Future Generation Computer Systems 28 (2012) 1045–1057

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

PFRF: An adaptive data replication algorithm based on star-topology data grids

Ming-Chang Lee [a], Fang-Yie Leu [b], Ying-ping Chen [a,*]

[a] Department of Computer Science, National Chiao Tung University, Taiwan

[b] Department of Computer Science, Tunghai University, Taiwan

ARTICLE INFO

Article history:
Received 5 November 2010
Received in revised form
5 July 2011
Accepted 5 August 2011
Available online 6 November 2011

Keywords:
Data grid
Data replication
Data access patterns
File popularity
PFRF

ABSTRACT

Recently, data replication algorithms have been widely employed in data grids to replicate frequently accessed data to appropriate sites. The purposes are shortening file transmission distance and delivering files from nearby sites to local sites so as to improve data access performance and reduce bandwidth consumption. Some of the algorithms were designed based on unlimited storage. However, they might not be practical in real-world data grids since currently no system has infinite storage. Others were implemented on limited storage environments, but none of them considers data access patterns which reflect the changes of users' interests, and these are important parameters affecting file retrieval efficiency and bandwidth consumption. In this paper, we propose an adaptive data replication algorithm, called the Popular File Replicate First algorithm (PFRF for short), which is developed on a star-topology data grid with limited storage space based on aggregated information on previous file accesses. The PFRF periodically calculates file access popularity to track the variation of users' access behaviors, and then replicates popular files to appropriate sites to adapt to the variation. We employ several types of file access behaviors, including Zipf-like, geometric, and uniform distributions, to evaluate PFRF. The simulation results show that PFRF can effectively improve average job turnaround time, bandwidth consumption for data delivery, and data availability as compared with those of the tested algorithms.

1. Introduction

Generally, a data grid, a specific grid system that provides users with a huge amount of storage space, often maintains a high volume of distributed data to serve users. Many recent large-scale scientific systems [1–4] and commercial applications [5], e.g., the Biomedical Informatics Research Network (BIRN) [6], the Large Hadron Collider (LHC) [7], the DataGrid Project (EDG) [8], and physics data grids [9,10], have collected a huge amount of data and performed complex experiments and analyses on the data [11–13].

According to the Pareto principle (also known as the 80/20 rule) [14], a part of data grid files is frequently accessed and transferred. If a file has no replicas distributed over the data grid, the efficiency of accessing the file is often poor since long distance data transfer always occupies a lot of bandwidth and causes long transmission delays [15]. Hence, how to decrease data access latency, lower bandwidth consumption for data transmission, and improve data availability have been the key issues in data grid research [16]. Data replication is a general and simple approach to achieve these goals. It has been widely used in many areas, such as the Internet, peer-to-peer systems, and distributed databases [17–21]. A well-defined data replication method should meet the requirements of being able to determine an appropriate time to replicate files, decide which files should be replicated, and store these replicas in appropriate locations [15,16,22–24].

On the other hand, the analyses of data access patterns have been the critical steps in designing efficient dynamic data replication schemes [25–27]. 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 [28,29]. Breslau et al. [28] claimed that using the Zipf-like distribution can more accurately model the distribution of webpage accesses. Cameron et al. [29] showed that the distribution of file accesses in data grids follows the Zipf-like distribution. Ranganathan and Foster [22,30] claimed that the geometric distribution can properly model file access behaviors and the property of temporal/geographical locality.

Further, Ranganathan and Foster [31] derived file popularity by using both Zipf and geometric distributions on a multi-tier data grid with unlimited storage space. Tang et al. [23] also used Zipf-like and geometric distributions to simulate users' file access behaviors on a multi-tier data grid. Chang et al. [32,24] proposed two data replication strategies on a cluster-based data grid with limited storage. However, the strategy they proposed in [32] did

* Corresponding author.
E-mail addresses: mingchang1109@gmail.com (M.-C. Lee), leufy@thu.edu.tw (F.-Y. Leu), ypchen@nclab.tw, ypchen@cs.nctu.edu.tw (Y.-p. Chen).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.08.015

not consider the data access pattern. Hence, it might lead to inefficient data access as the users' access pattern changes; the strategy proposed in [24] only replicates the file most frequently accessed in the last time period, consequently resulting in long file transmission delays for those files with similar but low weights.

In this study, we propose an adaptive data replication algorithm, called the *P*opular *F*ile *R*eplicate *F*irst algorithm (PFRF for short), which is developed on a star-topology data grid with limited storage space. A star-topology data grid is a simplified tree-topology data grid with a central cluster that connects all other clusters. A link l between two arbitrary clusters will go through the central cluster, and l might comprise several routers, and physical links. Directly evaluating the components of l is difficult since too many analytical items might be involved. Hence, this study treats l as a logical link to simplify the original topology as a whole [33, 34]. The simplification process will be proposed. To adapt to the changes of users' interests in files, the PFRF aggregates file access information and replicates popular files to suitable clusters/sites. We simulate several cases in which file popularity follows a Zipf-like distribution, geometric distribution, and uniform distribution under the assumption that user behaviors vary with the changes of users' interests. The simulation results show that PFRF provides 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. Section 2 introduces background and related work of this study. Section 3 describes the architecture of a star-topology data grid and the details of the PFRF. Simulation results are presented 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 several existing replication strategies and algorithms.

2.1. Data grid architecture

Data grids can be classified into multi-tier data grids, first proposed by the MONARC project [35], and cluster data grids, initially introduced by Chang et al. [32]. The multi-tier data grid architecture in which a leaf node represents a user or a computational node, and internal nodes are resource sites keeping sharable files. In this architecture, a file held by a site will also be held by all its ancestor sites. Therefore, the root site holds all files stored in the data grid. When an end user requires a file F which does not exist in his/her site, the user requests F from its immediate ancestor. If the ancestor does not have the file, it in turn requests F from its immediate ancestor. The process repeats until the user obtains the file from a node which holds the file. After that, the file will be replicated to all the nodes on this requesting path following the reverse direction of the requests. It is clear that file access latency can be reduced in a multi-tier data grid, but it leads to higher storage cost since files will be redundantly stored in multiple locations.

A cluster data grid consists of n clusters connected by the Internet [24]. Files are 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 determines which file should be replicated by computing file weights. After that, the file with the highest weight will be replicated to clusters that need the file. Sites in these clusters can then locally and quickly retrieve the file. Compared with a multi-tier data grid, a cluster data grid consumes less storage to hold files.

2.2. Existing data replication algorithms/strategies

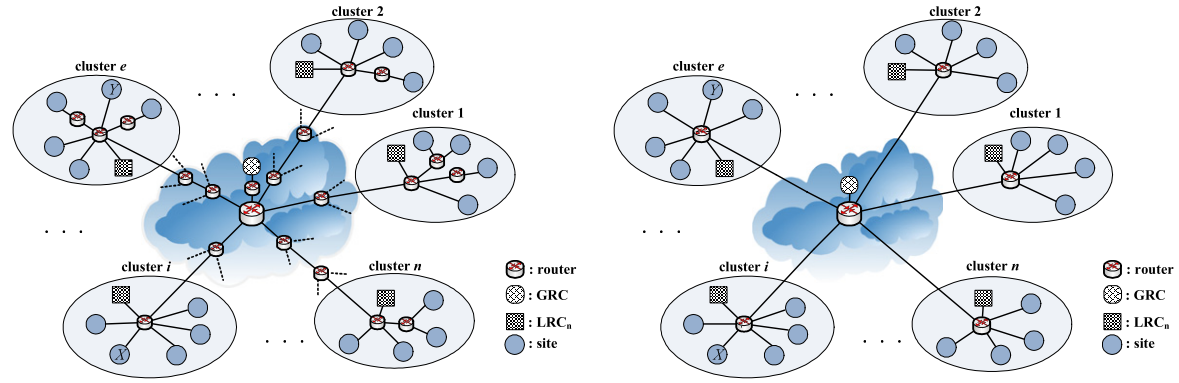
Least Frequently Used (LFU) [36] and Most Frequently Used (MFU) [36] are two simple dynamic replication strategies widely used in many areas, such as disk and cache memory duplication. If a storage device has insufficient space to hold a new file, LFU (MFU) will be invoked to choose the files that have been the least (most) frequently used as the victims to make room for the new one. In the experiments of this study, MFU and LFU are both involved, called the MFU/LFU strategy (M/LFU for short) in which MFU is used to choose the most frequently used files and LFU is employed to select victims once the destination cluster has insufficient storage space to save the replicated files.

Ranganathan and Foster [22] presented six replication/caching strategies for a multi-tier data grid: No Replication or Caching, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replication, and Fast Spread, and three types of localities: temporal locality, geographical locality, and spatial locality. The experimental results showed that the Fast Spread and Cascading Replication outperform the other four strategies and their file access latencies are shorter than those of the other four strategies. They also found that Fast Spread (Cascading) is better when the data access pattern is random (geographical locality). However, the six strategies cannot avoid the disadvantages of a multi-tier data grid, i.e., a file may be redundantly stored in a multi-tier. In fact, the storage space utilization and access latency are a trade-off [32]. Ranganathan and Foster [31] also proposed a suite of job scheduling and data replication algorithms for a multi-tier data grid and evaluated the performance of different combinations of the replication and scheduling strategies. One of the data replication algorithms, called DataRandom (DR for short), replicates a file when the corresponding access frequency exceeds a pre-defined threshold. Although DR is designed for an unlimited storage environment, it can also be run on a limited storage environment. DR is therefore involved in the experiments of this study.

Tang et al. [23] introduced Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU) algorithms to reduce the average data access response time for a multi-tier data grid. The basic idea of the two algorithms is to replicate a file to sites close to its requesting clients when the file's access rate is higher than a pre-defined threshold. SBU considers the file access history for individual site, but ABU aggregates the file access history for a system. With ABU, a node sends aggregated historical access records to its upper tiers, and the upper tiers do the same until these records reach the root. Due to the aggregation capability, ABU has a shorter job response time and less bandwidth consumption than those of SBU.

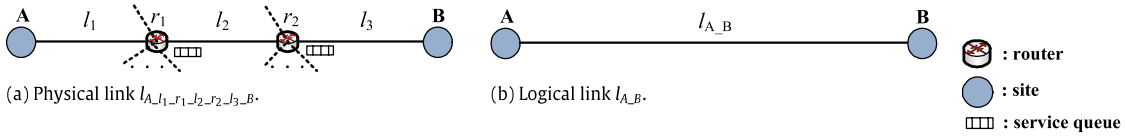
Khanli et al. [37] proposed an algorithm called Predictive Hierarchical Fast Spread (PHFS), which is an extended version of fast spread [22], in a multi-tier data grid. PHFS utilizes spatial locality [22,38] to predict data files required in the future, and pre-replicates these files to suitable sites to improve the performance of file accesses. Kunszt et al. [39] presented a replica management grid middleware to reduce file access/transfer time. Their experimental results showed that this middleware significantly reduces wide area transfer times. However, this model was developed for multi-tier data grids with unlimited storage space.

Chang et al. [24,32] presented two dynamic replication strategies, Latest Access Largest Weight (LALW) [24] and Hierarchical Replication Strategy (HRS) [32], on cluster-based data grids. LALW utilizes the half-life concept to evaluate file weights. A file with a higher access frequency has a larger weight. Their experimental results show that LALW outperforms LFU and no-replication data replication strategies [22] in network utilization and efficiency. However, LALW only replicates the most popular file in each time



(a) A tree-topology data grid. (b) A star-topology data grid.

Fig. 1. The architectures of a tree-topology data grid and a star-topology data grid.



(a) Physical link $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$. (b) Logical link l_{A_B}.

Fig. 2. (a) A physical link between site *A* and site *B*. (b) A logical link between site *A* and site *B*.

interval. Hence, the transmission delays of those files with similar but lower weights are still long. HRS, which is an extended version of BHR (Bandwidth Hierarchy based Replication) [40], aims to reduce expensive cluster-to-cluster replica transmission. Whenever the file is required, but cannot be retrieved from the local cluster, HRS replicates the file to a local site from a remote cluster. However, HRS does not consider data access patterns, and thus it might not be able to adapt to changes of user access behaviors and provide efficient data accesses.

3. System framework

The proposed data grid as shown in Fig. 1 consists of a *global replica controller* (GRC) and several clusters connected to the GRC through the Internet. As illustrated in Fig. 1(a), each connection comprises several routers and links, forming a tree-topology data grid rooted at the GRC. In other words, a file transmitted between two clusters will go through the routers and links on the two connections being considered. For example, when a site X in a cluster i issues a file access request to the GRC to request a file F stored in site Y which is a member of cluster e, the GRC then requests Y to deliver F to X. F then goes through routers and links between clusters i and e. Basically, it might not be easy to analyze the performance of the file transmission since in such a tree-topology too many network components and environmental parameters are involved. To simplify the evaluation of file transmission, in this study, we reduce a tree-topology data grid to a star-topology (see Fig. 1(b)).

3.1. Tree-to-star reduction

As shown in Fig. 2(a), site A connects to site B through link l_1, router r_1, link l_2, router r_2, and link l_3. The corresponding physical path is denoted by $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$. The bandwidth of the path is $\frac{|P|}{T_{l_1}+W_{r_1}+T_{l_2}+W_{r_2}+T_{l_3}}$ in which $|P|$ is the size of a packet P delivered through the path, $T_{l_i} = \frac{|P|}{B_{l_i}}$ is the transmission delays of link l_i, $W_{r_j} = \frac{1}{\mu_{r_j}-\lambda_{r_j}}$ is the service (queueing) delay of router r_j under the assumption that r_j is an M/M/1 queueing model where B_{l_i} is the bandwidth of l_i, $i = 1, 2$, and λ_{r_j} and μ_{r_j} are respectively the arrival rate and departure rate of r_j, $j = 1, 2, 3$. l_{A_B} shown in Fig. 2(b) is the logical link of $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$. Its bandwidth is $\frac{|P|}{T_{A\_B}}$ where T_{A_B} is the transmission delay of P from A to B. It is clear that $T_{A\_B} = T_{l_1} + W_{r_1} + T_{l_2} + W_{r_2} + T_{l_3}$. Therefore, we can conclude that $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$ can be reduced to a logical link l_{A_B}, and the tree-topology data grid shown in Fig. 1(a) could be reduced to a star-topology data grid shown in Fig. 1(b). Now a star-topology with the performance equivalent to that of the tree-topology can be obtained.

3.2. GRC and LRC

In the proposed architecture, each cluster comprises sites and a local replica controller (LRC). They are connected by a LAN or LANs. The LRC maintains a local replica table (*LRT*) includes filename, file location, access count, file weight, and master file fields, to record file access information. A master file is an original file that cannot be deleted from the data grid. File weight is the popularity of the file. Its calculation will be described later. File access count shows the frequency that the file is accessed by sites within a cluster. All master files are distributed to sites of different clusters. The GRC as a central server located somewhere in the Internet is responsible for aggregating file access records for all clusters and determining which files should be replicated to which clusters. The GRC maintains a global replica table (*GRT*) to collect the information recorded in *LRT*s. When the GRC decides to replicate a file to a cluster, it records the location of the new replica in *GRT* so that some time later when a LRC requests the location of the file, it can answer the LRC accordingly. Similarly, the cluster holding this new replica will record the related information in its *LRT*.

3.3. Zipf-like distribution and geometric distribution

To achieve a better file access performance, we need to keep track of users' file access behaviors to accordingly predict which files will be accessed frequently in the near future. The prediction is a main task of a data replication algorithm/strategy based on the assumption of temporal locality [22] in which a popular

PFRF algorithm:
Input: The total access count of f_i, $i = 1, 2, \ldots, N_k$, where N_k is the number of files that cluster c has in round r;
Output: Duplicating files that should be replicated to cluster c, denoted by $F_c = \{f_1, f_2, \ldots, f_{N_f}\}$, to cluster c;
Procedure:

```
1:  for each cluster c, c = 1, 2, ..., Nc {   /*Nc is the number of clusters in the particular data grid*/
2:  Aggregate A_c^r(f_i), i = 1, 2, ..., N_k;
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 LRT_C;
6:  Calculate PW_c^r(f_i) and PW_avg^r(f_i), i = 1, 2, ..., N_k;   /* Eqs. (3) and (4)*/
7:  Sort the set S according to the PW_avg^r(f_i);
8:  Set the value of x = 0.8, 0 < x < 1;   /*according to the 80/20 rule*/
9:  Calculate N_f;   /*N_f, derived with Eq. (5) for 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
11:                                                                    candidates*/
12: for each file f_j in S'{
13:     check LRC_C to see whether cluster c has f_j;
14:     if cluster c does not have f_j {
15:         if any site of cluster c has sufficient storage to save f_j
16:             replicate f_j to the site from a nearest cluster which has f_j;
17:         else if{ for each site Y in cluster c { /* check storage space of each site Y in cluster c*/
18:                     for each file f_k kept in site Y;
19:                     compare PW_avg^r(f_k) with PW_avg^r(f_j);
20:                     if there are v files that are less popular than f_j and the total size of these v
21:                       files plus the remaining storage space of site Y is sufficient to keep f_j
22:                       { PFRF deletes these v files and replicates f_j to Y;
23:                         break;}}}
24:         else /*The total size of all files that are less popular than f_j plus the remaining storage
25:                 space is insufficient to hold f_j*/
26:         {   print ("no sites in cluster c can keep f_j"); /*PFRF will not replicate f_j*/
27: }}}}
```

Fig. 3. PFRF data replication algorithm.

file will be accessed more frequently than unpopular ones [23]. Breslau et al. [28] showed that webpage requests follow a Zipf-like distribution [29,41] derived from Zipf's law [42]. In the Zipf-like distribution, the access probability of the i-th most popular file, denoted by $P(f_i)$, is

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

where $i = 1, 2, \ldots, n$ and α is a factor determining the file access distribution, $0 \leq \alpha < 1$.

Ranganathan and Foster [30,31] adopted the geometric distribution to simulate file popularity in which the access probability of the i-th most popular file, denoted by $P(i)$, is

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

where $i = 1, 2, \ldots, n$ and $0 < p < 1$. A larger p represents that a smaller portion of files has been frequently accessed. As stated above, we assume that our users' access behaviors follow either Zipf-like or geometric distributions with different parameters.

3.4. Popular file replicate first (PFRF) algorithm

The PFRF algorithm, as illustrated in Fig. 3, is performed by the GRC at the end of a round, where a round is a fixed time period T_d in which y jobs, $y \geq 0$, are submitted by users from each cluster. A job might require several files as its input data. 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 f_i stored in cluster c at round r, denoted by $A_c^r(f_i)$, sorts all the files on $A_c^r(f_i)$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 TNF_c^r, based on the information stored in LRT_C. Note that $1 \leq i \leq N_k$, and $1 \leq c \leq N_c$ where N_k is the number of files in cluster c in round r, and N_c is the number of clusters that the data grid has, and $r = 1, 2, 3, \ldots$.
2. File popularity calculation phase: In line 6, PFRF calculates a popularity weight for file f_i, denoted by $PW_c^r(f_i)$,

$$PW_c^r(f_i) = \begin{cases} PW_c^{r-1}(f_i) + A_c^r(f_i) \cdot a, & \text{if } A_c^r(f_i) > 0 \\ PW_c^{r-1}(f_i) - b, & \text{otherwise} \end{cases},$$
$$r \geq 1,\ c \geq 1, i \geq 1. \tag{3}$$

where a and b are constants and $a < b$. The reason why $a < b$ is described later. If $A_c^r(f_i) > 0$, i.e., f_i has been accessed by users in round r, PFRF increases $PW_c^{r-1}(f_i)$ by $A_c^r(f_i) \cdot a$. Otherwise, it decreases $PW_c^{r-1}(f_i)$ by b. Basically, a higher $PW_c^r(f_i)$ implies that f_i is more popular. We assume that in round 0 all files follow the binomial distribution, i.e., $PW_c^0(f_i) = 0.5$, which means that the initial access probability of f_i is 0.5. Note that the minimum value of each $PW_c^{r-1}(f_i)$ is 0. From previous access records of f_i, PFRF derives the variation of the popularity of f_i and predicts the popularity of f_i for the next round, where $1 \leq i \leq N_k$. For instance, if f_3 has been accessed 5 times by cluster 2 in round 1, $PW_2^1(f_3) = 0.5 + 5 \cdot a$. After the derivation and prediction, PFRF calculates the average popularity of the files in all clusters, denoted by $PW_{avg}^r(f_i)$,

$$PW_{avg}^r(f_i) = \frac{\sum_{k=1}^{N_q} PW_c^r(f_i)}{N_q} \tag{4}$$

where N_q is the total number of clusters holding f_i in the 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

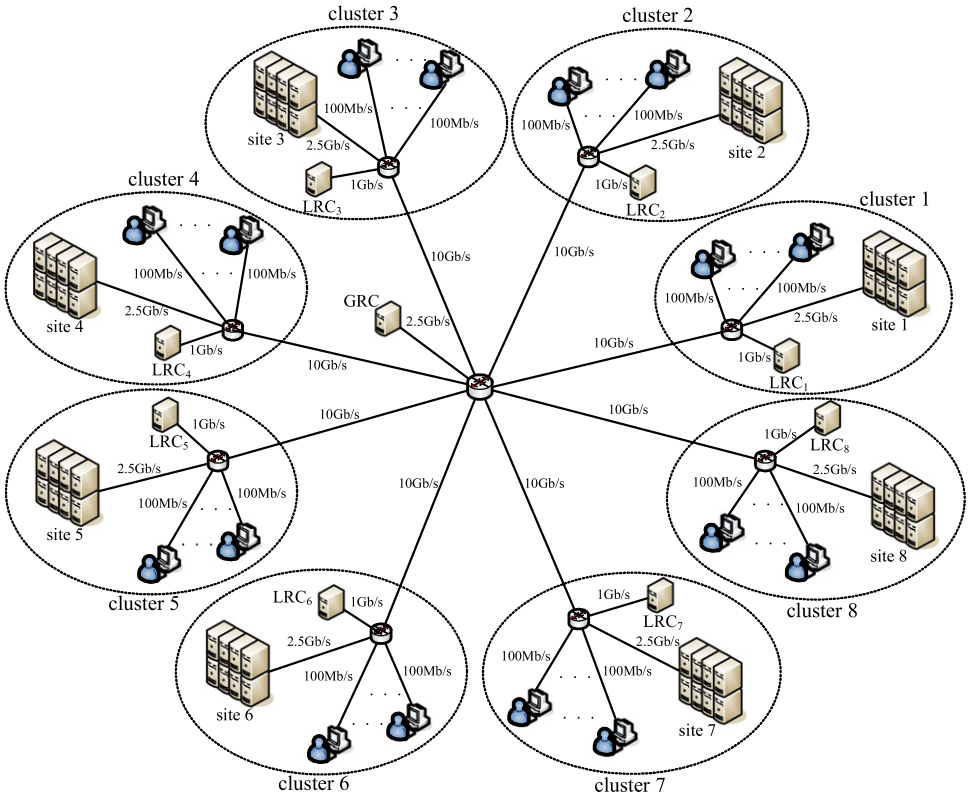


Fig. 4. The simulation topology.

selects the first N_f files as cluster c's replication candidates from S, where

$$N_f = \lfloor TNF_c^r \cdot (1 - x) \rfloor \quad (5)$$

in which x is a constant, $0 < x < 1$. In this study, we use the 80/20 rule as an example, i.e., PFRF will replicate the top 20% of frequently accessed files in this phase. Therefore, x is set to 0.8. If different rules or principles are employed, x may be changed.

4. File replication phase: Between lines 12 and 27, PFRF first checks to see whether each file, e.g., file f_j, in cluster c's replication candidates is stored in cluster c or not. If yes, PFRF does nothing. Otherwise, it further checks to see whether a site in cluster c has sufficient storage space to accommodate f_j or not. If yes, PFRF replicates f_j to the site from a nearest cluster holding f_j. Otherwise, PFRF deletes v files ($v \geq 1$) that are less popular than f_j from a site of cluster c so that it has enough storage space to keep f_j.

Since files and users may be located at different sites in the same cluster or different clusters in a star-topology data grid, a file transmission might go across clusters, consequently raising transmission delays. With the PFRF, popular files can be properly replicated and inter-cluster access can be dramatically reduced.

4. Simulation and performance comparison

To evaluate the proposed scheme, a real grid testbed or a grid simulator is required. However, to establish and maintain a real testbed is expensive. Instead, we choose a grid simulator as the simulation tool. Many grid simulators have been introduced, such as GridSim [43], MicroGrid [44], OptorSim [45], SimGrid [46], MONARC [47], and ChicSim [48], among which GridSim is chosen since it provides a flexible and extensible simulation environment and allows researchers to add new components/functions. In the following simulation, the existing data replication algorithms including M/LFU (recall MFU/LFU), DR (recall DataRandom), and No Replication (NR for short) are implemented and compared with the PFRF on a star-topology data grid.

4.1. Experimental environment and parameters

The test environment illustrated in Fig. 4 consists of a GRC and eight clusters. Each cluster has a LRC. The resource specifications and job parameters are listed in Tables 1 and 2, respectively. Each site comprises six computers, and each computer has four processors, i.e., a cluster has 24 processors. The processor rate of a processor is 1600 MIPS, i.e., the total processor rate of a cluster is 38,400 (=1600 × 24) MIPS. A cluster has 75 GB storage space to accommodate files. The inter-router, router-to-site, user-to-router, GRC-to-router, and LRC-to-router bandwidths are 10 Gb/s, 2.5 Gb/s, 100 Mb/s, 2.5 Gb/s, and 1 Gb/s, respectively. A total of 200 master files, each 1 GB in size, are randomly stored in this environment. A job randomly requests 5–10 files as its input files. On average, 10 jobs are submitted by each cluster in each round, i.e., on average a total of 80 jobs are submitted by all clusters in each round, and the time period of a round T_d is 1600 s.

In round r, M/LFU replicates cluster's most frequently used files to a cluster c in descending order. The replication is performed one by one until exhausting c's storage space. When c has insufficient storage space to replicate a remote file F, M/LFU selects k local files of c, denoted by $v = \{f_{c_1}, f_{c_2}, \ldots, f_{c_k}\}$, as the victims in the situation where the access count of f_{c_i} is smaller than that of F and k is the smallest integer that satisfies $\sum_{i=1}^{k} f_{c_i} \geq |F|$, where $|F|$ is the size of F. After that, M/LFU deletes the k files and replicates F to c.

We implemented two types of DR. One is DR-Local, in which a replication threshold of cluster c is set to the average access counts of all the files stored in c. For example, c has k files which have been accessed a total of m times in round r. The replication threshold is $\frac{m}{k}$. When a file, e.g., F, that is not stored in c has a

Table 1
Resources specifications of the following experiments.

Resources	Value
Total number of clusters	8
Total number of processors in a cluster	24
Single processor rate (MIPS)	1600
Total processor rate of a cluster	38,400 (=1600 × 24)
Storage available in a cluster	75 GB
Inter-router bandwidth	10 Gb/s
Router-to-site bandwidth	2.5 Gb/s
User-to-router bandwidth	100 Mb/s
GRC-to-router bandwidth	2.5 Gb/s
LRC-to-router bandwidth	1 Gb/s

Table 2
Job parameters of the following experiments.

Job parameters	Value
Total number of master files	200
Size of a master file	1 GB
Average number of jobs submitted by each cluster in a round	10
Average number of jobs submitted by all clusters in each round	80 (8 × 10 jobs)
Number of files accessed by a job	5–10
The duration of a round (T_d)	1600 s

higher access count than $\frac{m}{k}$ in round r, F will be replicated to c at the end of r, $1 \leq c \leq 8$. The other is DR-Global, in which the replication threshold is set to the average access counts of all files in all clusters, i.e., if the 200 files have been accessed h times in round r, the replication threshold will be $\frac{h}{200}$ for all clusters. When F has been accessed at least $\frac{h}{200}$ times by c in round r, it will be replicated to c. If c has insufficient space to hold the file, DR-Local and DR-Global will delete the files that have been least frequently accessed from c to make room for F.

Two types of NR were also implemented, denoted by NR-GRC and NR-LRC. In the NR-GRC, the 200 master files are all stored in the GRC. When a job requires a file, it remotely accesses the file from the GRC without storing the file locally. In the NR-LRC, the 200 master files are randomly distributed to the eight clusters. The GRC only maintains the *GRT*. A job locally accesses a file if the file is locally available. When it requires a remote file, it has to consult the GRC for the file location, and then remotely access the file. Due to duplicating no files, files are only stored in fixed sites and fixed clusters. Table 3 summarizes the master file settings for the tested algorithms.

4.2. Access patterns

Five access patterns listed in Table 4 were employed to simulate user access behaviors. File popularities follow Zipf-like (ZipfL for short), geometric (Geo for short), and uniform distributions (Uniform for short) where a uniform distribution represents that the probability of accessing a file by each user is the same. JRR, standing for job repeating rate, of round r is the probability of re-accessing those files that have been accessed in round $r-1$, $0 \leq JRR \leq 1$, and parameters α and p are respectively used when ZipfL and Geo are employed.

Table 4
Different data access patterns employed.

No.	Data access pattern	α/p	JRR (%)	$p(f_i)/p(i)$
1	ZipfL-0.8	0.8	25	$1/i^{0.8}$
2	ZipfL-0.6	0.6	25	$1/i^{0.6}$
3	Geo-0.2	0.2	25	$(1-0.2)^{i-1} \cdot 0.2$
4	Geo-0.5	0.5	25	$(1-0.5)^{i-1} \cdot 0.5$
5	Uniform	None	25	None

To effectively analyze the algorithms, we evenly partitioned the popularities of the 200 master files into 10 levels and divided twenty consecutive rounds into three phases. As listed in Table 5, the first, the second, and the third phase respectively contain rounds 1–7, 8–14, and 15–20. In the first phase, we assume that File0–File19 are the most popular files, i.e., belonging to the first popularity level. File20–File39 are the second popular files, thus belonging to the second popularity level, and so on. In the second phase, we swap the files of the first two popularity levels, i.e., File20–File39 become the most popular, File0–File19 become the second, to simulate the change of file popularities, and other files' popularities remain unchanged. In the third phase, File40–File59 are the most popular files, File20–File39 the second, and File0–File19 the third. Other levels' file popularities remain unchanged.

To better understand the behaviors of data access patterns, i.e., file popularities, over the three phases, we first conduct the following experiments: 2000 jobs, instead of 80 jobs, were submitted for file accesses in each phase. The experimental results of ZipfL-0.8, ZipfL-0.6, Geo-0.2, Geo-0.5, and Uniform are illustrated in Figs. 5–9, respectively. Fig. 5(a) shows users' file access behaviors in the first phase; Fig. 5(b) and Fig. 5(c) respectively plot those in the second and the third phases. The access count (AC) of each most popular file in all the three phases/figures is about 215, and unpopular files, i.e., File60 to File199, are accessed less. In the case of ZipfL-0.6 (see Fig. 6), 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, are not evidently different, like a uniform distribution, in all three phases. When Geo-0.2 is invoked (see Fig. 7), the AC of each most popular file is about 175. When file IDs increase, the ACs decline more sharply than those in ZipfL-0.6 and ZipfL-0.8. In the Geo-0.5 case (see Fig. 8), the difference between/among the most popular files' ACs and those of the second and the third popular files in 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 File59 are frequently accessed, while accesses of File140 to File199 are rare. The difference among the ACs of different popularity levels on the Uniform as shown in Fig. 9 is insignificant.

4.3. Simulation results

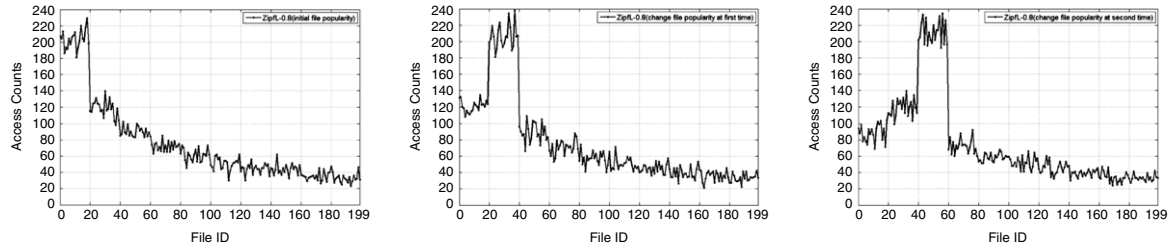
The tested algorithms were run on the same experimental environment so their performance can be fairly compared. Several

Table 3
Master files settings for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC algorithms.

No.	Data replication algorithm	Setting
1	PFRF	200 master files are randomly distributed to the eight clusters
2	M/LFU	"
3	DR-Local	"
4	DR-Global	"
5	NR-GRC	200 master files are all stored in the GRC
6	NR-LRC	200 master files are randomly distributed to the eight clusters

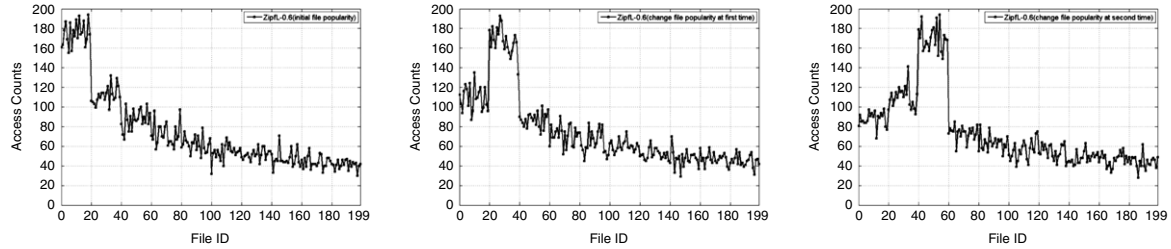
Table 5
The file popularities in the three phases (Rounds 1–7, 8–14, and 15–20).

Popularity level (files)	Phases		
	Phase 1 (rounds 1–7)	Phase 2 (rounds 8–14)	Phase 3 (rounds 15–20)
1 (File0–File19)	1st popular	2nd popular	3rd popular
2 (File20–File39)	2nd popular	1st popular	2nd popular
3 (File40–File59)	3rd popular	3rd popular	1st popular
…	…	…	…
10 (File180–File199)	10th popular	10th popular	10th popular



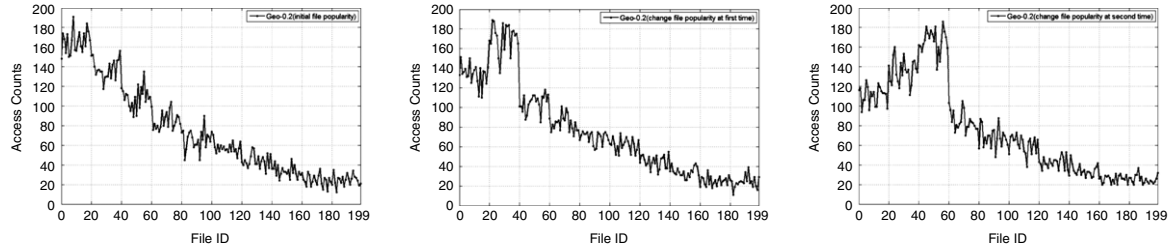
(a) Phase 1. (b) Phase 2. (c) Phase 3.

Fig. 5. Distributions of file requests against the 200 master files in the simulation process on ZipfL-0.8.



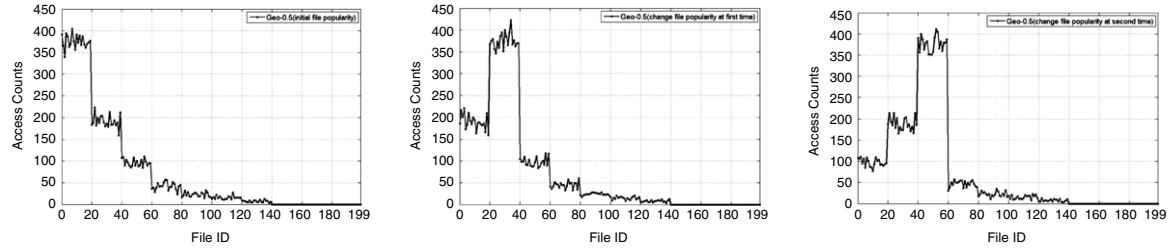
(a) Phase 1. (b) Phase 2. (c) Phase 3.

Fig. 6. Distributions of file requests against the 200 master files in the simulation process on ZipfL-0.6.



(a) Phase 1. (b) Phase 2. (c) Phase 3.

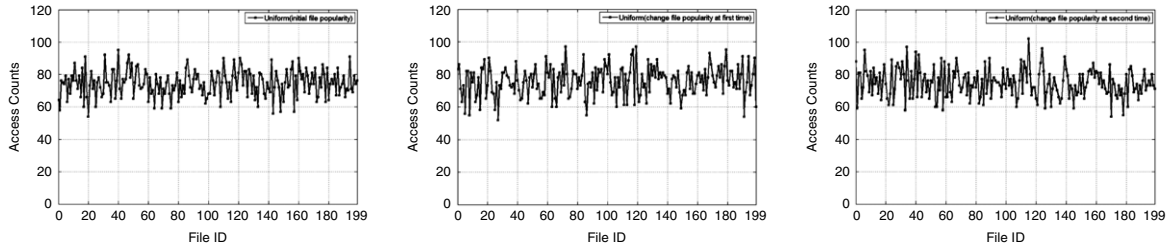
Fig. 7. Distributions of file requests against the 200 master files in the simulation process on Geo-0.2.



(a) Phase 1. (b) Phase 2. (c) Phase 3.

Fig. 8. Distributions of file requests against the 200 master files in the simulation process on Geo-0.5.

test metrics were used. The first is *average job turnaround time* (*ATT*), which is an average 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 received by the job. *ATT* is derived by dividing the total turnaround time of all jobs in all clusters in round r by the total number of jobs. The second is average



(a) Phase 1. (b) Phase 2. (c) Phase 3.

Fig. 9. Distributions of file requests against the 200 master files in the simulation process on Uniform.

data availability (*ADA*). Data availability, which was proposed by GridSim [43] for a job, e.g., job_x, in cluster c to access a file f_i of size d_{x_i}, denoted by $Avail_{x,c}$, is defined as

$$Avail_{x,c} = \frac{\sum_{i=1}^{k} t_{x_i,c}}{\sum_{i=1}^{k} d_{x_i}} \tag{6}$$

where k is the number of files accessed by job_x running on cluster c in round r, and $t_{x_i,c}$ is the time that job_x consumes to acquire f_i, locally or remotely. Let $avgAvail_c$ be the average data availability of cluster c which is defined as

$$avgAvail_c = \frac{\sum_{x \in jobs_c} Avail_{x,c}}{|jobs_c|} \tag{7}$$

where $jobs_c$ is the set of jobs submitted to cluster c by users to access files. The *ADA* of all clusters is defined as

$$ADA = \frac{\sum_{m=1}^{N_c} avgAvail_m}{N_c} \tag{8}$$

where N_c is the number of clusters in the data grid. The last is average bandwidth cost ratio (*ABCR*). *Bandwidth cost ratio* of cluster c in a round, denoted by BCR_c, is defined as

$$BCR_c = \frac{LFA_c \cdot LC_c + RFA_c \cdot RC_c}{AFA_c \cdot C_{baseline}} = \frac{LFA_c \cdot LC_c + RFA_c \cdot RC_c}{(LFA_c + RFA_c) \cdot C_{baseline}} \tag{9}$$

where LFA_c (RFA_c) is the number of files that users in cluster c can locally (should remotely) access, $AFA_c = LFA_c + RFA_c$, LC_c is the cost for a user in cluster c to locally access a file. RC_c is the cost for the user to access a file from a remote cluster or the GRC, and $C_{baseline}$ is the average cost for a user to access a file from a remote cluster to local cluster c. If LFA_c is larger than RFA_c, that implies the particular data replication algorithm can more accurately predict user access behaviors. Otherwise, the algorithm due to inaccurate prediction would consume a lot of network resources to remotely access files. Eq. (9) only involves the number of files and neglects file sizes since in the simulation all files are of the same size, i.e., 1 GB. The *ABCR* used to determine whether a data replication algorithm could accurately predict popular files or not is defined as

$$ABCR = \frac{\sum_{m=1}^{N_c} BCR_m}{N_c} \tag{10}$$

where N_c is the total number of clusters in the data grid. A data replication algorithm with a smaller *ABCR* value will lead to better grid performance since most data can be locally retrieved. In the following, each simulation was performed ten times to obtain the values of the three performance metrics.

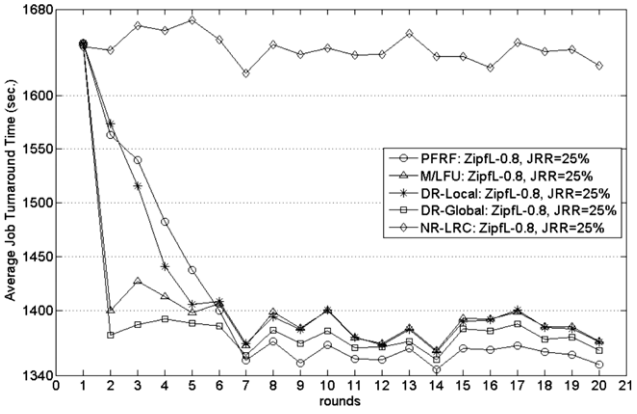


Fig. 10. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.8 with JRR = 25%.

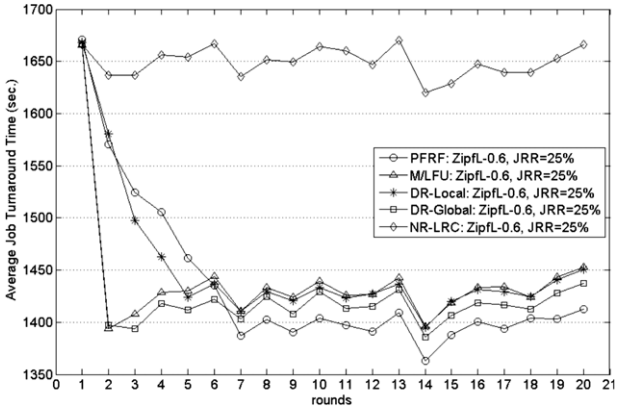


Fig. 11. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.6 with JRR = 25%.

4.3.1. Average job turnaround time (ATT) and average data availability (ADA)

Fig. 10 show the experimental results of *ATT*s for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on access pattern ZipfL-0.8 with JRR = 25%. The results of ZipfL-0.6, Geo-0.2, Geo-0.5, and Uniform are illustrated in Figs. 11–14, respectively. When ZipfL-0.8 with JRR = 25% is employed, as shown in Fig. 10, PFRF's *ATT*s are shorter than those of M/LFU, DR-Local, and DR-Global after the sixth round. This is also true on ZipfL-0.6 and Geo-0.2 (see Figs. 11 and 12, respectively). Fig. 13 plots the experimental results of Geo-0.5. On Uniform with JRR = 25% as shown in Fig. 14, *ATT*s of PFRF, M/LFU, DR-Local, and DR-Global are longer than those shown in Figs. 10–13 since the tested algorithms cannot effectively discriminate file popularity. It is clear that PFRF has shorter *ATT*s than those of M/LFU, DR-Local, and DR-Global on all of ZipfL-0.8, ZipfL-0.6, and Geo-0.2.

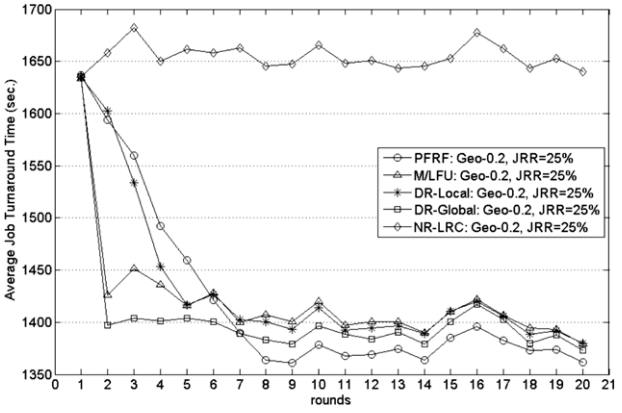


Fig. 12. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on Geo-0.2 with JRR = 25%.

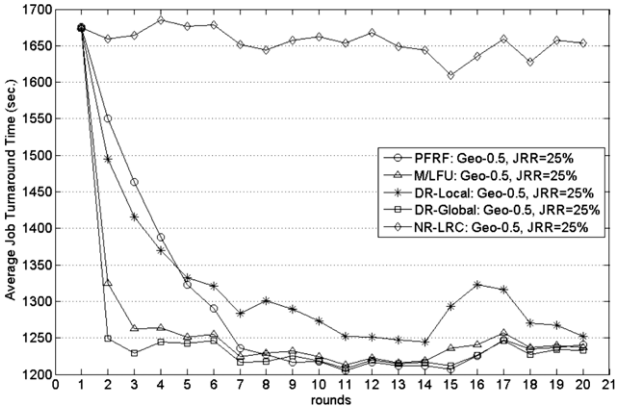


Fig. 13. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on Geo-0.5 with JRR = 25%.

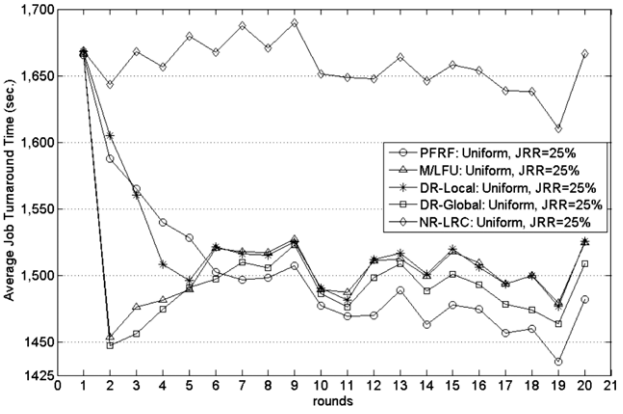


Fig. 14. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on Uniform with JRR = 25%.

When the data access pattern is Geo-0.5 with 25% (see Fig. 13), PFRF, M/LFU, and DR-Global have similar *ATT*s since the popular files shown in Fig. 8 can be easily identified. This is also why these algorithms on Geo-0.5 have the shortest *ATT*s compared with *ATT*s of these algorithms on other data access patterns (see scale of *Y*-axis of Figs. 10–14). Note that, on Geo-0.5, DR-Local does not effectively keep up with the change of user access behaviors especially at the end of phase 1 and phase 2. The reason is that it keeps accumulating the access count of certain popular files, resulting in a higher average access count as the replication threshold. It is impossible for a rising-popularity file, e.g., *F*, to have its access count suddenly larger than the threshold. Hence, *F* will not be replicated.

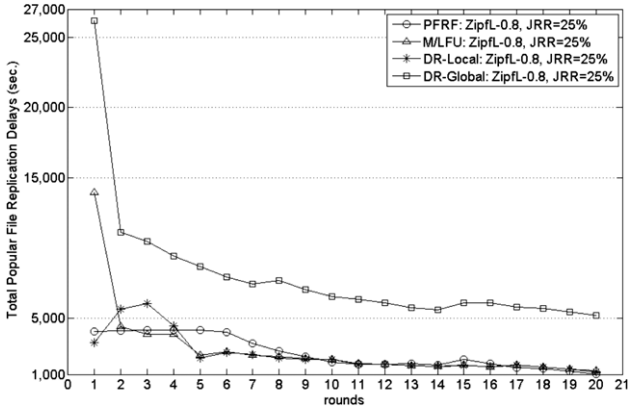


Fig. 15. Total popular file replication delays for PFRF, M/LFU, DR-Global, and DR-Local on ZipfL-0.8 with JRR = 25%.

*ATT*s of PFRF on ZipfL-0.8, ZipfL-0.6, and Geo-0.2 in the twenty rounds are shown in Figs. 10–12, in which when file popularities change, i.e., between rounds 7 and 8 and between rounds 14 and 15 according to Table 5, *ATT*s of PFRF increase less sharply than those of other algorithms, implying that the PFRF can quickly adapt to the change of user access behaviors and provide better access performance as compared with all the other algorithms.

We have not mentioned NR-GRC and NR-LRC since NR-LRC's plots are high above those of the other four access patterns (see Figs. 10–14), and NR-GRC leads to a longer *ATT* (4714 s in average) which is about three times the highest scale of each figure. If we plot the results of NR-GRC in Figs. 10–14, the *ATT*s of the other algorithms will be close to each other and cannot be discriminated. The cause of longer *ATT*s for NR-GRC is that because there are no local files, all jobs have to remotely access all required files from the GRC. On NR-LRC, a job in each round spent about 1625–1675 s (see Figs. 10–14). Apparently, NR's access performance is not better than those of the other four algorithms on all access patterns.

Fig. 15 shows the total popular file replication delays of the PFRF, M/LFU, DR-Local, and DR-Global on ZipfL-0.8 with JRR = 25%. Note that NR does not replicate files. Hence, its experimental results are absent from this figure. Due to the page limit, we omit the total popular file replication delays of these algorithms on the other access patterns since they are similar to those illustrated in Fig. 15. According to the 80/20 rule, PFRF only replicates the top 20% of popular files, and thus PFRF always spends less than 5000 s to replicate popular files from remote clusters to local sites in each round. For each cluster, due to the number of remote files becoming less over time, *ATT*s of PFRF as shown in Figs. 10–14 reduce gradually. M/LFU replicates required files from remote clusters in each round, consequently, like that of PFRF, taking a longer time (about 14,000 s) to replicate files in the first round, and spending less in the later rounds.

After the fourth rounds, the total popular file replication delays of DR-Local are almost the same as those of M/LFU. Thus, *ATT*s of DR-Local are then similar to those of M/LFU on all data access patterns except on Geo-0.5. That is why in Figs. 10–12 and 14, the two curves almost overlap. DR-Global results in longer file replication delays than those of other algorithms and the delays are always higher than 5000 s in each round since it keeps replicating files in each round. That is why *ATT*s of DR-Global as shown in Figs. 10–14 can remarkably decline after the first round. However, *ATT*s of DR-Global are longer than those of PFRF after the sixth round in all access patterns except on Geo-0.5. The reason has been stated above.

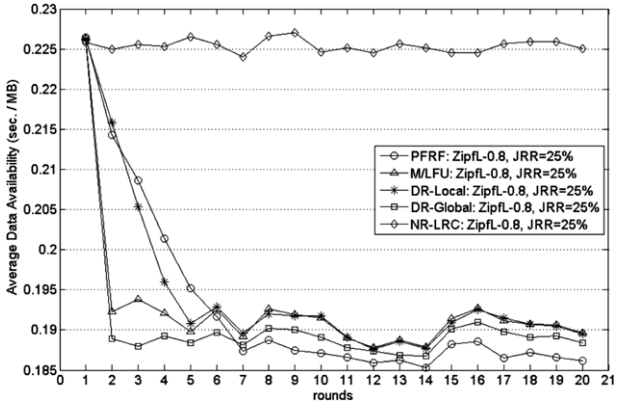


Fig. 16. Average data availabilities for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.8 with JRR = 25%.

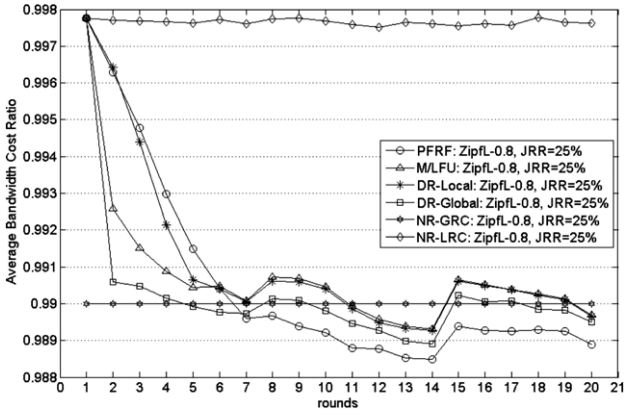


Fig. 17. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on ZipfL-0.8 with JRR = 25%.

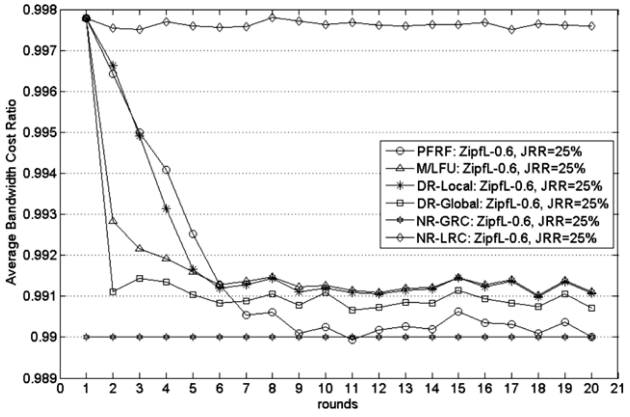


Fig. 18. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on ZipfL-0.6 with JRR = 25%.

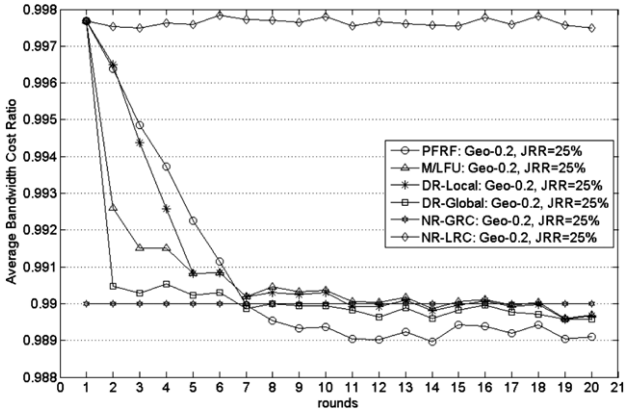


Fig. 19. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on Geo-0.2 with JRR = 25%.

Fig. 16 illustrates the *ADA*s for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.8 with JRR = 25%. According to the definition of $Avail_{x,c}$ presented in Eq. (6), the numerator $t_{x_i,c}$ is the turnaround time of job_x in cluster c, and hence the *ADA*s of all the tested algorithms have similar trends to that of their *ATT*s on all data access patterns. For example, all curves in Fig. 10 are similar to those in Fig. 16 on ZipfL-0.8 with JRR = 25%. Note that the *ADA*s of NR-GRC in each round are also omitted from Fig. 16 because they are the worst (about 0.637 which is far above the top scale, 0.23, of Fig. 16). The cause of these high *ADA*s was mentioned above.

4.3.2. Average bandwidth cost ratio (ABCR)

Fig. 17 illustrates the *ABCR*s of PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on ZipfL-0.8 with JRR = 25%. The bandwidths of the inter-router link, router-to-site link, user-to-router link, and GRC-to-router link as listed in Table 1 are respectively 10, 2.5, 0.1, and 2.5 Gb/s, and the corresponding unit costs are respectively $\frac{1}{10}$, $\frac{1}{2.5}$, $\frac{1}{0.1}$, and $\frac{1}{2.5}$. With the tested algorithms other than NR-GRC, files required by a job may be stored in remote clusters or a local cluster. To fairly compare all these algorithms, the router-to-site link and GRC-to-router link are given the same bandwidth, i.e., 2.5 Gb/s.

With NR-GRC, LC_c and LFA_c in Eq. (9) are zero since all master files are located at the GRC, i.e., $AFA_c = RFA_c$, $RC_c = \frac{1}{2.5} + \frac{1}{10} + \frac{1}{0.1} = 10.5$, and $C_{baseline} = \frac{1}{2.5} + \frac{1}{10} + \frac{1}{10} + \frac{1}{0.1} = 10.6$. Thus, $BCR_c = \frac{RC_c}{C_{baseline}} = 0.99$, $ABCR = BCR_c$, and $ABCR = 0.99$ in all rounds on all access patterns. For PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC, $RC_c = C_{baseline} = 10.6$, and $LC_c = \frac{1}{2.5} + \frac{1}{0.1} = 10.4$.

Comparing the plots shown in Figs. 17–21, *ABCR*s of NR-LRC in the five figures are all the worst, between 0.997 and 0.998. The reason is that NR-LRC does not replicate files among clusters; hence, each cluster has to access required files from remote clusters, even though it has frequently accessed these files. PFRF can effectively adjust file weights and lead to the best *ABCR*s on ZipfL-0.8, ZipfL-0.6, and Geo-0.2 after the sixth round, as compared with all the other algorithms. However, *ABCR*s of NR-GRC are better than those of other algorithms on ZipfL-0.6 (see Fig. 18) since all unpopular files have similar access count (see Fig. 6), like those of a uniform distribution. As shown in Fig. 20, PFRF, M/LFU, and DR-Global on Geo-0.5 have similar *ABCR*s, which are better than those of DR-Local, NR-GRC, and NR-LRC since the former three algorithms can effectively identify popular files and replicate them to the clusters requiring these files.

As shown in Fig. 21, NR-GRC on Uniform with JRR = 25% outperforms all the other algorithms since all required files can be accessed from the GRC rather than from remote clusters. On the other hand, *ABCR*s of PFRF, M/LFU, DR-Local, and DR-Global are similar because popularities of all files as shown in Fig. 9 are similar and remain unchanged over time. According to Eqs. (9) and (10), the increase of RFA_cs will result in higher BCR_cs and *ABCR*s, indicating that the bandwidths consumed by all algorithms except NR-GRC on Uniform, of which *ABCR* is about 0.993 after the sixth (see Fig. 21), are higher than those on the other access patterns. Please compare the *ABCR* value 0.993 with those shown in Figs. 17–20, all between 0.983 and 0.992.

If we change the bandwidth of the user-to-router links shown Fig. 4 from 100 Mb/s to 1 Gb/s, when NR-GRC is employed, $RC_c = 1.5$, $C_{baseline} = 1.6$, and $BCR_c = ABCR = 0.937$ which is lower

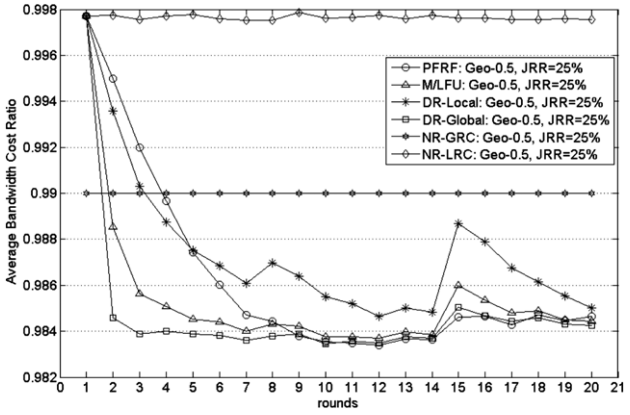


Fig. 20. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on Geo-0.5 with JRR = 25%.

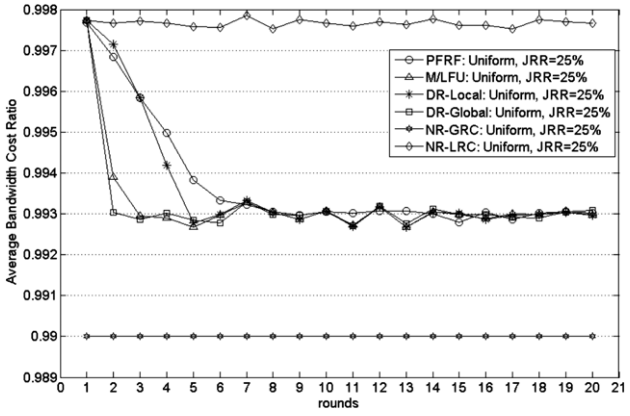


Fig. 21. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on Uniform with JRR = 25%.

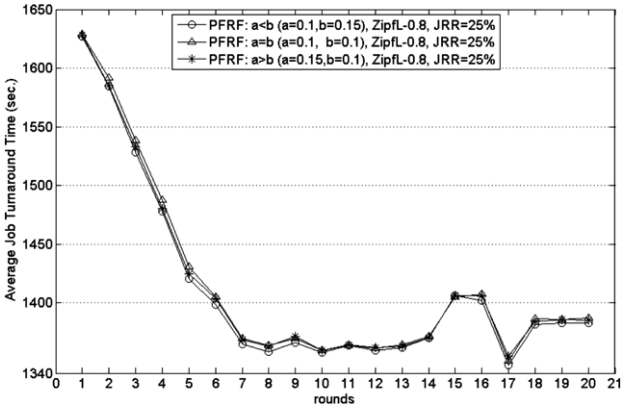


Fig. 22. The average job turnaround time against different rounds.

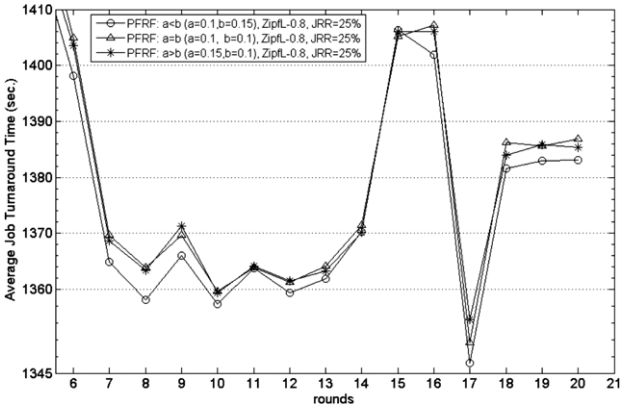


Fig. 23. The zoom-in figure between rounds 6 to 20 in Fig. 22.

than that calculated above, i.e. 0.99. When NR-LRC is used, $RC_c = C_{baseline} = 1.6$, $LC_c = 1.4$, and $ABCR \approx 0.984$ which is also lower than the value calculated above for NR-LRC, i.e., between 0.997 and 0.998, implying that when there is a bottleneck along a path, the *ABCR* will be bigger. Other algorithms have similar phenomena.

4.3.3. Comparison of PFRF parameters a and b

The experimental environment used to evaluate the parameters a and b in Eq. (3) is the same as the one mentioned in Section 4.1. Fig. 22 shows experimental results for *ATT*s of PFRF when $a < b$(i.e., $a = 0.1, b = 0.15$), $a = b$ (i.e., $a = 0.1, b = 0.1$), and $a > b$ (i.e., $a = 0.15, b = 0.1$) on ZipfL-0.8 with JRR = 25%. Fig. 23, a zoomed-in portion of Fig. 22, illustrates the *ATT*s of PFRF from round 6 to round 20. Evidently, the case of ($a = 0.1, b = 0.15$) has the best *ATT*s, showing that it can accurately reflect which files are more popular. Therefore, in this study we select ($a = 0.1, b = 0.15$) to do the previous simulations.

4.3.4. Discussion

From an end user viewpoint, the goal of invoking a data replication algorithm is to shorten average turnaround time and enhance data availability. 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, M/LFU, DR-Local, and DR-Global can improve turnaround time, bandwidth cost ratio, and data availability. Although NR-LRC can slightly reduce job turnaround time and improve data availability, its average bandwidth cost is still high. NR-GRC has the best average bandwidth cost ratios on the uniform distribution, but it has the worst turnaround time and data availability on all data access patterns.

In most situations, PFRF outperforms M/LFU, DR-Local, DR-Global, and NR-LRC on the three performance metrics. Although DR-Global has similar performance to that of PFRF, its high replication frequency will run out of storage space quickly, and consume considerable network bandwidth, resulting in high disk I/O costs for all clusters in a data grid. In our simulations, the average job turnaround time does not include the total popular file replication time. In other words, DR-Global is not really optimal.

Note that the GRC is only responsible for supplying the locations of files/replicas, instead of files/replicas themselves, to all clusters when PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC algorithms are employed. Therefore, the bandwidth of the GRC will not be the bottleneck of these algorithms. However, with NR-GRC, all requested files are transmitted from the GRC. The bandwidth of the GRC might be a bottleneck when it receives a high volume of file requests from clusters.

5. Conclusions and future work

In this paper, we propose a novel data replication algorithm for a star-topology data grid with limited storage space to improve system performance. Although some previous studies have done that, such as providing shorter job response time, higher network usage, and less storage occupation on limited storage space, they did not consider data access patterns and the change of user

access behaviors over time. Therefore, PFRF is designed to improve these weaknesses. It can effectively adapt to the changes of users' interests by dynamically adjusting file weights and replicating these files to appropriate clusters to improve performance of the whole system. We also analyze the average job turnaround time, average data availability, and average bandwidth cost ratio as the performance metrics of PFRF and compare them with those of five existing algorithms on five data access patterns. The simulation results show that PFRF outperforms all the tested algorithms when file popularity changes with time.

In the future, we plan to validate our simulation results on real data grids so that the proposed scheme can be evaluated on a real testbed. We would also like to enhance the reliability of the architecture by providing a hot-standby GRC like that presented in [49] to take over as the GRC when the GRC cannot work properly. We will also try to replicate popular files to users' local sites, rather than to users' local clusters. This can further reduce intra-cluster bandwidth consumption and unnecessary data transmission time. Finally, we plan to develop a reliability model to evaluate how many replicas are required for a file such that the file can stand against site failures. Those constitute our future studies.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grants NSC 99-2221-E-009-123-MY2 and NSC 100-2221-E-029-018.

The authors would like to thank the anonymous referees for their helpful comments that improved the quality of this paper.

References

[1] A. Folling, C. Grimme, J. Lepping, A. Papaspyrou, Robust load delegation in service grid environments, IEEE Transactions on Parallel and Distributed Systems 21 (9) (2010) 1304–1316.

[2] O. Sonmez, H. Mohamed, D. Epema, On the benefit of processor coallocation in multicluster grid systems, IEEE Transactions on Parallel and Distributed Systems 21 (6) (2010) 778–789.

[3] H. Li, Realistic workload modeling and its performance impacts in large-scale escience grids, IEEE Transactions on Parallel and Distributed Systems 21 (4) (2010) 480–493.

[4] H. Mohamed, D. Epema, Koala: a co-allocating grid scheduler, Concurrency and Computation: Practice and Experience 20 (16) (2008) 1851–1876.

[5] B. Tierney, W. Johnston, J. Lee, M. Thompson, A data intensive distributed computing architecture for grid applications, Future Generation Computer Systems 16 (5) (2000) 473–481.

[6] BIRN. http://www.nbrin.net/.

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

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

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

[10] PPDG. http://www.ppdg.net.

[11] R.S. Chang, M.S. Hu, A resource discovery tree using bitmap for grids, Future Generation Computer Systems 26 (1) (2010) 29–37.

[12] J. Wu, X. Xu, P. Zhang, C. Liu, A novel multi-agent reinforcement learning approach for job scheduling in grid computing, Future Generation Computer Systems 27 (5) (2011) 430–439.

[13] S. Ebadi, L.M. Khanli, A new distributed and hierarchical mechanism for service discovery in a grid environment, Future Generation Computer Systems 27 (6) (2011) 836–842.

[14] M.E.J. Newman, Power laws, Pareto distributions and Zipf's law, Contemporary Physics 46 (2005) 323–351.

[15] K. Sashi, A.S. Thanamani, Dynamic replication in a data grid using a modified BHR region based algorithm, Future Generation Computer Systems 27 (2) (2011) 202–210.

[16] M. Lei, S.V. Vrbsky, X. Hong, An on-line replication strategy to increase availability in data grids, Future Generation Computer Systems 24 (2) (2008) 85–98.

[17] O. Wolfson, S. Jajodia, Y. Huang, An adaptive data replication algorithm, ACM Transactions on Database Systems 22 (2) (1997) 255–314.

[18] M. Rabinovich, I. Rabinovich, R. Rajaraman, Dynamic replication on the internet, Technical Report, HA6177000–980305-01-TM, AT&T Labs, March 1998.

[19] J.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 26 (1) (2010) 12–20.

[20] M. Vrable, S. Savage, G.M. Voelker, Cumulus: filesystem backup to the cloud, ACM Transactions on Storage 5 (4) (2009).

[21] H. Shen, An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems, IEEE Transactions on Parallel and Distributed Systems 21 (6) (2010) 827–840.

[22] K. Ranganathan, I. Foster, Identifying dynamic replication strategies for a high performance data grid, in: Proceedings of the Second International Workshop on Grid Computing, Denver, CO, November 2001, pp. 75–86.

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

[24] R.S. Chang, H.P. Chang, A dynamic data replication strategy using access-weights in data grids, Journal of Supercomputing 45 (3) (2008) 277–295.

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

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

[27] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010).

[28] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: evidence and implications, in: Proceedings of IEEE INFOCOM'99, no.1, New York, USA, ppp. 126–134, 21–25 March 1999.

[29] D.G. Cameron, R. Carvajal-Schiaffino, A. Paul Millar, C. Nicholson, K. Stockinger, F. Zini, Evaluating scheduling and replica optimisation strategies in optorsim, in: The International Workshop on Grid Computing, Phoenix, Arizona, November 17, IEEE Computer Society Press, 2003.

[30] K. Ranganathan, I. Foster, Decoupling computation and data scheduling in distributed data intensive applications, in: International Symposium for High Performance Distributed Computing, HPDC-11, Edinburgh, 2002.

[31] K. Ranganathan, I. Foster, Simulation studies of computation and data scheduling algorithms for data grids, Journal of Grid Computing 1 (2003) 53–62.

[32] R.S. Chang, J.S. Chang, S.Y. Lin, Job scheduling and data replication on data grids, Future Generation Computer Systems 23 (7) (2007) 846–860.

[33] M. Coates, A. Hero, R. Nowak, B. Yu, Internet tomography, IEEE Signal Processing Magazine 19 (3) (2002).

[34] G. Levitin, Y.S. Dai, B.H. Hanoch, Reliability and performance of star topology grid service with precedence constraints on subtask execution, IEEE Transactions on Reliability 55 (3) (2006) 507–515.

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

[36] A. Silberschatz, P.B. Galvin, G. Gagne, Operating System Concepts, 7th ed., Wiley, 2004.

[37] L.M. Khanli, A. Isazadeh, T.N. Shishavan, PHFS: a dynamic replication method, to decrease access latency in the multi-tier data grid, Future Generation Computer Systems 27 (3) (2011) 233–244.

[38] M.L. Yiu, H. Lu, N. Mamoulis, M. Vaitis, Ranking spatial data by quality preferences, IEEE Transactions on Knowledge and Data Engineering 23 (3) (2011).

[39] P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, File-based replica management, Future Generation Computer Systems 21 (2005) 115–123.

[40] S.M. Park, J.H. Kim, Y.B. Ko, W.S. Yoon, Dynamic Data Grid Replication Strategy Based on Internet Hierarchy, in: Lecture Notes in Computer Science, vol. 3033, 2004, pp. 838–846.

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

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

[43] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, R. Buyya, A toolkit for modelling and simulating data grids: an extension to gridsim, in: Concurrency & Computation: Practice and Experience, Wiley Press, New York, USA, 2008.

[44] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, A. Chien, The microgrid: a scientific tool for modeling computational grids, in: Proc. of IEEE Supercomputing Conference, Dallas, USA, November 4–10 2000.

[45] W. Bell, D. Cameron, L. Capozza, P. Millar, K. Stockinger, F. Zini, Simulation of dynamic grid replication strategies in optorsim, in: Proc. of the 3rd International Workshop on Grid Computing, GRID, Baltimore, USA, 18 November 2002.

[46] A. Legrand, L. Marchal, H. Casanova, Scheduling distributed applications: the simgrid simulation framework,in: Proc. of the 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, May 12–15 2003.
[47] C.M. Dobre, C. Stratan, Monarc simulation framework, in: Proc. of the RoEduNet International Conference, Timisoara, Romania, May 27–28 2004.
[48] ChicSim—the Chicago grid simulator, 5 October 2007. http://people.cs.uchicago.edu/_krangana/ChicSim.html.
[49] F.Y. Leu, C.T. Yang, F.C. Jiang, Improving reliability of a heterogeneous grid-based intrusion detection platform using levels of redundancies, Future Generation Computer Systems 26 (4) (2010) 554–568.

Ming-Chang Lee received the M.S. degree in Computer Science Department from TungHai University, Taiwan, in 2006. Currently, he is a Ph.D. student of National Chiao Tung University, Hsinchu, Taiwan. His research interests include grid applications, distributed computing, network security, and estimation of distribution algorithms.

Fang-Yie Leu received his B.S., Master and Ph.D. degrees from National Taiwan University of Science and Technology, Taiwan, in 1983, 1986 and 1991, respectively, and another Master's degree from Knowledge System Institute, USA, in 1990. His research interests include wireless communication, network security, Grid applications and Chinese natural language processing. He is currently a professor of Tunghai University, Taiwan, and the director of database and network security laboratory of the University. He is also a member of IEEE Computer Society.

Ying-ping Chen received the B.S. degree and the M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in 2004 from the Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA.

He is currently an associate professor in the Department of Computer Science, National Chiao Tung University, Taiwan. His research interests include data grid and MapReduce technologies in distributed computation as well as theories, working principles, and dimensional/facet-wise models in genetic and evolutionary computation.