] also introduced a file-based replication method to manage a Grid middleware, with which file access/transfer time can be then reduced. However, the three algorithms were all developed on unlimited storage space and multi-tier Data Grids.

Chang et al. presented Latest Access Largest Weight (LALW) dynamic replication strategy in [14] and Hierarchical Replication Strategy (HRS) in [26]. The LALW utilizes the half-life concept to weigh files. A file with a higher access frequency has a larger weight. With the weight, the LALW outperformed LFU and no replication data replication strategies [12] in bandwidth utilization. However, they did not consider the fact that file popularity is changed with time, and only the most popular file is replicated in each time interval. The HRS, a dynamic replication strategy for a cluster Data Grid, replicates a locally needed file to the local site from an appropriate site so as to reduce future-file-access communication costs. However, the HRS did not consider file popularity and data access pattern.

## 3. System Framework

The proposed star-based Data Grid architecture as shown in Fig. 3 consists of a *global replica controlle*r (GRC) and several clusters connected to the GRC through the Internet. Each cluster comprises sites connected by a LAN or LANs, and a local replica controller (LRC) which maintains a local replica table ($LRT$) to record file access information, including filename, file location, access count, file weight, and master file information. In this study, a master file is an original file that cannot be deleted from the Data Grid. Files are stored in sites of different clusters. The GRC, a centralized server located on the Internet, is responsible for aggregating file access records for all clusters and determining which files should be replicated to which clusters. To achieve these, it maintains a global replica table ($GRT$) to collect the information recorded in $LRT$s, e.g., the first record of $GRT$ shown in Fig. 4 indicates that the information of the file $F_1$ is kept in both $LRT_1$ and $LRT_3$. $LRT_1$ shows that $F_1$ as a master file with current weight 1 is now stored in site 1 and has been accessed for 5 times. When the GRC determines to replicate files to a cluster, e.g., cluster $i$, it records the location of the new replicas in $GRT$ so that it can reply the location of files/replicas requested by LRCs. $LRT_i$ of course will record information of the replicas.

Fig. 5 illustrates the replica catalog structure of the star-based Data Grid, in which GRC is the root. When site 1 needs a file, it checks $LRC_1$ to see whether the file exists in cluster 1 or not. If not, $LRC_1$ requests GRC to provide the file location. Then, the $LRC_1$ replies site 1 with the file location.

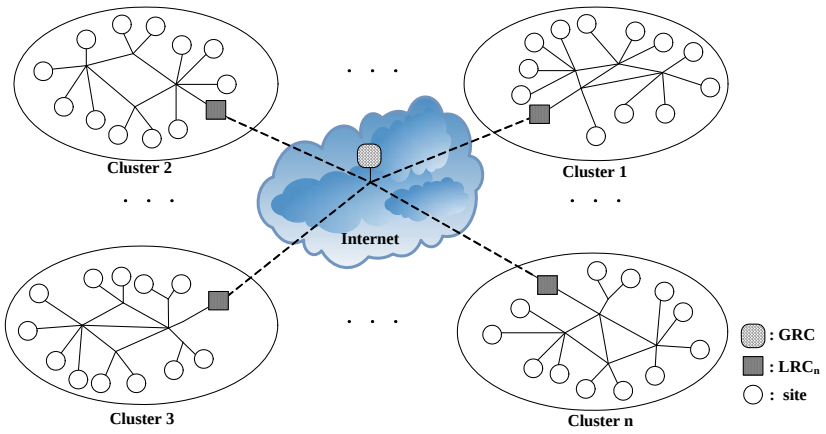



Fig. 3. The star-based Data Grid architecture.

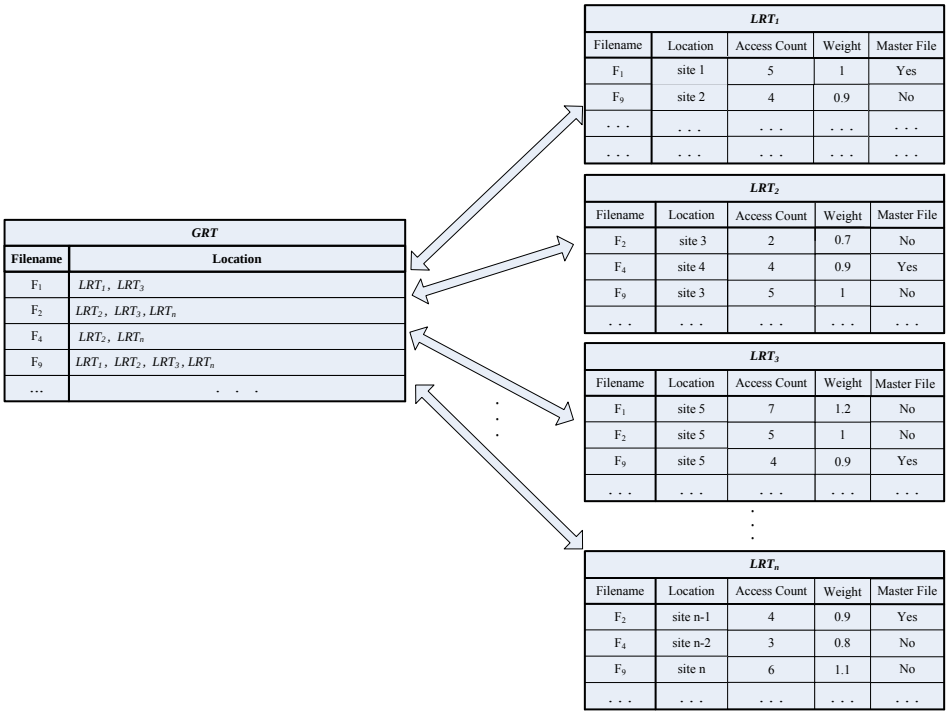



Fig. 4. An example of $GRT$ and $LRTs$.

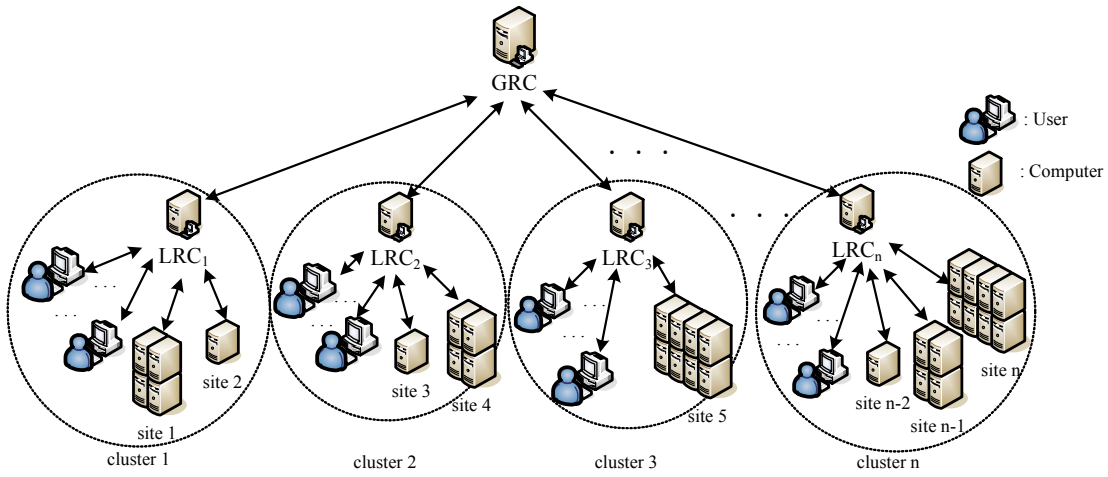



Fig. 5. The hierarchical replica catalog topology of a star-based Data Grid.

## 3.1 Zipf-like distribution and Geometric distribution

To achieve a better file access performance, we need to keep track of the changing behaviors for users' file accesses so that we can accurately predict which files will be accessed frequently in the near future. The prediction is a main task of the data replication algorithm/strategy based on the assumption that a popular one will be accessed more frequently than unpopular files will [13]. This

assumption is called temporal locality [12]. Breslau et al. [18] as stated above showed that the webpage requests follow a Zipf-like distribution [19][29], which is derived from Zipf's law [30] and in which the access probability of the $i$-th most popular file, denoted by $P(f_i)$, is

$$P(f_i) = 1/i^{\alpha} \qquad (1)$$

where $i$=1, 2,…, n and $\alpha$ is a factor to determine the file access distribution, $0 \leq \alpha < 1$.

Ranganathan and Foster [21][22] adopted Geometric distribution to simulate the file popularity in which the access probability of the $i$-th most popular file, denoted by $P(i)$, is

$$P(i) = (1-p)^{i-1} \cdot p \qquad (2)$$

where $i$=1, 2,…, n and $0 < p < 1$. A larger value of $p$ represents that a smaller portion of files has been frequently accessed. In this study, we assume that the users' access behaviors follow both Zipf-like distribution or Geometric distribution with different parameters, with which PFRF is developed.

### 3.2 Popular File Replication First (PFRF) algorithm

The PFRF algorithm as illustrated in Fig. 6 is performed by GRC at the end of a round where a round is a dynamic period of time $T_d$ in which at least a fixed number of jobs, e.g., *x* jobs (e.g., 10 or 20 jobs), is submitted by each cluster. $T_d$ has its maximum value $T$. If some clusters do not generate at least *x* jobs in $T$, $T$ will be treated as a round. The algorithm comprises four phases: file access aggregate phase, file popularity calculation phase, file selection phase, and file replication phase.

1. File access aggregate phase: Between lines 2 and 5 of the algorithm, PFRF aggregates the access count for each file, e.g., file $f_i$ in cluster c, at a round, e.g., round $r$, denoted by $\mathrm{A}_{\mathrm{c}}^{r}(f_i)$, sorts all the files on $\mathrm{A}_{\mathrm{c}}^{r}(f_i)$s in a descending order and stores the sorted result into a set S. After that, PFRF calculates the total number of files having been accessed by all sites in cluster c at round $r$, denoted by $\mathrm{TNF}_{\mathrm{c}}^{r}$, based on the information stored in $\mathrm{LRC}_{\mathrm{c}}$.
2. File popularity calculation phase: At line 6, PFRF calculates popularity weight for file $f_i$,

denoted by $PW_{\mathrm{c}}^{r}(f_i)$, where $i = 1,2,\ldots$ and

$$PW_{\mathrm{c}}^{r}(f_i) = \begin{cases} PW_{\mathrm{c}}^{r-1}(f_i) + \mathrm{A}_{\mathrm{c}}^{r}(f_i) \cdot \mathrm{a} & \text{if } \mathrm{A}_{\mathrm{c}}^{r}(f_i) > 0 \\ PW_{\mathrm{c}}^{r-1}(f_i) - \mathrm{b} & \text{otherwise} \end{cases}, \quad r \geq 1, \mathrm{c} \geq 1, i \geq 1. \qquad (3)$$

in which a and b are constants and $\mathrm{a} < \mathrm{b}$. We will discuss why $\mathrm{a} < \mathrm{b}$ later. If $\mathrm{A}_{\mathrm{c}}^{r}(f_i) > 0$, i.e., $f_i$ has been accessed by users in round $r$, PFRF increases $PW_{\mathrm{c}}^{r-1}(f_i)$ by $\mathrm{A}_{\mathrm{c}}^{r}(f_i) \cdot \mathrm{a}$. Otherwise, it decreases $PW_{\mathrm{c}}^{r-1}(f_i)$ by b. Basically, a higher $PW_{\mathrm{c}}^{r}(f_i)$ implies that $f_i$ is more popular. In this study, we assume that at round 0 all files follow binomial distribution, i.e., $PW_{\mathrm{c}}^{0}(f_i) = 0.5$, indicating the initial probability that $f_i$ is accessed is 0.5, and the minimum value of each $PW_{\mathrm{c}}^{r-1}(f_i)$ is 0. From previous access records of $f_i$, PFRF can derive the variation of the popularity of $f_i$ and predict the popularity of $f_i$ for the next round. For instance, if $f_3$ has been accessed 5 times by cluster 2 in round 1, $PW_{2}^{1}(f_3) = 0.5 + 5 \cdot \mathrm{a}$.

The average popularity of $f_i$ in all clusters, denoted by $PW_{\mathrm{avg}}^{r}(f_i)$, is

$$PW_{\mathrm{avg}}^{r}(f_i) = \frac{\sum_{\mathrm{k}=1}^{N_{\mathrm{c}}} PW_{\mathrm{c}}^{r}(f_i)}{N_{\mathrm{c}}} \qquad (4)$$

where $N_{\mathrm{c}}$ is the total number of clusters having $f_i$ in the concerned Data Grid.

3. File selection phase: Between lines 7 and 10, PFRF sorts the set S on the average popular weights in a decreasing order, calculates $N_f$ which is the number of files that might be replicated, and selects the first $N_f$ files as cluster c's duplication candidates from S, where

$$N_f = \lfloor \mathrm{TNF}_{\mathrm{c}}^{r} \times (1 - x) \rfloor \qquad (5)$$

in which $x$ is a constant, $0 < x < 1$.

4. File replication phase: The replication phase, between lines 11 and 30 in Figure 6, first checks to see whether each file, e.g., file $f_i$, in cluster c's replication candidates is in cluster c or not. If yes, PFRF does nothing. Otherwise, it further checks to see whether there is a site in cluster c that has sufficient storage space to accommodate $f_j$ or not. If yes, PFRF duplicates $f_j$ to the site from a nearest cluster having $f_j$. Otherwise, PFRF deletes $v$ files ($v \geq 1$) that are less popular than $f_j$ from a site of cluster c, leaving enough storage space to keep $f_j$.

From the algorithm we can realize that files and the users may be located at different sites in the same cluster or different clusters. But PFRF tries to localize files since inter-cluster access is much more time-consuming than intra-cluster access is [14].

**PFRF data replication algorithm**

Input: The total access count of $f_i$, $i = 1,2,\dots,N_k$, where $N_k$ is the number of files in cluster c in round $r$;

Output: Duplicating the files that should be replicated to cluster c denoted by $F_c = \{f_1, f_2, \dots, f_{N_f}\}$ to cluster c;

```
1: for each cluster c, c = 1,2,...,N_c{      /*N_c is the number of cluster that the concerned Data Grid has*/
2:     Aggregate A_c^r(f_i), i = 1,2,...,N_k; /*N_k is the number of files that cluster c has*/
3:     Sort all the files according to A_c^r(f_i) in a descending order;
4:     Store the sorting result into set S;
5:     Calculate TNF_c^r for LRC_c;
6:     Calculate PW_c^r(f_i) and PW_avg^r(f_i), i = 1,2,...,N_k;   /* equations (3) and (4)*/
7:     Sort the set S according to the average popular weights;
8:     Set the value of x = 0.8, 0 < x < 1;      /*according to 80/20 rules*/
9:     Calculate N_f with equation (5);   /*N_f derived from equation (5) on x=0.8 is the number of
                                            popular files. */
10:    Select the first N_f files from S and put them into the set S'; /* S': the set of replication
                                                                          candidates*/
11:    for each file f_j in S'{
12:        check LRC_c to see whether cluster c has f_j;
13:        if cluster c does not have f_j{
14:            if any site of cluster c has sufficient storage to save f_j
15:                replicate f_j to the site from a nearest cluster which has f_j;
16:            else if{ for each site Y in cluster c { /* check storage space of each site Y in cluster c*/
17:                    for each file f_k kept in site Y;
18:                    compare PW_avg^r(f_k) with PW_avg^r(f_j);
19:                    if there are v files that are less popular than f_j and the total size of these
20:                      v files plus the remaining storage space of site Y is sufficient to keep f_j
21:                      { PFRF deletes these v files and replicates f_j to Y;
22:                        break;}
23:                }
24:            }
25:            else    /*The total size of all files that are less popular than f_j plus the remaining storage
```

```
                        space is not enough to hold f_j*/
26:             {   print ("no sites in cluster c can keep f_j");
27:                 PFRF will not replicate f_j;
28:             }
29:         }
30:     }
31:}
```

Fig. 6. PFRF data replication algorithm.

## 4. Simulation and Performance Comparison

To evaluate the proposed scheme, a Grid testbed or a Grid simulator is required. However, the maintenance cost of a real testbed is high and our TigerGrid [31] is not organized as a star topology. In fact, the cost of reconfiguring the TigerGrid to a star topology is also high. Thus, we choose a Grid simulator as the simulation tool. Many Grid simulators have been introduced, such as MicroGrid [32], OptorSim [33], SimGrid [34], MONARC [35], ChicSim [36], and GridSim [20], in which GridSim due to providing a flexible and extensible simulation environment and allowing researchers to increase new components/functions is chosen to construct our test platform. The compared algorithms include LFU and No Replication (NR) data replication algorithms.

### 4.1 Experimental Environment and Parameters

The test environment illustrated in Fig. 8 consists of a GRC and four clusters. Each cluster has a LRC. The specifications of all resources and job parameters are listed in Table 1. Each site comprises six computers, and each computer has four processors, i.e., each cluster has 24 processors. A processor's processor rating is 1,600 MIPS. Therefore, the total processor rating of a cluster is 38,400 (=24×1,600) MIPS. Each cluster has 50GB storage space to accommodate files, and the file size of a master file is 1GB. Fig. 9 illustrates that job execution is divided into rounds based on the definition defined above. We assume that $T_d$=800 seconds and in each round at least 10 jobs are submitted by users in each cluster, i.e., at least 40 jobs are submitted, and each job requests 5 to 10 files. Fig. 10 shows an example of job execution in cluster c in which the

durations of different rounds may be different. Each simulation is performed twenty rounds, and we assume that users in cluster c in the first round cannot locally access files $F_1$, $F_4$, and $F_9$, so it needs more time to duplicate these files from remote clusters. Thus, uses can locally retrieve the three files in the second round. Theoretically, the duration of a later round is shorter than any previous rounds if users do not change their file access behaviors. The phenomenon will be shown in the following simulation.

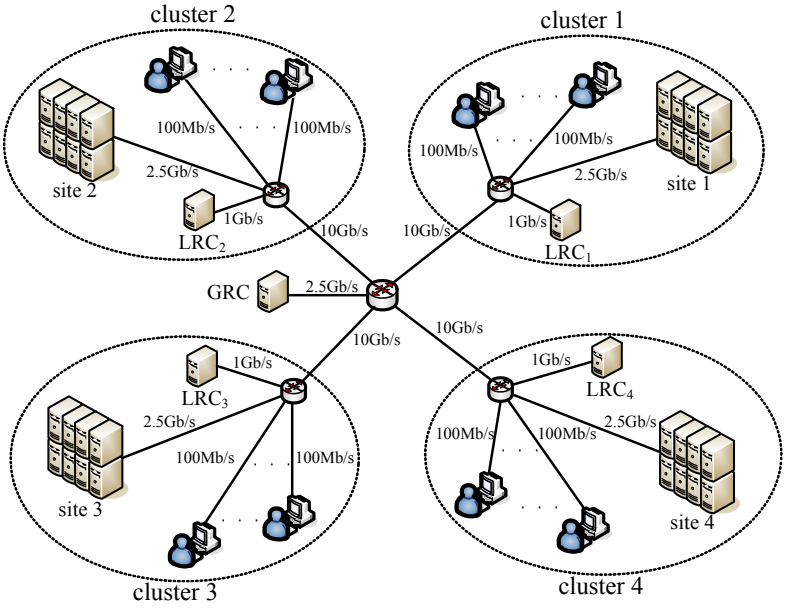



Fig. 8. The simulation Topology.

Table 1 Resources and Job parameters

| Resources | Value |
|---|---|
| Number of clusters | 4 |
| Storage available in a cluster | 50GB |
| Single processor rating (MIPS) | 1,600 |
| Number of processors in a cluster | 24 |
| Processor rating of a cluster | 38,400 (=24×1600) |
| Inter-router bandwidth | 10Gb/s |
| Router-to-site bandwidth | 2.5Gb/s |
| User-to-router bandwidth | 100Mb/s |
| GRC-to-router bandwidth | 2.5Gb/s |
| LRC-to-router bandwidth | 1Gb/s |
| Job parameters | Value |
| Number of master files | 100 |

| | |
|---|---|
| Minimum number of jobs performed by users in each cluster in a round | 10 |
| The minimum number of jobs performed in each round | 40 (at least 4×10 jobs) |
| Size of a master file | 1GB |
| Number of files accessed by a job | 5~10 |
| The longest duration of a round ($T_d$) | 800 seconds |

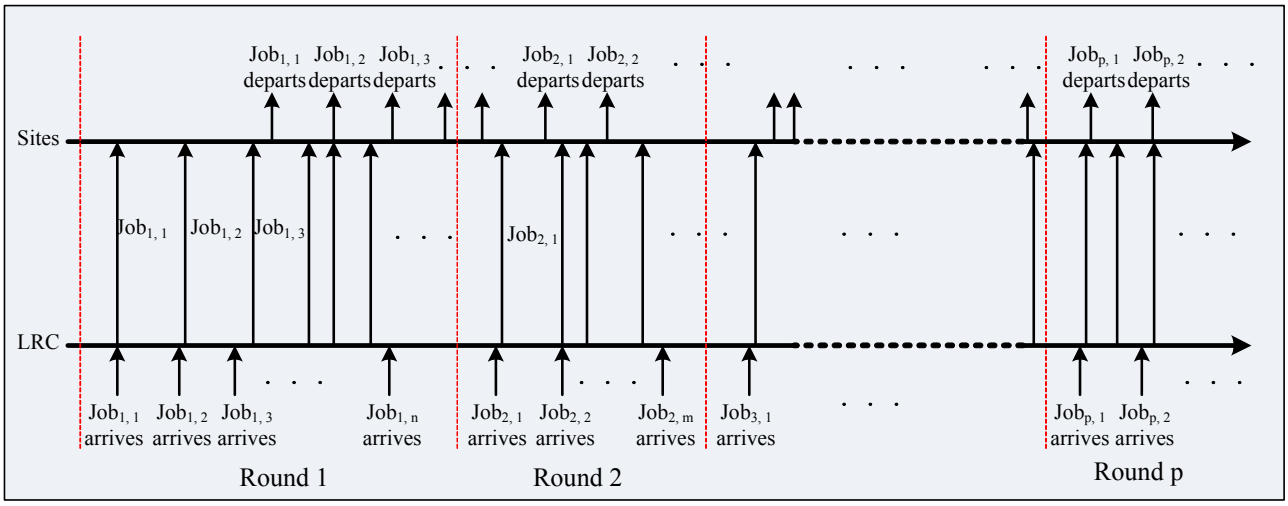



Fig. 9. Job execution in a cluster in different rounds.

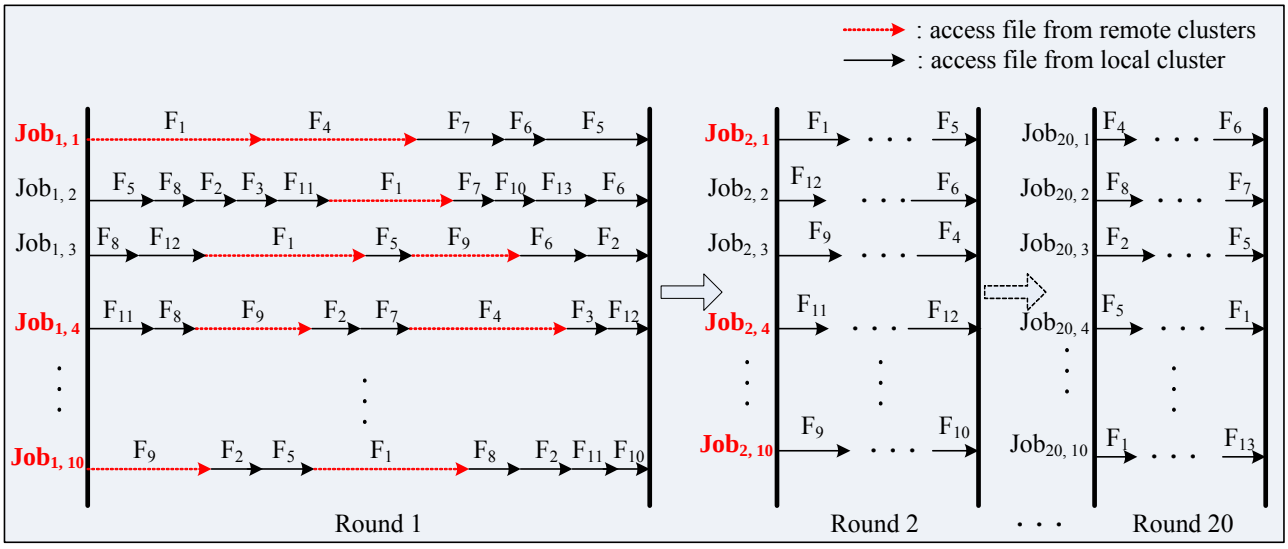



Fig. 10. Files accessed by jobs of cluster c in different rounds.

To compare PFRF and LFU algorithms, 100 master files are randomly distributed to the four clusters (see Table 2). With NR algorithm, there are two cases. The first, denoted by NR case1, is

that 100 masters files are all stored in GRC, and the other, denoted by NR case2, is that the 100 files are randomly distributed to the four LRCs. Neither cases replicate files. In NR case1, a cluster does not locally hold a file. Hence, a job has to remotely access GRC to acquire the files. In NR case2, some accessed files are local and some are remote. So if users would like to access a non-local file, they have to remotely retrieve it from other cluster, instead of from GRC.

Table 2 Master files settings for PFRF, LFU, NR case1, and NR case2 algorithms

| Data replication algorithm | Setting |
|---|---|
| PFRF | 100 master files are randomly distributed to the four clusters |
| LFU | 100 master files are randomly distributed to the four clusters |
| NR case1 | 100 master files are all stored in GRC |
| NR case2 | 100 master files are randomly distributed to the four LRCs |

## 4.2 Access Patterns

Ten access patterns listed in Table 3 are employed to simulate user file access behaviors. File popularity is calculated as stated above under the assumption that file accesses follow Zipf-like, Geometric, and Uniform distributions where Uniform distribution (Uniform for short) represents that the probability of accessing a file by each user is the same. JRR, standing for job repeating rate, is the probability of accessing the files that were accessed in the previous round, $0 \leq \text{JRR} \leq 1$, and $\alpha$ and $p$ are the parameters respectively used when employing Zipf-Like distribution and Geometric distribution.

Table 3 Different access patterns employed

| No. | File Popularity | $\alpha/p$ | JRR (%) | $p(f_i)/p(i)$ |
|---|---|---|---|---|
| 1 | Zipf-like | 0.8 | 0 | $1/i^{0.8}$ |
| 2 | Zipf-like | 0.8 | 25 | $1/i^{0.8}$ |
| 3 | Zipf-like | 0.6 | 0 | $1/i^{0.6}$ |
| 4 | Zipf-like | 0.6 | 25 | $1/i^{0.6}$ |
| 5 | Geometric | 0.2 | 0 | $(1-0.2)^{i-1} \cdot 0.2$ |
| 6 | Geometric | 0.2 | 25 | $(1-0.2)^{i-1} \cdot 0.2$ |
| 7 | Geometric | 0.5 | 0 | $(1-0.5)^{i-1} \cdot 0.5$ |
| 8 | Geometric | 0.5 | 25 | $(1-0.5)^{i-1} \cdot 0.5$ |
| 9 | Uniform | none | 0 | none |

| | | | | |
|---|---|---|---|---|
| 10 | Uniform | none | 25 | none |

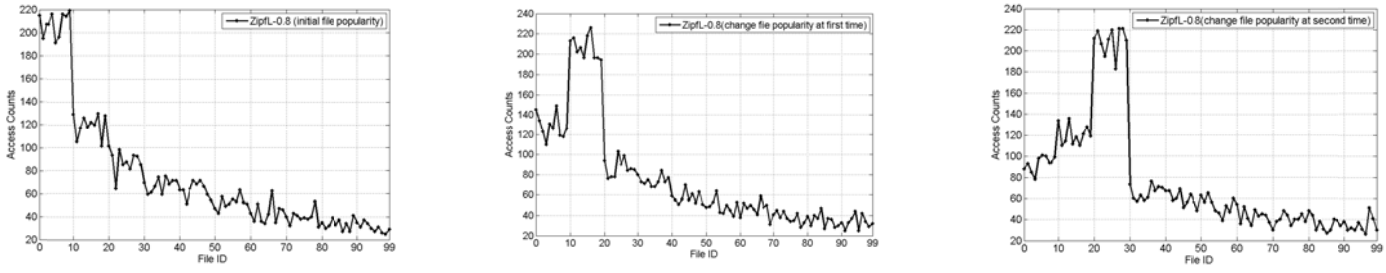
To effectively analyze the proposed system, we evenly partition the popularities of the 100 master files into 10 levels, and divide consecutive twenty rounds into three phases. As listed in Table 4, the first phase includes rounds 1 to 7. The second and the third respectively contain rounds 8 to 14 and rounds 15 to 20. In the first phase, we assume that File0 to File9 are the most popular files, i.e., belonging to the first popularity level. File10 to File19 are the second popular files, hence they are the second popularity level, and so on. In the second phase, we swap the files of the first two popularity levels, i.e., File10 to File19 become the most popular, File0 to File9 become the second, to simulate file popularity change, and other files' popularities remain unchanged. In the third phase, File20 to File29 are the most popular, File10 to File19 the second, and File0 to File9 the third. Other levels' file popularities remain unchanged. With the settings listed in Table 4, we conduct the following experiments to evaluate whether PFRF on LFU, NR case1, and NR case2 can adapt themselves to the change of file popularities or not.

In the following experiments, 1000 jobs, instead of 40 jobs, were submitted in each phase. The experimental results are illustrated in Figs. 11 to 15, in which Figs. 11a to 15a show users' file access behaviors in the first phase; Figs. 11b to 15b (Figs. 11c to 15c) plot those in the second (the third) phase. Taking ZipfL-0.8 (see Fig. 11) as an example, the access count (AC) of each most popular file in all the three phases (figures) is about 215, and unpopular files, i.e., File30 to File99, are accessed less and less. In the case of ZipfL-0.6 (see Fig. 12), the AC of each most popular file in all the three phases is about 170. However, the ACs of the other popularity levels, i.e., between levels 4 and 10, in the three phases are not evidently different, like a Uniform distribution. When Geo-0.2 is invoked (see Fig. 13), the AC of each most popular file is about 175, and the ACs decline sharply when file IDs increase. In the Geo-0.5 case (see Fig. 14), the difference between/among the most popular files' ACs and those of the second and the third popular files in each of the three phases is significant. Generally, the ACs of the first three popularity levels on all access patterns are

clearly different from those of the other popularity levels, implying that File0 to File29 are frequently accessed, while File70 to File99 are rare. The difference among the ACs of different popularity levels on the Uniform as shown in Fig. 15 is not significant.

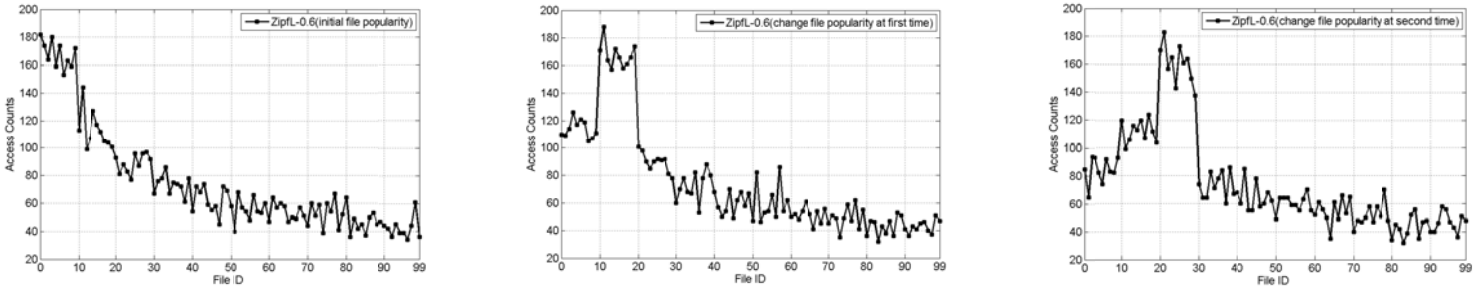
Table 4 The file popularities in the three phases (Rounds 1~7, 8~14, and 15~20)

| Phases / Popularity level (Files) | Phase 1 (Rounds 1~7) | Phase 2 (Rounds 8~14) | Phase 3 (Rounds 15~20) |
|---|---|---|---|
| 1 (File0~File9) | 1st Popular | 2nd Popular | 3rd Popular |
| 2 (File10~File19) | 2nd Popular | 1st Popular | 2nd Popular |
| 3 (File20~File29) | 3rd Popular | 3rd Popular | 1st Popular |
| … | … | … | … |
| 10 (File90~File99) | 10th Popular | 10th Popular | 10th Popular |



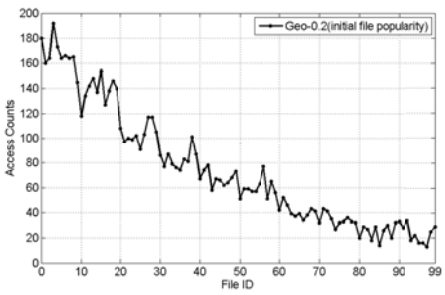

(a) phase 1 (b) phase 2 (c) phase 3

Fig. 11. Distributions of file requests in the simulation process on ZipfL-0.8.

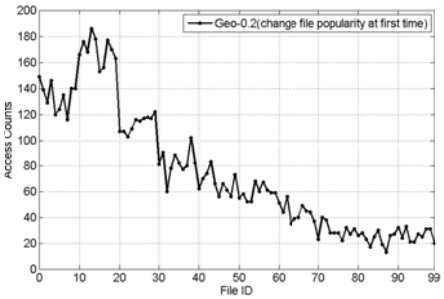



(a) phase 1 (b) phase 2 (c) phase 3

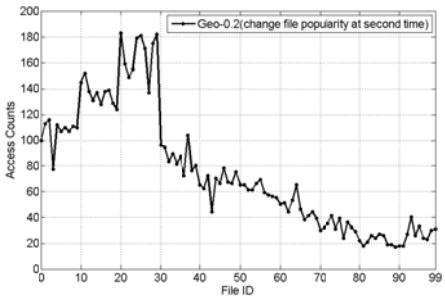
Fig. 12. Distributions of file requests in the simulation process on ZipfL-0.6.




(a) phase 1

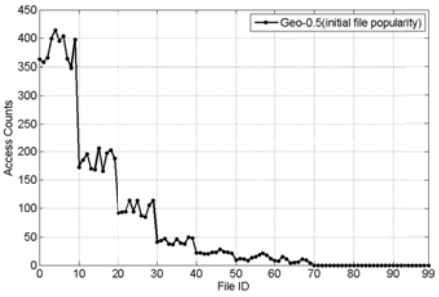



(b) phase 2

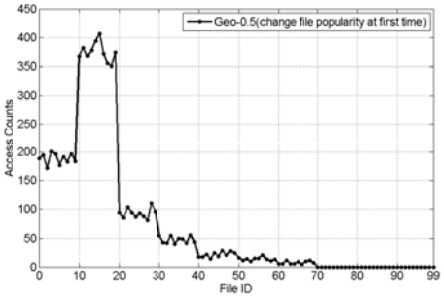



(c) phase 3

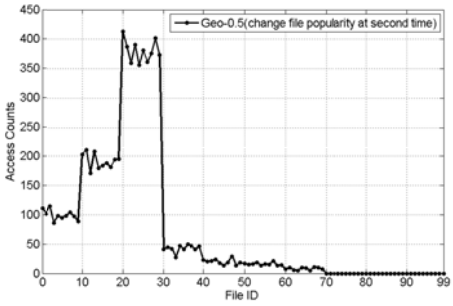
Fig. 13. Distributions of file requests in the simulation process on Geo-0.2.




(a) phase 1

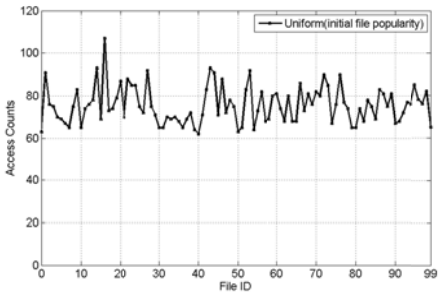



(b) phase 2

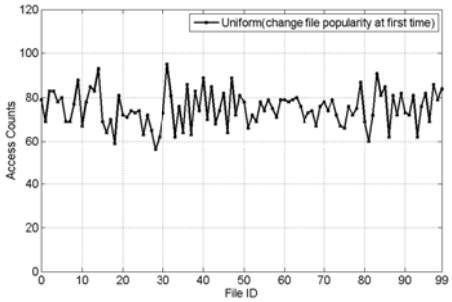



(c) phase 3

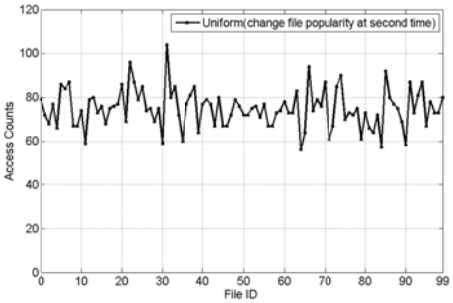
Fig. 14. Distributions of file requests in the simulation process on Geo-0.5.




(a) phase 1




(b) phase 2




(c) phase 3

Fig. 15. Distributions of file requests in the simulation process on Uniform.

## 4.3 Simulation results

The tested algorithms are run on the same experimental environment so we can fairly compare the performance. Several test metrics are used. The first is *Job response time (in seconds)* defined as the time interval from the time point when a job sends a file request to its LRC to the time point

when the requested files are successfully delivered. The second is *Data availability* which proposed by GridSim [20] for a job, e.g., $\text{Job}_x$, to access $N_x$ files stored in cluster y, denoted by $Avail_{xy}$, is formally defined as

$$Avail_{xy} = \frac{t_{xy}}{N_x} \qquad (6)$$

where $t_{xy}$ is the time that $\text{Job}_x$ consumes to acquire the $N_x$ files from cluster $y$. The average data availability over all jobs, e.g., $\text{Jobs}_y$ submitted to $y$, denoted by $avgAvail_y$, in a round is defined as

$$avgAvail_y = \frac{\sum_{x \in \text{Jobs}_y} Avail_{xy}}{|\text{Jobs}_y|}. \qquad (7)$$

The *bandwidth cost ratio* of cluster c in a round, denoted by $BCR_c$, is defined as

$$BCR_c = \frac{LC_c \cdot LFA_c + RC_c \cdot RFA_c}{C_{baseline} \cdot AFA_c} = \frac{LC_c \cdot LFA_c + RC_c \cdot RFA_c}{C_{baseline} \cdot (LFA_c + RFA_c)} \qquad (8)$$

where $LFA_c$ ($RFA_c$) is the number of files that cluster c can locally (should remotely) accessed, $AFA_c = LFA_c + RFA_c$, $LC_c$ is the cost of file access inside cluster c, $RC_c$ is the cost of file access between cluster c and a remote cluster and $C_{baseline}$ is the access cost when the file request issued by a cluster c job is in a remote cluster. We further assume that all file have the same size, i.e., 1 GB. So formula (8) only involves number of files and neglects file sizes. The *Average Bandwidth Cost Ratio* (*ABCR*) defined as

$$ABCR = \frac{\sum_{u=1}^{N_c} BCR_c}{N_c} \qquad (9)$$

is to determine whether a data replication algorithm could accurately predict popular files or not where $N_c$ is the number of clusters the concerned Data Grid has. If $LFA_c$ is larger than $RFA_c$, that means the concerned data replication algorithm can more accurately predict user file access behaviors. Otherwise, the algorithm due to inaccurate prediction would consume a lot of network resources to remotely deliver files required by jobs. In other words, a data replication algorithm that produce a smaller *ABCR* value will result in better Grid performance.

In Section 4.3.1 and 4.3.2, each simulation is performed fifteen times.

4.3.1 Average Job Response Time (*ART*) and Data Availability (*DA*)

Figs. 16a to 20a show the experimental results of *ART*s for PFRF, LFU, and NR case2 given the ten access patterns. When ZipfL-0.8 with JRR=0% (JRR=25%) is used, after the third round (the fourth round), as shown in Fig.16a PFRF has shorter *ART*s than LFU has. Figs. 17a and 18a show that the experimental results on ZipfL-0.6 and Geo-0.2 are similar to those of Fig. 16a. PFRF's and LFU's response delays on Uniform with JRR=0% and with JRR=25% as shown in Fig. 20a are longer than those shown in Figs. 16a to 19a since the tested algorithms like that shown in Fig. 15 cannot effectively discriminate the file popularities for files. Also, in Figs. 16a to 18a and Fig. 20a, PFRF has shorter *ART*s than those of LFU, i.e., PFRF gives the best *ART*s among the tested algorithms. Nevertheless, when the data access pattern is Geo-0.5 with JRR=0% and 25% (see Fig. 19a), PFRF and LFU have similar *ART*s since the popular files as shown in Fig. 14 can be easily identified, and both of the two algorithms can accurately identify users' file access behaviors. Their *ART*s are the smallest, about 460 seconds, compared with PFRF's and LFU's *ART*s shown in Figs. 16a, 17a, 18a, and 20a. Table 5 lists *ART*s of PFRF on ZipfL-0.8 with JRR=25% in twenty rounds. The trend is that the duration of a later round as stated above is shorter than any previous rounds, except when the file popularities were changed in specific rounds, e.g., in rounds 8 and 15 since as listed in Table 4 we swap popular files after round 7 (the end of phase 1) and round 14 (the end of phase 2).

With NR case1 algorithm, a job in each round spent a longer time, about 1,740 seconds (not shown in Figs. 16a to 20a, otherwise the difference between PFRF's and LFU's *ART*s cannot be identified), to access files since all files are located in GRC. With NR case2, a job in each round spent about 730 to 760 seconds (see Figs. 16a to 20a). Apparently, *ART*s of both PFRF and LFU on all access patterns are all less than those of NR case2. From these 5 figures, we can also see that the *ART*s in the first three rounds reduced quickly, and PFRF effectively adapted itself to the change of file popularities.

Fig. 21 shows the replication delays of PFRF and LFU on ZipfL-0.8 with JRR=0% and

JRR=25%. Since PFRF learns users' file access behaviors step by step, its *ART*s reduce gradually. But LFU's *ART*s remarkably decline after first round since LFU continuously replicates needed files until storage of a cluster is full.

In fact, the difference of PFRF's *ART*s on JRR=0% and JRR=25% is insignificant. The key reason is that regardless of whether JRR=25% or 0% the access patterns do follow Zipf-like or Geometric distribution. The LFU's *ART*s have the similar phenomenon. But PFRF's (LFU's) *ART*s on Uniform with JRR=25% as shown in Fig. 20a are better than PFRF's (LFU's) *ART*s on Uniform with JRR=0% since with JRR=25%, the probability that the 25% of files are local in current round due to replication algorithms duplicating popular files to local cluster at the end of a round will be higher than that with JRR=0%.

Figs. 16b to 20b illustrate the *Data availabilities* for the ten access patterns. A relatively smaller *Data availability* represents a better performance. According to the definition of data availability presented in formula (6), the numerator $t_{xy}$ =*ART* so that *Data availabilities* are similar to the corresponding *ART*s, i.e., Fig. *X*(a) and Fig. *X* (b) are similar where *X*=16, 17, 18, 19, and 20. Note that NR case1 is also the worst, about 0.235 in each round (not shown, otherwise the difference between PFRF's and LFU's *Data availabilities* cannot be identified), in accessing files since files are all stored in GRC. No required files are local.

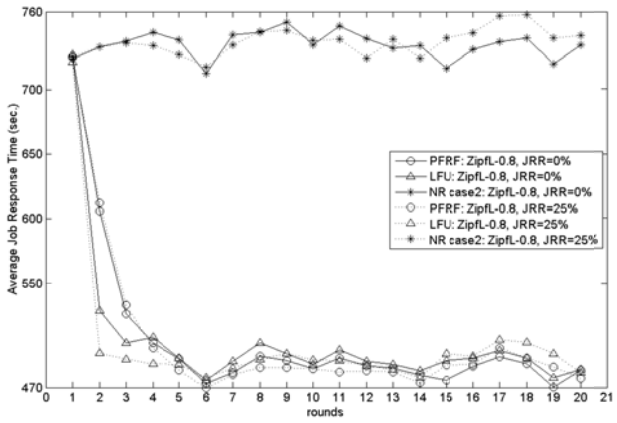



Fig. 16(a). Average job response time for PFRF, LFU, and NR case2 on ZipfL-0.8 with JRR=0% and 25%.

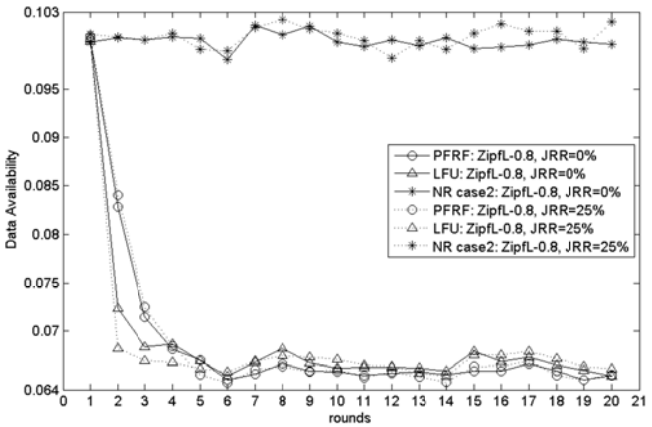



Fig. 16(b). Data Availability for PFRF, LFU, and NR case2 on ZipfL-0.8 with JRR=0% and 25%.

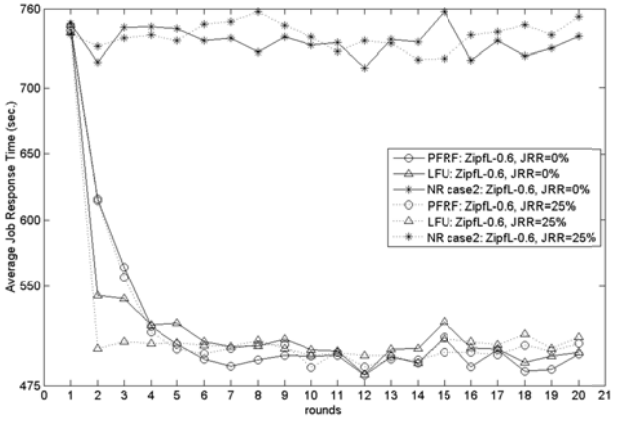



Fig. 17(a). Average job response time for PFRF, LFU, and NR case2 on ZipfL-0.6 with JRR=0% and 25%.

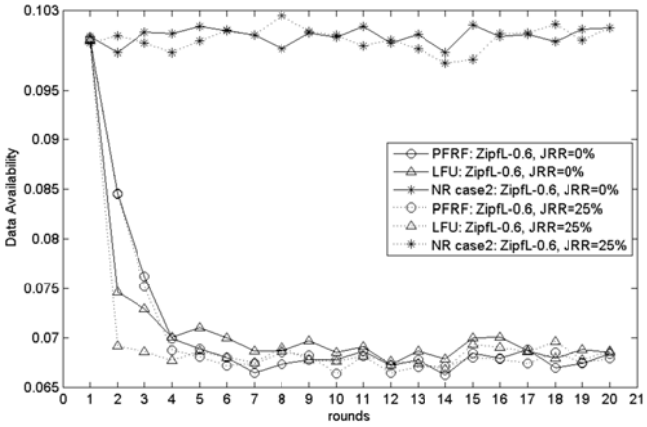



Fig. 17(b). Data Availability for PFRF, LFU, and NR case2 on ZipfL-0.6 with JRR=0% and 25%.

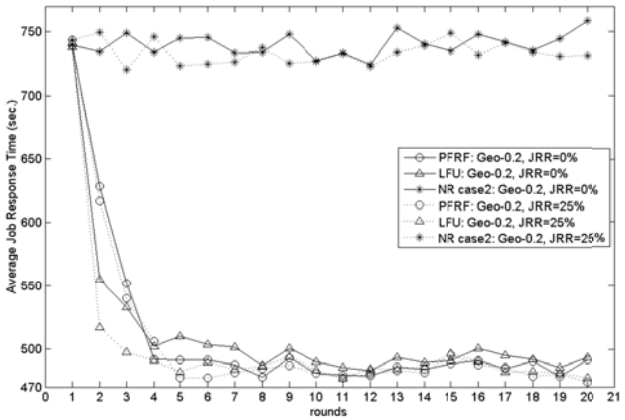



Fig. 18(a). Average job response time for PFRF, LFU, and NR case2 on Geo-0.2 with JRR=0% and 25%.

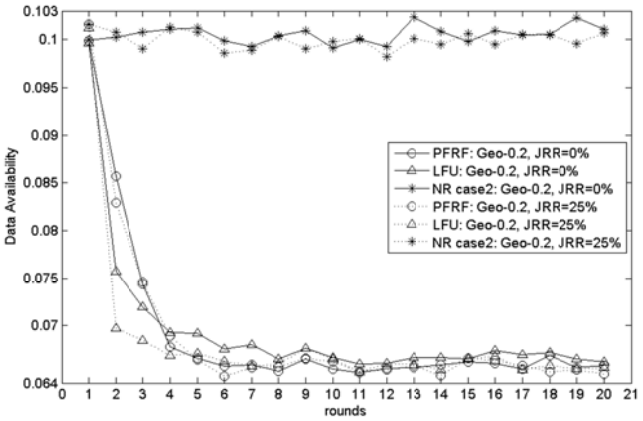



Fig. 18(b). Data Availability for PFRF, LFU, and NR case2 on Geo-0.2 with JRR=0% and 25%.

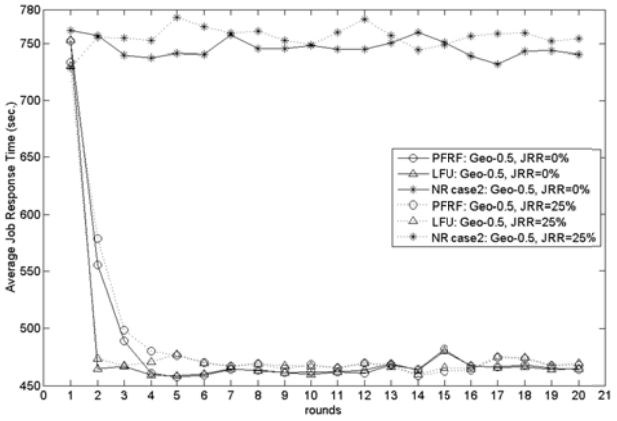



Fig. 19(a). Average job response time for PFRF, LFU, and NR case2 on Geo-0.5 with JRR=0% and 25%.

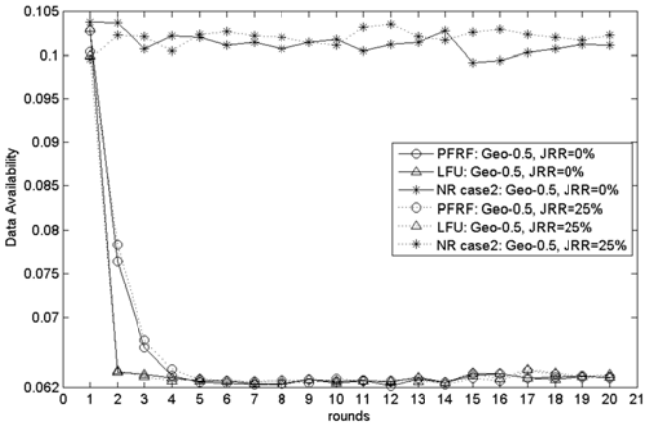



Fig. 19(b). Data Availability for PFRF, LFU, and NR case2 on Geo-0.5 with JRR=0% and 25%.

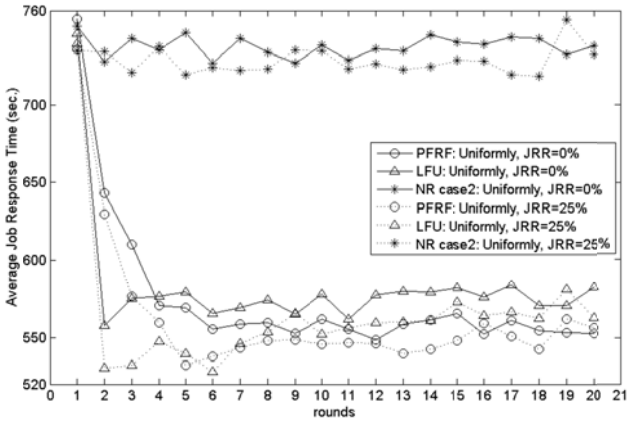



Fig. 20(a). Average job response time for PFRF, LFU, and NR case2 on Uniform with JRR=0% and 25%.

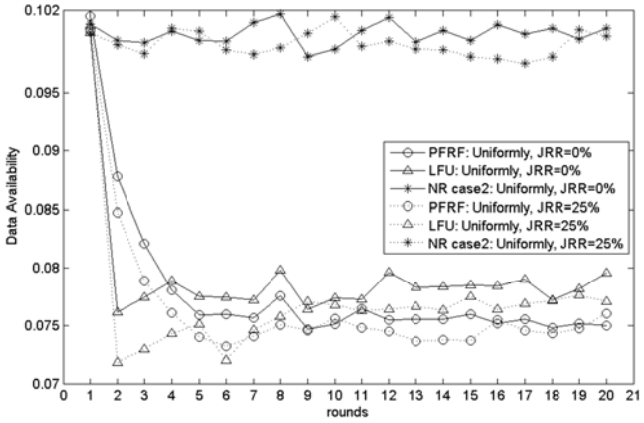



Fig. 20(b). Data Availability for PFRF, LFU, and NR case2 on Uniform with JRR=0% and 25%.

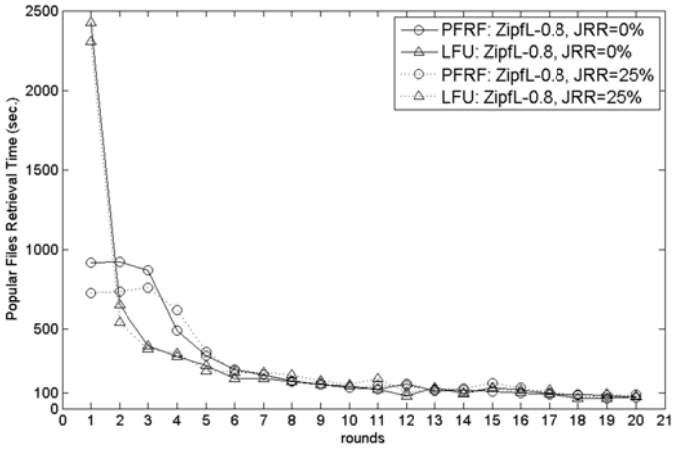



Fig. 21. Popular files retrieval time for PFRF and LFU on ZipfL-0.8 distribution with JRR=0% and 25%.

Table 5 Average job response delays of PFRF on ZipfL-0.8 with JRR=25% in twenty rounds

| Round | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Time (sec.) | 724.84 | 612.29 | 533.23 | 500.09 | 483.41 | 469.54 | 480.15 | 485.32 | 485.51 | 484.54 |
| Round | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Time (sec.) | 481.88 | 482.90 | 482.05 | 473.54 | 487.06 | 487.57 | 500.12 | 492.58 | 485.88 | 476.86 |

### 4.3.2 Average Bandwidth Cost Ratio (*ABCR*)

Figs. 22 to 26 illustrate the *ABCR*s of PFRF, LFU, NR case1, and NR case2 on the ten access patterns. The bandwidths of an inter-router link, a router-to-site link, and a GRC-to-router link as listed in Table 1 are respectively 10, 2.5, and 2.5, and the corresponding unit costs are respectively $\frac{1}{10}$, $\frac{1}{2.5}$, and $\frac{1}{2.5}$. With NR case1, $LC_c$ and $LFA_c$ in formula (8) are 0s since all master files are located in GRC with no replicas. $AFA_c$ is equal to $RFA_c$, $RC_c$ is 0.5 ($=\frac{1}{10}+\frac{1}{2.5}$), and $C_{baseline} =$

$0.6\left(=\frac{1}{10}+\frac{1}{10}+\frac{1}{2.5}\right)$. Thus, $BCR_c$ (NR case1) $=\frac{RC_c}{C_{baseline}}=0.834$, and *ABCR* (NR case1) $= BCR_c$. Therefore, *ABCR* (NR case1)s are all $0.834$ in all rounds on all access patterns.

For PFRF, LFU, and NR case2, $RC_c = C_{baseline} = 0.6$, and $LC_c = \frac{1}{2.5}$. Figs. 22 to 26 show PFRF's, LFU's, NR case1's, and NR case2's *ABCR*s on ZipfL-0.8, ZipfL-0.6, Geo-0.2, Geo-0.5, and Uniform, respectively. We can see that *ABCR*s of NR case2 in the five figures are all the worst, about 0.91 to 0.92. The reason is that NR case2 replicates remote files needed by jobs from remote clusters, even though *ART*s of NR case2 are shorter than those of NR case1 (not shown in Figs 16a to 20a, as stated above). Due to the change of file popularities at the end of phase 1 and phase 2, we can see that PFRF's *ABCR*s on ZipfL-0.8, ZipfL-0.6, and Geo-0.2 are lower than LFU's *ABCR*s. PFRF's and LFU's *ABCR*s on Geo-0.5 (see Fig. 25) are similar and the smallest compared to PFRF's and LFU's *ABCR*s on other access patterns. In other words, Geo-0.5 is the best access pattern. The reason has been stated above.

On Uniform with JRR=0% and 25% (see Fig. 26), PFRF's and LFU's *ABCR*s are similar since popularities of all files as shown is Fig. 15 are the same and without changing with time. According to formulas (8) and (9), the increase of $RFA_c$s will result in higher $BCR_c$s and *ABCR*s. It is also the reason why the bandwidth cost on Uniform is higher than those on the other access patterns. Moreover, although PFRF's *ART*s are shorter than those of LFU on Uniform, the bandwidths consumed by PFRF and LFU are similar from the whole system viewpoint.

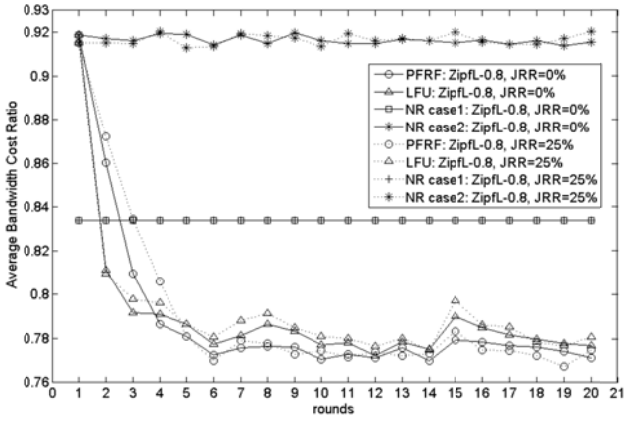



Fig. 22. Average Bandwidth Cost Ratios for PFRF, LFU, NR case1, and NR case2 on ZipfL-0.8 with JRR=0% and 25%.

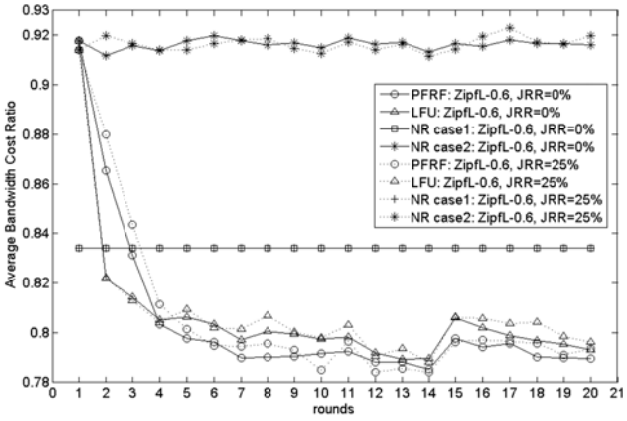



Fig. 23. Average Bandwidth Cost Ratios for PFRF, LFU, NR case1, and NR case2 on ZipfL-0.6 with JRR=0% and 25%.

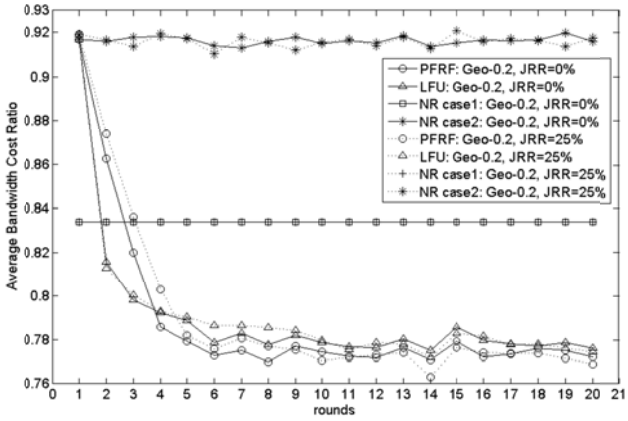



Fig. 24. Average Bandwidth Cost Ratios for PFRF, LFU, NR case1, and NR case2 on Geo-0.2 with JRR=0% and 25%.

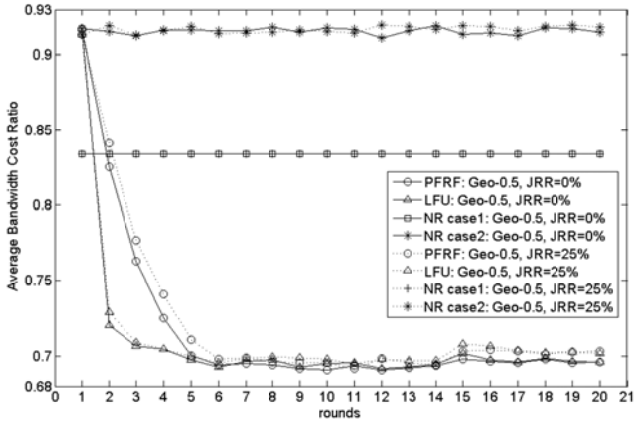



Fig. 25. Average Bandwidth Cost Ratios for PFRF, LFU, NR case1, and NR case2 on Geo-0.5 with JRR=0% and 25%.

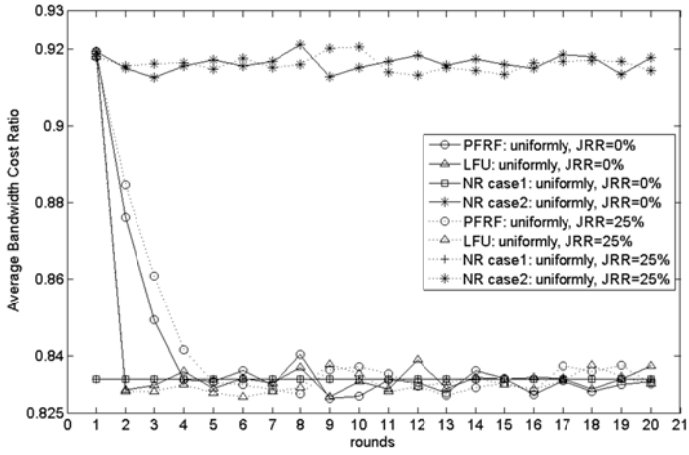



Fig. 26. Average Bandwidth Cost Ratios for PFRF, LFU, NR case1, and NR case2 on Uniform with JRR=0% and 25%.

### 4.3.3 Comparison of PFRF parameters a and b

The experimental environment used to evaluate the parameters a and b in formula (3) is also

the topology shown in Fig. 8 and the specifications listed in Table 1. But the file popularities do not change with time and each simulation is performed thirty times. Fig. 27 shows experimental results for PFRF's *ART*s when $a < b\ (a = 0.1,\ b = 0.15)$, $a = b\ (a = 0.1,\ b = 0.1)$, and $a > b\ (a = 0.15,\ b = 0.1)$ on ZipfL-0.8 with JRR=0%. Fig. 28 is the zoomed-in figure between round 6 and round 15 shown in Fig. 27. Evidently, the case when $a < b\ (a = 0.1,\ b = 0.15)$ has the best *ART*s, showing that it can accurately reflect which files are more popular. Therefore, in this study the case $a < b\ (a = 0.1,\ b = 0.15)$ is then selected.

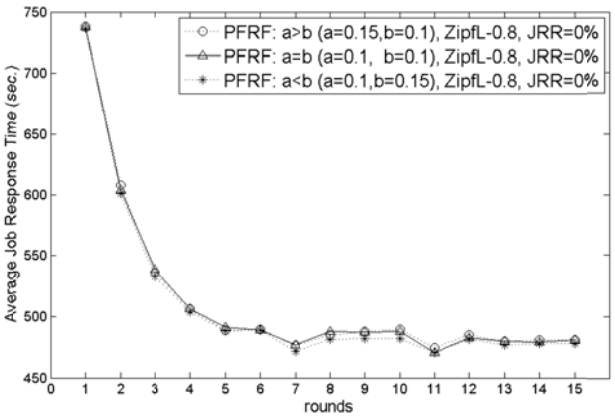



Fig. 27. The average job response time against different rounds.

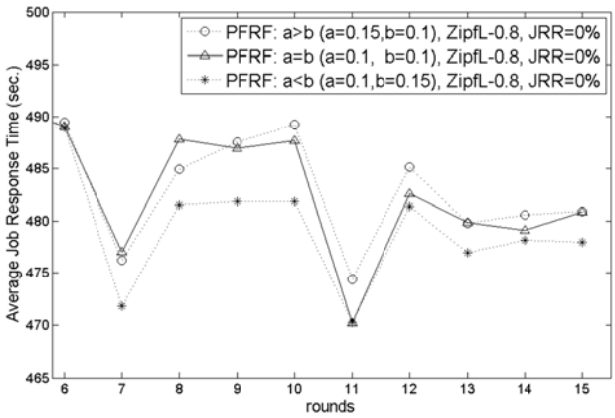



Fig. 28. The zoom-in figure between rounds 6 to 15 in Fig. 27.

### 4.3.4 Discussion

From an end user viewpoint, the primary goal of invoking a data replication algorithm is to shorten the average response time and data availability, and from the whole system viewpoint, the data replication algorithm should reduce bandwidth cost/consumption for Grid systems. From the simulation results, we can see that PFRF and LFU can reduce job response time and average bandwidth cost ratio and improve data availability. Although NR case2 can reduce job response time and improve data availability, it cannot reduce average bandwidth cost ratio. In most situations, PFRF outperforms LFU and NR case2 on the three given performance metrics since PFRF accurately predicts the popular files/replicas in each round based on data access history, and then replicates required files/replicas to appropriate sites. Furthermore, the influence of the file

popularity change on LFU is higher than that on PFRF.

Furthermore, the change of file popularity has a larger impact on LFU as compared with PFRF.

LFU's performance is generally similar to that of PFRF, but it performs replication frequently, consequently consuming a huge amount of storage, causing high workload and overhead for a Grid system. In the simulation, the average job response time does not include job replication time, so LFU is superficially optimal.

## 5. Conclusions

In this paper, we propose the PFRF data replication algorithm for a star-based Data Grid constrained on limited storage space to improve its file access performance. We have also instantiated three algorithms PFRF, LFU, and NR to create ten access patterns, and evaluate the performance of these algorithms on a simulation tool, GridSim. PFRF calculates popularity weights for files to predict which files will be frequently accessed by users in the next round. Whenever users request popular files that do not currently exist in local cluster, PFRF replicates these files to appropriate site/cluster. Nevertheless, if unpopular files which do not exist in local sites are requested by users, PFRF will not duplicate them to local sites, implying users have to remotely access them. In addition to average job response time and data availability, bandwidth cost ratio is also involved to evaluate the data replication algorithms. We also demonstrate that PFRF efficiently shortens the file access response time, increases data availability, and decreases bandwidth cost/consumption compared with those of LFU and NR algorithms, even though the users' file access behaviors change from time to time.

In the future, we will try to replicate files to users' local sites, instead of to users' current clusters. This can more efficiently reduce intra-cluster bandwidth consumption and unnecessary transmission time. Furthermore, we will develop a reliability model for sites to evaluate how many file replicas are required for a file so that the data reliability can fulfill the reliability requirement. We also plan

to validate our simulation results on real Data Grids so as to evaluate the proposed scheme on a real testbed. Those constitute our future research.

## 6. References


[1] EU Data Grid project, http://www-eu-egee.org/.

[2] Data grid, http://en.academic.ru/dic.nsf/enwiki/247172 [16 September 2010]

[3] B. Tierney, W. Johnston, J. Lee, and M. Thompson, "A Data Intensive Distributed Computing Architecture for Grid Applications," *Future Generation Computer Systems* vol. 16, no.5, pp. 473–481, 2000.

[4] BIRN, http://www.nbrin.net/.

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

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

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

[8] PPDG, http://www.ppdg.net.

[9] M. Lei, S. V. Vrbsky, and X. Hong, "An on-line replication strategy to increase availability in data grids," *Future Generation Computer Systems,* vol. 24, no. 2, pp. 85–98, 2008.

[10] O. Wolfson, S. Jajodia, and Y. Huang, "An adaptive data replication algorithm," *ACMTrans. Database Syst*, vol. 22, no.4, pp. 255–314, 1997.

[11] M. Rabinovich, I. Rabinovich, and R. Rajaraman, "Dynamic replication on the Internet," *Technical Report,* HA6177000–980305-01-TM, AT&T Labs, March 1998.

[12] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High Performance Data Grid," *Proceedings of the Second International Workshop on Grid Computing*, Denver, CO, pp. 75–86, November 2001.

[13] M. Tang, B.-S. Lee, C.-K. Yeo, X. Tang, "Dynamic replication algorithms for the multi-tier data grid," *Future Generation Computer Systems* vol. 21, pp. 775–790, 2005.

[14] R. S. Chang and H. P. Chang, "A dynamic data replication strategy using access-weights in

data grids," *The Journal of Supercomputing,* vol. 45, no. 3, pp. 277–295, 2008.

[15] S.Y. Ko, R. Morales, and I. Gupta, "New worker-centric scheduling strategies for data-intensive grid applications," *Proc. ACM/IFIP/USENIX Int'l Conference on Middleware*, pp. 121–142, 2007.

[16] L. Meyer, J. Annis, M. Wilde, M. Mattoso, and I. Foster, "Planning spatial workflows to optimize grid performance," *Proc. ACM Symp. Applied Computing*, pp. 786 –790, 2006.

[17] A. R. Abdurrab and T. Xie, "FIRE: A File Reunion Based Data Replication Strategy for Data Grids," *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp.215–223, 2010.

[18] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *In INFOCOM no.1*, New York, USA, pages 126–134, 21–25 March 1999.

[19] D.G. Cameron, R. Carvajal-Schiaffino, A. Paul Millar, C. Nicholson, K. Stockinger, and F. Zini, "Evaluating scheduling and replica optimisation strategies in OptorSim," *4th International Workshop on Grid Computing (Grid2003)*, Phoenix, Arizona, November 17, IEEE Computer Society Press, 2003.

[20] A. Sulistio, U.Cibej, S. Venugopal, B. Robic, and R. Buyya. "A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim," *Concurrency & Computation: Practice and Experience*, Wiley Press, New York, USA, 2008.

[21] K. Ranganathan and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications," *International Symposium for High Performance Distributed Computing (HPDC-11)*, Edinburgh, 2002.

[22] K. Ranganathan and I. Foster, "Simulation studies of computation and data scheduling algorithms for data grids," *J. Grid Comput*. vol. 1, pp. 53–62, 2003.

[23] K. Ranganathan, A. Iamnitchi, I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," *in: Proceedings of the Workshop*

*on Global and Peer-to-Peer Computing on Large Scale Distributed Systems*, Berlin, May 2002.

[24] F. Schintke, A. Reinefeld, Modeling replica availability in large Data Grids, *Journal of Grid Computing*, vol.1, no. 2, 2003.

[25] Peter Kunszt, Erwin Laure, Heinz Stockinger, Kurt Stockinger, "File-based replica management," *Future Generation Computer Systems Journal*, vol. 21, pp.115–123. 2005.

[26] R. S. Chang, J. S. Chang, and S. Y. Lin, "Job scheduling and data replication on data grids," *Future Generation Computer Systems,* vol. 23, no. 7, pp. 846-860, 2007.

[27] The MONARC project, http://monarc.web.cern.ch/MONARC/.

[28] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 7 ed.: Wiley, 2004.

[29] D. G. Cameron, R. C. Schiaffino, J. Ferguson, P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "OptorSim v2.0 Installation and User Guide," November 2004. http://edgwp2.web.cern.ch/edg-wp2/optimization/optorsim.html

[30] George Kingsley Zipf. Relative frequency as a determinant of phonetic change. Reprinted from the Harvard Studies in Clasical Philiology, Volume XL, 1929.

[31] C. T. Yang, T. F. Han, W. C. Shih, W. C. Chiang, and C. H. Chang, "Metropolitan-Scale Grid Environment: The Implementation and Applications of TIGER Grid," *Paper presented at the Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops*, Sorrento, Italy, Dec. 4–6, 2006.

[32] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien, "The MicroGrid: A scienti_c tool for modeling computational grids," *Proc. of IEEE Supercomputing Conference*, Dallas, USA, November 4–10 2000.

[33] W. Bell, D. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini, "Simulation of Dynamic Grid Replication Strategies in OptorSim," *Proc. of the 3rd International Workshop on Grid Computing (GRID)*, Baltimore, USA, 18 November, 2002.

[34] A. Legrand, L. Marchal, and H. Casanova, "Scheduling distributed applications: The SimGrid simulation framework," *Proc. of the 3rd International Symposium on Cluster Computing and*

*the Grid*, Tokyo, Japan, May 12–15 2003.

[35] C. Mihai Dobre and C. Stratan, "Monarc simulation framework," *Proc. of the RoEduNet International Conference*," Timisoara, Romania, May 27–28 2004.

[36] ChicSim - the Chicago Grid Simulator, http://people.cs.uchicago.edu/_krangana/ChicSim.html, 5 October 2007.