DOI 10.12694/scpe.v15i1.969
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 1, pp. 101–119. http://www.scpe.org

A NEW DECENTRALIZED PERIODIC REPLICATION STRATEGY FOR DYNAMIC DATA GRIDS*

HANÈNE CHETTAOUI† AND FAOUZI BEN CHARRADA‡

Abstract. Data grids provide scalable infrastructure for storage resource and data files management, which support data-intensive applications that need to access to huge amount of data stored at distributed locations around the world. The size of these data can reach the scale of terabytes or even petabytes in many applications. These applications require reaching several main goals, namely efficient accessing, storing, transferring and analyzing a large amount of data in geographically distributed locations. In this situation, replication is a general and simple technique used in data grids to achieve these goals. Indeed, it has as main purposes improving data access efficiency, providing high availability, decreasing bandwidth consumption, improving fault tolerance and enhancing scalability. In this paper, we propose a new classification of replication strategies through two complementary criteria as well as a survey of the induced categories of strategies. In addition, we introduce a new decentralized periodic replication strategy for dynamic data grids assuming limited storage for replicas, called DPRSKP, which stands for Decentralized Periodic Replication Strategy based on Knapsack Problem. This strategy takes into consideration the changing availability of sites. DPRSKP is based on two polynomial-time complexity algorithms. The first one starts by selecting the best candidate files for replication while the second places them in the best locations. The replication problem in DPRSKP is formulated according to the Knapsack problem. In addition, DPRSKP extends the well known LRU and LFU strategies. The simulation experiments were carried out using OptorSim and a dynamic period rather than a static one. The obtained results show that DPRSKP can effectively improve response time, bandwidth consumption, remote file accesses number and local file accesses number as compared with other replication strategies.

Key words: data grid, replication, knapsack problem, strategy, period, decision taking, complexity

119

1. Introduction and motivations. Nowadays, a huge amount of data is generated in many fields of science and engineering, some of which are geographic information science [43], astrometry [45], medical [32], and remote instrumentation [47]. These produced data need to be distributed around the world for the sake of data sharing and collaboration. Hence, ensuring an efficient and fast access to such massive data is a challenge that must be addressed. The data grid, which represents a type of grid computing [15], is a solution for this problem. A data grid connects a collection of hundreds of geographically distributed computers and storage resources located in different parts of the world to facilitate sharing of data and resources [23]. Biomedical Informatics Research Network (BIRN)[1], Large Hadron Collider (LHC)[2] and, the European DataGrid Project (EDG)[3] are some examples of existing data grids.

In this paper, we focus on data grids. Since the data represent the most important resource in data grids, how to decrease access time, reduce the bandwidth consumption and improve the data availability and system reliability have become challenging tasks. Replication is a key technique often used to achieve these tasks. The general idea of replication is to create multiple copies of the same data in several storage resources to improve response time, reduce the bandwidth consumption, increase data availability, and to improve system reliability. Currently, data replication in grids mostly deals with file replication [20]. Several replication strategies have been proposed in the literature. A replication strategy must answer the following questions [9, 21, 38]:

1- Which files must be replicated?
2- Where to place the candidate files for replication (*i.e.*, the new replicas)?
3- How to select the best replica of a file among many replicas available in the grid?

In the literature, replication strategies are categorized according to several criteria. Indeed, replication strategies can be classified according to the adopted grid model. We then distinguish between strategies dedicated for hierarchical grids and other dedicated for Peer-To-Peer or hybrid grids. In addition, replication strategies can be classified according to the taking decision type. Hence, two strategy types appeared which are centralized and decentralized strategies. On the one hand, a centralized replication strategy is based on a central site that collects all information about files and grid sites. The central site makes the replica and

*This paper is a largely extended version of our work proposed in [14].

†Department of Computer Sciences, Faculty of Sciences of Tunis, Tunis, Tunisia (hanene1ch@gmail.com).
‡Department of Computer Sciences, Faculty of Sciences of Tunis, Tunis, Tunisia (f.charrada@gnet.tn).

[1]BIRN: http://www.birncommunity.org/
[2]LHC accelerator project: http://lhc.web.cern.ch/lhc/
[3]EU Data Grid Project: http://www.eu-datagrid.org/

placement decisions. On the other hand, in a decentralized approach, all grid sites keep files track in order to decide which files to replicate or to remove locally. We also find static or dynamic replication strategies. A static replication way decides the distribution files initially and retains files location during the grid running. However, in a dynamic replication type, replicas are automatically created and deleted according to changes in the grid users' behavior.

The majority of strategies on data replication have considered only static grids (having invariable number of sites). For several types of grids, this assumption does not reflect reality. Indeed, new sites can join the grid and others can leave, to join it possibly again. For this reason, the dynamicity becomes a fundamental criterion to establish an effective replication strategy.

In this paper, we propose a new decentralized periodic replication strategy, called DPRSKP (Decentralized Periodic Replication Strategy based on Knapsack Problem) for dynamic data grids. In fact, DPRSKP has two main features:

- the storage space of each site is supposed to be limited, contrary to several approaches of the literature which suppose it unlimited.
- the number of grid sites is variable in the sense that sites can leave or join the grid at any time.

In our strategy, we will address these issues characterizing the grid dynamicity. For each site, the DPRSKP strategy selects files to be replicated or deleted based on an adaptation of the well known Knapsack problem [22, 30]. We then evaluate the benefit of our strategy through simulations. The obtained experiment results, using OptorSim, show that our strategy outperforms other replication strategies in terms of several parameters. It is important to note that, as this will be shown hereafter, the proposed strategy is periodic-fee in the sense that it can also be used as a non periodic strategy.

The remainder of the paper is organized as follows. Section 2 gives an overview of previous work on data grid replication. Section 3 presents our replication strategy DPRSKP as well as an illustrative example of its different components. Section 4 describes a study of the complexity of our proposed strategy. Section 5 provides the simulation framework and the obtained experimental results. The last section summarizes our contributions and depicts future work.

2. Related work. Replication in data grids is a technique used to reduce the response time and the bandwidth consumption. Replication also increases data availability thereby enhancing system reliability. In this section, we classify and survey the main replication strategies of the literature.

We propose in this paper another type of classification according to the following two complementary criteria:

- Periodicity: this criterion specifies when the replication strategy is triggered, at each demand of a file or after a given period of time.
- Taking decision type: this criterion distinguishes between centralized and decentralized replication strategies.

As a consequence, replication strategies in the literature can be categorized as follows:

- A non periodic replication strategy is triggered by the requesting site when the needed file is not found locally and assumes limited storage for replicas. In that case, each strategy offers a replacement strategy when there is not enough space to replicate a new file. For this type of strategy, the decision about replicating or deleting files is taken by each requesting site in the grid. The associated strategies are then called non periodic decentralized strategies.
- A periodic replication strategy is triggered at each period. In general, these strategies assume unlimited replica storage. For this type of strategies, the selection of files to replicate and of sites to place them is done either by a central site or by the requesting site. We then distinguish between centralized periodic and decentralized periodic replication strategies.

The proposed classification of replication strategies is depicted in Fig. 2.1.

The following two subsections describe the main centralized and decentralized periodic and decentralized non periodic strategies of the literature.

2.1. Periodic strategies.

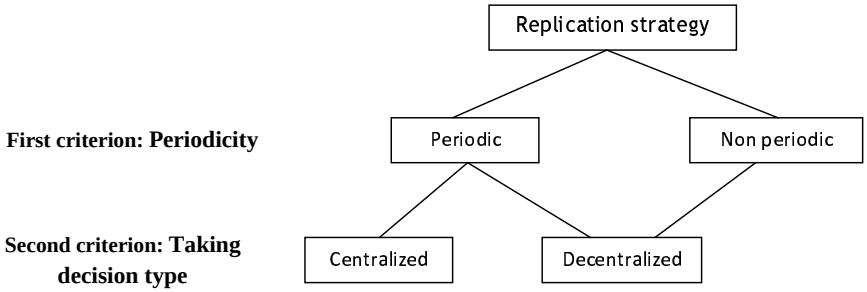

FIG. 2.1. *Classification of replication strategies*

2.1.1. Centralized strategies. This part exposes according to the chronological order the centralized periodic replication strategies where the replication algorithm is triggered by a central site.

In [37, 38], Ranganathan and Foster propose six different replication strategies (periodic and non periodic) for hierarchical data grids. They distinguish between replication and caching. Indeed, the authors consider replication as a server-side phenomenon and caching is assumed to be a client-side phenomenon. The proposed periodic centralized strategies are:

- No Replication: the entire data set is available at the root of the hierarchy. During the grid execution, the number of replicas and their locations remain the same.
- Cascading Replication: once the requests number of a file exceeds a given threshold, a replica is created at the next level which is on the path to the best client.

All these strategies take into account the requests number of each file to identify files to replicate and sites where to place them. In addition, a replacement strategy is triggered in the case there is a shortage of storage space at the selected site to replicate a new file. The least popular file is expunged. If more than one file are equally unpopular, the oldest file is removed.

In [34], Rahman *et al.* propose four replication algorithms based on utility and risk. The utility and risk are calculated using the current network load and the user requests. All these algorithms select one site to host the new replica. An improvement is proposed in [35, 36] by selecting p sites to place the new file using three models, namely p-median, p-center and multi-objective. Note that the number p is a parameter specified by the grid administrator.

In [18, 19], the authors propose a replication strategy by organizing the data into several data categories that it belongs to. They consider the requests number of each file to decide which one needs to be replicated. In addition, they take into account the file size and the bandwidth width to choose the best place for the new replica.

To trigger replica creation, Al Mistarihi and Yong [21] present a replication strategy for data grids that uses the requests number of each file as well as the time starting from the creation date of a replica until the current date. In addition, the candidate file for replication is identified based on the copies number [4] and the requests number of each file. Finally, the new replica is placed at the best site based on the number of requests and the response time.

Rasool *et al.* introduce a replication strategy, called Fair Share Replication (FSR) [40], for hierarchical data grids. The proposed strategy takes into account the requests number of each file to select the candidate file for replication. On the other hand, the replica placement is based on the server workload and the requests number of each client.

Chang *et al.* propose a replication strategy called Least Access Largest Weight (LALW) [13]. The information received at different time intervals has different weights. The concept of half-time is used to set the information weights. LALW works in three steps. The first step finds the candidate file for replication. The second computes how many replicas are required. The last step identifies clusters to host new replicas. These three steps are based on the requests number and the weight of each file.

[4]The copies number of a file is equal to the number of its replicas in the grid.

In [8, 9], Ben Charrada *et al.* propose a replication strategy called Periodic Optimiser for data grids. The first step of Periodic Optimiser identifies the candidate files for replication based on the requests number and the copies number associated to each file. The second step selects the best sites to host new replicas based on the requests number of each file and the availability of each site. The selection of the best replica depends on the availability of sites and the bandwidth.

Lee *et al.* propose a replication strategy called Popular File Replicate First (PFRF) [25] for hierarchical data grids containing a global replica controller (GRC) and several clusters connected to the GRC. After each period, the GRC selects the candidate files for replication based on the popularity weight for each file. The popularity weight is calculated according to the requests number of each file. For each candidate file for replication, the GRC selects the candidate clusters to place it. The candidate file is replicated in a site belonging to the candidate cluster when there is sufficient storage space to accommodate it. Otherwise, PFRF deletes some files which are less popular than the candidate file to be replicated.

2.1.2. Decentralized strategies. This part surveys, w.r.t. the chronological order, the decentralized periodic replication strategies where the replication algorithm is triggered by a requesting site.

As mentioned above (*see* page 102), Ranganathan and Foster propose in [37, 38] six, periodic and non periodic, replication strategies for hierarchical data grids. The authors design a decentralized periodic replication strategy called Best Client. In this strategy, each node maintains a detailed history describing each file that it contains (more precisely the requests number of each file and its requesting nodes). At each period, each node selects the popular files (having a requests number exceeding a given threshold) and a replica is created at the best client that has the largest number of requests for the file.

Ranganathan *et al.* introduce a replication strategy for a decentralized P2P environment [39]. The proposed strategy computes first the number of replicas needed per file based on the file availability and the accuracy of the RLS (Replica Location Service). Then, the best location of each new replica is chosen. The best site is then that maximizing the difference between replication benefits and replication costs.

Suri and Singh present a replication strategy, called DR2 (Two-Stage Dynamic Replication), for data grids [44]. DR2 works in two stages. Indeed, each site starts by selecting the files to replicate and then determines the best sites to transfer these files. The selection of the candidate file for replication takes into account the requests number, the size and the copies number associated to each file. The best replica is determined according to the bandwidth and the number of hops between the grid sites. In addition, LRU [3] is adopted as a replacement strategy.

Cui *et al.* propose a replication strategy called BSCA (Based on Support, Confidence and Access numbers) for hierarchical data grids [17]. After each period, each site selects the candidate files for replication whose the requests number exceeds a given threshold. Each candidate file and those files strongly correlated to it are replicated if there is enough space to accommodate them. Otherwise, the weakly correlated files are removed. If the storage space is still insufficient to replicate these files, BSCA deletes files having a requests number lower than a threshold. It is important to mention that no indication on the threshold choice is made.

2.2. Non periodic strategies. This section provides an overview in the chronological order of decentralized non periodic replication strategies. These strategies often replicate files at the request of any site and for any file. Therefore, a replacement strategy is used when there is not enough space to replicate a new file.

As mentioned above (*see* page 102), Ranganathan and Foster [37, 38] propose non periodic replication strategies for hierarchical grids. These strategies are:

- Plain Caching: the client that requests a file stores a copy locally.
- Caching plus Cascading Replication: this strategy combines Cascading Replication (*cf.* page 103) and Plain Caching strategies. Each client stores the requested files locally. In addition, at each period, each server identifies the popular files and replicates them down the hierarchy.
- Fast Spread: this strategy replicates files at each node along its path to the client.

All these strategies take into account the requests number of each file to identify files to replicate and sites where to place them. In addition, a replacement strategy is triggered in the case there is a shortage of storage space at the selected site to replicate a new file. The least popular file is expunged. If more than one file are equally unpopular, the oldest file is removed.

In [4], a replication strategy called LFU (Least Frequently Used) is introduced. LFU always replicates the requested file in the site where the job needing the replica is running. If there is not enough space to accommodate the replication, the least accessed file in the local storage is deleted. The strategy LRU (Least Recently Used) [3] proceeds in the same way as the LFU strategy, except that the oldest replica is removed.

Park *et al.* [33] introduce a replication strategy, called BHR (Bandwidth Hierarchy based Replication), which is based on bandwidth hierarchy of Internet. Nearest sites are located in the same network region. BHR tries to replicate files that will be used within the same region in the near future. If there is not enough space to accommodate the new replica, the least accessed files that are replicated in other sites within the same region will be deleted. If the available space is still insufficient to host the new replica, those files having a lower number of requests than the new replica are expunged.

In [26, 27], the authors propose a replication strategy called MinDmr (Minimize Missing Data Rate) which performs as follows. If the required file is not present locally, a new replica is created if the available free storage space is large enough. Otherwise, the candidate files for deletion are selected depending upon their weights. The file weight is computed according to the availability, the requests number, the size, and the copies number of the file.

Bsoul *et al.* [10] present a replication strategy named MFS (Modified Fast Spread) that is based on the Fast Spread strategy [37, 38]. MFS considers a network topology in which there is one server node and a number of client nodes. If there is enough storage space to replicate the required file, a new replica is created. Otherwise, a group of replicas needs to be removed if it is less important than the new replica. The file importance depends on the available storage space, the size, and the requests number of the file.

Zhao *et al.* [48] propose a replication strategy called VBRS (Value-Based Replication Strategy) that first calculates a threshold deciding whether the requested file should be replicated or not. The computed threshold depends on the requests number of each file. If the available storage space is insufficient to place the new replica, some files will be removed based on access time, bandwidth and file size.

Sashi and Thanamani [42] propose a strategy, called Modified BHR, to overcome the limitations of the BHR strategy (*cf.* page 105). According to Modified BHR, sites that are located in the same network region are grouped together. The main idea of Modified BHR is to replicate files within a region and to store replica in a site where the file has been accessed frequently. If there is not enough space to replicate the new replica in a site and the replica is duplicated in other sites within the region, no action is done. If there is no duplication, a replacement strategy is executed. It sorts files using the LFU strategy [4]. Then, it deletes files that are duplicated in other sites within the same region. If the available space is still insufficient to hold the new file, unpopular files having a requests number smaller than the new file requests number are deleted.

Ben Charrada *et al.* propose a replication strategy called LWF (Least Weight File) [5] which performs as follows. The required file is replicated in the requesting site if there is enough space to host the new replica. Otherwise, the candidate files for deletion are selected according to their weights. If the weight of the new replica is greater than the sum of the candidate files weights, they will be replaced by the new replica. Note that the file weight takes into account file-related parameters, *i.e.*, its size and requests number, as well as topology-related parameters, namely bandwidth and site availability.

Mehraban *et al.* propose a replication strategy called Prediction Replica Replacement Strategy (noted PRA) [31]. PRA is started when a site requests a file and does not have sufficient space to accommodate it. In this case, the requesting site computes the value of each file stored locally based on the requests number and the copies number of each file. Then, the requesting site determines the candidate files for deletion according to their sizes. Once the candidate files for deletion are selected, they will be sorted in ascending order according to their values. The files having the least values will be removed and the requested file will then be replicated.

In [2], Beigrezaei *et al.* propose a replication strategy called FUZZY_REP improving the modified BHR method [42]. According to the FUZZY_REP strategy, when there is not enough storage space to replicate the new requested file, the potential candidates for replacement will be chosen according to their weight. The weight computation is based on a fuzzy inference system and takes into account two factors, namely the number of replica access and the difference between the current time and the last access time of replica. The candidate replicas for replacement are sorted according to their respective weights in ascending order and, then, minimum weight replicas will be deleted. If the available storage space is still insufficient to place the new one, FUZZY_REP

tries to delete the replicas that were not stored in the other grid sites of the current region.

In [46] Vashisht *et al.* propose a replication strategy called EDRA (Efficient Dynamic Replication Algorithm) adapting agents for data grid. In EDRA, a data grid contains several regions connected with a master node. A number of grid nodes collectively form one sub-region. EDRA tries to place a requested replica in each sub-region. The head node located in regions is responsible for scheduling the jobs to the nodes where the replica is placed. An agent is placed on each head node of the region, which keeps information of the nodes located in the sub-regions. The new replica is placed on the nodes based on two factors, namely high data access frequency and sufficient storage capacity. If there is enough space to accommodate the new file, the replica is placed on the node. If the node has insufficient space, EDRA checks free space in the nearby node within the sub-region and replicates the new file. If there is insufficient space in the sub-region, old files are deleted from the node. The file having less access frequency is thus deleted and is replaced by the new replica.

Mansouri and Dastghaibyfard propose in [29] a replication strategy called Modified Dynamic Hierarchical Replication (MDHR) which is an enhanced version of Dynamic Hierarchical Replication (DHR) [28]. In MDHR, a data grid contains several regions. Each region contains several LAN. When a requested file is not available in the local storage, MDHR selects the best site to replicate the new file. The best site has the highest number of requests of the new file. If there is enough space in the selected site, the new file is replicated. Otherwise, if the new file is available within the LAN of the requested site, the file is accessed remotely. If the file is not available in the same LAN and there is not enough space to replicate it, files existing in the best site as well in the local LAN are deleted using the LRU strategy. Moreover, if the available storage is still insufficient to accommodate the new file, files existing in the best site as well in the local region are deleted using the LRU strategy. If the required space is still insufficient, remaining files in the best site are deleted also using the LRU strategy.

In the next section, we introduce our new strategy for dynamic data grids, called Decentralized Periodic Replication Strategy based on Knapsack Problem (DPRSKP). A dynamic data grid has a variable number of sites *i.e.* new sites can join the grid and others can leave, to join it possibly again.

3. DPRSKP strategy. As mentioned above, the majority of strategies on data replication have considered only static grids *i.e.* those having an invariable number of sites. For several types of grids, this assumption does not reflect the reality. Indeed, new sites can join the grid and others can leave it, to join it possibly again after. For this reason, the dynamicity becomes a fundamental criterion to design an effective replication strategy. In other words, by "dynamicity" we mean that the availability of a given site, *i.e.* the fact that it can be joined or not at given moment, will be taken into consideration in our approach.

In this context, we proposed in [8, 9] a centralized periodic replication strategy called Periodic Optimiser. Centralized replication is based on a central site that collects all information about files and grid sites. The central site makes the replica and placement decisions. However, in a decentralized strategy, all grid sites keep files track in order to decide which files to replicate or to remove locally. The benefit of a decentralized decision is that there is no single point of failure [1]. This motivates our choice of a decentralized way for replicating files. Moreover, in [5], we proposed a decentralized non period replication strategy called Least Weight File (LWF). LWF computes the weight of each file in a site S_i requesting a file F_{new}. In the first step of LWF, these files of S_i are sorted in an ascending order according to their weight. In the second step, LWF searches for the smallest index p of the files list as the sum of their sizes is greater than or equal to the missing space allowing to store F_{new}. Finally, F_{new} will be replicated if the sum of the weights of these p files is less than the weight of F_{new}. However, the choice of p is not performed optimally as shown in the following counter example. Let $Size_{F_1} = 2$, $Size_{F_2} = 3$, $Size_{F_3} = 5$ and $Size_{F_4} = 6$ where $Size_{F_k}$ denotes the size of a file F_k locally stored in the site S_i. Also, let $FileWeight_{F_1} = 4$, $FileWeight_{F_2} = 5$, $FileWeight_{F_3} = 6$, $FileWeight_{F_4} = 10$ and $FileWeight_{F_{new}} = 7$ where $FileWeight_{F_k}$ denotes the weight of a file of S_i (for F_1, F_2, F_3 and F_4) and that to be replicated (F_{new}). Suppose the missing space is 5, *i.e.*, $Size_{F_{new}}$ - $S_i.FreeSpace = 5$. LWF chooses F_1 and F_2 whose total weight is 9 (> 7). Therefore, F_{new} will not be replicated. On the other side, if F_3 is chosen ($FileWeight_{F_3} = 6$), then F_{new} is replicated and F_3 is deleted from S_i since (i) $FileWeight_{F_3} = 6 < 7$, (ii) $Size_{F_3} = 5$.This counter-example can be avoided through making the replication problem as a Knapsack problem with bivalent variables [22, 30, 41].

3.1. Proposed strategy. In this subsection, we present our decentralized periodic replication strategy DPRSKP for dynamic P2P data grids assuming limited replica storage. DPRSKP aims at increasing files availability, improving response time and reducing bandwidth consumption.

DPRSKP performs as follows. After each period, each site S_i identifies the candidate files for replication. These files are replicated if there is enough storage space. However, if the available storage space is insufficient to hold these files, the site S_i should place the files with the highest priority among the candidate files and local ones of S_i by using the Knapsack problem. Thus, our strategy mainly deals with the following steps:

1. Selection of candidate files for replication;
2. Placement of a set of files in the site.

It is worth mentioning that, for answering the third question required by a replication strategy (*cf.* page 101), we rely on the formula proposed in [9].

3.1.1. Selection of the candidate files for replication. At each period, each site S_i identifies the candidate files for replication based on the following parameters:

- $\#Request_{S_i,F_k}$: the number of times that a file F_k has been requested by the site S_i.
- $\#Replica_{F_k}$: the number of available copies of a file F_k on the whole grid.

We consider that a file F_k needs to be replicated in S_i if it has been requested many times by S_i and there are not enough copies of F_k in the grid. For that purpose, a metric is introduced, called average value of a file F_k for a site S_i, noted $AVG\_VALUE_{S_i,F_k}$, and defined as follows:

$$AVG\_VALUE_{S_i,F_k} = \frac{\#Request_{S_i,F_k}}{\#Replica_{F_k}}$$

The average value of a file F_k for a site S_i represents the average number of requests for a replica of F_k by S_i.

Similarly to the previous metric, we introduce a metric called average value of a site S_i, denoted $AVG\_VALUE_{S_i}$, and defined by the following expression:

$$AVG\_VALUE_{S_i} = \frac{\sum_{k=1}^{n} \#Request_{S_i,F_k}}{\sum_{k=1}^{n} \#Replica_{F_k}}$$

where n is the total number of requested files by the site S_i.

These two metrics are used in Algorithm 3.1.1 for choosing the best candidate files for replication in a site S_i. Algorithm 3.1.1 takes as entry the requests number and the copies number of each file requested by S_i as a list of triplets $S_F = \{(F_k, \#Request_{S_i,F_k}, \#Replica_{F_k}) \mid k = 1..n\}$, where n is the total number of requested files by the site S_i.

In the first loop (*cf.* Lines 5 - 8), we calculate the average value of each requested file by the site S_i. In the second loop (*cf.* Lines 10 - 12), we select the candidate files for replication. A file F_k is considered as a candidate file for replication if its average value exceeds the average value of the site S_i.

3.1.2. Replica placement. After selecting the candidate files for replication in a site S_i, it is necessary to check the available storage space of S_i that contains locally a set of files noted $\mathcal{F}$. If the free storage space is sufficient to hold the set of all candidate files for replication, noted $\mathcal{F}_{ToReplicate}$, then the replication of these files will be conducted without any constraint. If there is not enough storage space, the potential candidate files for replacement will be chosen to place all or part of $\mathcal{F}_{ToReplicate}$ in S_i. As our strategy aims to reduce the response time and the bandwidth consumption, the choice of the potential replacement candidate files is important because they can be requested in the future by the same site S_i.

To achieve these objectives by choosing the best potential candidate files for replacement in a site S_i, we introduce the average number of requests of each file $F_k \in \mathcal{F} \cup \mathcal{F}_{ToReplicate}$ by S_i, noted $\#AVG\_Request_{S_i,F_k}$,

Algorithm 1: Selection of the candidate files for replication

Input: $S_F = \{(F_k, \#Request_{S_i,F_k}, \#Replica_{F_k})\}$
Output: $CandidateFile$: the set of candidate files for replication

```
Begin
  Request_Sum ← 0; /*Computation of the numerator of the formula (3.2)*/
  Replica_Sum ← 0; /*Computation of the denominator of the formula (3.2)*/
  CandidateFile ← ∅;
  For k = 1 to | S_F | do
      AVG_VALUE_{S_i,F_k} ← #Request_{S_i,F_k} / #Replica_{F_k};
      Request_Sum ← Request_Sum + #Request_{S_i,F_k};
      Replica_Sum ← Replica_Sum + #Replica_{F_k};
  AVG_VALUE_{S_i} ← Request_Sum / Replica_Sum;
  For k = 1 to | S_F | do
      If AVG_VALUE_{S_i,F_k} ≥ AVG_VALUE_{S_i} then
          CandidateFile ← CandidateFile ∪ {F_k}
  return CandidateFile
End
```

and defined as follows:

$$\#AVG\_Request_{S_i,F_k} = \frac{\#Request_{S_i,F_k}}{T_{Current} - T_{FirstRequest_{S_i,F_k}} + 1}$$

where:

- $\#Request_{S_i,F_k}$: the total number of requests of F_k by S_i;
- $T_{Current}$: the number of the current period;
- $T_{FirstRequest_{S_i,F_k}}$: the period number of the first request of F_k by S_i.

Then, we associate to each file $F_k \in \mathcal{F} \cup \mathcal{F}_{ToReplicate}$ a file weight (denoted $FileWeight_{F_k}$) defined by the following expression:

$$FileWeight_{F_k} = \frac{Size_{F_k} \cdot \#AVG\_Request_{S_i,F_k}}{BW_{(S_i,S_j)F_k} \cdot P_{S_j}}$$

where:

- $Size_{F_k}$: the size of F_k;
- $\#AVG\_Request_{S_i,F_k}$: the average number of requests of F_k by S_i;
- $BW_{(S_i,S_j)F_k}$: the bandwidth between the requesting site S_i and the site S_j ($j \neq i$) containing the best replica of F_k. The choice of the best replica is done using the formula proposed in [8, 9] and is determined as follows:

$$\underset{j=1}{\overset{|\Xi_{F_k}|}{Max}} \left(BW_{(S_i,S_j)F_k} \cdot P_{S_j}\right)$$

  where Ξ_{F_k} is the set of sites containing a replica of F_k.
- P_{S_j}: the availability of the site S_j defined by the following expression [8, 9]:

$$P_{S_j} = 1 - \frac{\#Failure_{S_j}}{\#SiteRequest_{S_j}}$$

  where $\#Failure_{S_j}$ is the number of times the site S_j is failing (*i.e.*, the number of requests to S_j which are not satisfied) and $\#SiteRequest_{S_j}$ is the total number of times the site S_j has been requested.

For the site S_i, according to the expression (3.1.2), the most popular files situated in unstable sites having a low bandwidth with S_i are the most preferred to be placed in it. Indeed, it is preferred to have such files in

S_i since it requests them several times. On the other side, if they are not locally stored in S_i, each time they will be of need in a job executed by S_i, their use will be high costly due to the low quality of bandwidth with S_i for the best site where they are contained as well as the low availability of these latter.

For each site S_i, the goal of our strategy DPRSKP is to find a set of files $\Omega \subset F \cup \mathcal{F}_{ToReplicate}$ that can be stored in S_i and having a maximum total files weights.

To achieve this goal, our strategy is formulated according to the Knapsack problem to select files to be replicated and those to be deleted. The replication problem is then formulated as follows.

We associate to each file $F_k \in \mathcal{F} \cup \mathcal{F}_{ToReplicate}$ the value $X_k \in \{0,1\}$ defined by:

$$X_k = \begin{cases} 1\text{: } F_k \text{ should be placed in } S_i \\ 0\text{: } F_k \text{ should not be placed in } S_i \end{cases}$$

$$(K) \begin{cases} \sum_{k=1}^{N} (Size_{F_k} \cdot X_k) \leq SE_{S_i} \text{ such that } X_k \in \{0,1\} \\ \sum_{k=1}^{N} (FileWeight_{F_k} \cdot X_k) = Z(Max) \end{cases}$$

where:
N is the cardinality of the set $\mathcal{F} \cup \mathcal{F}_{ToReplicate}$;
$Z(Max)$ is the objective function to maximize;
SE_{S_i} is the storage space of S_i.

3.1.3. Resolution algorithm. The resolution of our replication problem formulated through to the Knapsack problem by exact algorithms using the Branch and Bound method, the polyhedral techniques or the Dynamic Programming can be costly in terms of response time. In such a situation, it would be better to use approximation methods with a reasonable compromise between response time and approximation quality. In general, the approximation algorithms are effective. This is the case of greedy algorithms having generally a low complexity. The idea often used in such algorithms is to keep the option of the local optimum according to some criteria in order to have a global optimal solution [16, 22, 30].
Our proposed algorithm gives an approximate solution that is acceptable given response time constraints. The greedy algorithm 3.1.3 presents the placement of a set of files in a site S_i.

Algorithm 2: Replication of a set of files in a site S_i

```
   Input: S_i, SE_{S_i}, F = {F_1, F_2, ..., F_p}, F_ToReplicate
1  Begin
2    If S_i has enough free space for F_ToReplicate then
3      For Each file F_k of F_ToReplicate do
4        Get F_k from S_j and replicate it in S_i /*S_j denotes the site containing
         the best replica of F_k*/
5    Else
6      For Each file F_k of F ∪ F_ToReplicate do
7        Calculate Efficiency_{F_k} = FileWeight_{F_k} / Size_{F_k}
8      Sort the files of F ∪ F_ToReplicate in descending order according to the
       Efficiency metric in the list denoted SortList
9      AllocateSpace=0
10     For Each file F_k of SortList do
11       If AllocateSpace + Size_{F_k} ≤ SE_{S_i} then
12         X_k ← 1
13         AllocateSpace ← AllocateSpace + Size_{F_k}
14       Else
15         X_k ← 0
16 End
```

In Algorithm 3.1.3, we use the following terminology. We introduce the file efficiency for a file F_k, noted $Efficiency_{F_k}$, as the ratio of the weight to the size of F_k (*cf.* Line 7 of Algorithm 3.1.3). The more the weight

of F_k is large relative to the storage capacity consumption, the more F_k is interesting. To summarize:

- For each file F_k such as $X_k = 1$: F_k will be placed in S_i.
- For each file F_k such as $X_k = 0$:

$$\begin{cases} F_k \textit{ is deleted from } S_i \textit{ if } F_k \in \mathcal{F} \\ F_k \textit{ is not replicated in } S_i \textit{ if } F_k \in \mathcal{F}_{ToReplicate} \end{cases}$$

3.2. Illustrative example. Let us consider the data grid, shown in Fig. 3.1, which contains four sites. The arcs sketch the available bandwidth between sites in Mb/s. The availability of each site is given into brackets. For instance, the availability of S_2 is 0.5. The master files[5] are placed in the site S_0. All the other sites have a storage capacity of 5 Gb. We will apply DPRSKP strategy to the site S_1.

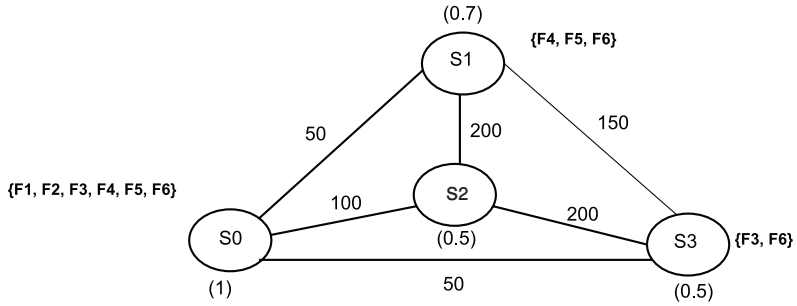

FIG. 3.1. *Topology of the grid*

Step 1: Determination of the candidate files for replication in S_1

Table 3.1 sketches the requested files by the site S_1. The execution of Algorithm 3.1.1 provides:

- The average value of S_1 is $AVG\_VALUE_{S_1} = 300$.
- F_1 and F_3 as candidate files for replication in S_1 since the associated average values are greater than or equal to 300.

TABLE 3.1
The values of $\#Request_{S_1,F_k}$, $\#Replica_{F_k}$ and $AVG\_VALUE_{S_1,F_k}$ for the three files requested by S_1

F_j	$\#Request_{S_1,F_k}$	$\#Replica_{F_k}$	$AVG\_VALUE_{S_1,F_k}$
F_1	1 000	2	500.00
F_2	500	3	166.66
F_3	300	1	300.00

Step 2: Files placement in S_1

Table 3.2 describes the candidate files for replication (F_1 and F_3) and those stored locally (F_4, F_5 and F_6) for the site S_1. We note that the number of the current period ($T_{Current}$) is 10.

We remark that the free space at the site S_1 is not enough to replicate F_1 and F_3. Indeed, the size of the files stored in S_1 is 4 Gb. Therefore, the available free space in S_1 is 1 Gb which is insufficient to replicate F_1 and F_3 because the total size of F_1 and F_3 is 3 Gb. Thus, we need to calculate the weight and the efficiency of these files and those local ones. Table 3.3 sketches the obtained results.

The execution of Algorithm 3.1.3 provides the following result: the selected files to be placed in S_1 are F_1, F_3, F_4 and F_6. Consequently, the file F_5 will be deleted from S_1 while F_1 and F_3 will be replicated in S_1.

[5]A master file contains the original copy of some data sample and cannot be deleted [4].

TABLE 3.2
Characteristics of the files set $\mathcal{F} \cup \mathcal{F}_{ToReplicate}$

F_k	$\#Request_{S_1,F_k}$	$Size_{F_k}$ in Gb	S_j	$T_{FirstRequest_{S_1,F_k}}$
F_1	1 000	2	S_2	10
F_3	300	1	S_3	10
F_4	800	1	S_2	7
F_5	300	2	S_2	6
F_6	700	1	S_3	5

TABLE 3.3
Weight and efficiency of $\mathcal{F} \cup \mathcal{F}_{ToReplicate}$

F_k	$Size_{F_k}$ in Gb	$FileWeight_{F_k}$	$Efficiency_{F_k}$
F_1	2	20.00	10.00
F_3	1	4.00	4.00
F_4	1	2.00	2.00
F_5	2	1.20	0.60
F_6	1	1.55	1.55

3.3. DPRSKP extends the LFU and LRU strategies. A main feature of our strategy DPRSKP is that it allows subsuming the LFU and LRU strategies. To explain this, it is necessary to prove that each file retained by either the LRU or the LFU strategy is also retained by DPRSKP strategy. Recall that:

- The file weight of a file F_k is defined as follows:

$$FileWeight_{F_k} = \frac{Size_{F_k} \cdot \#AVG\_Request_{S_i,F_k}}{BW_{(S_i,S_j)F_k} \cdot P_{S_j}}$$

- the file efficiency for a file F_k is defined as follows:

$$Efficiency_{F_k} = \frac{FileWeight_{F_k}}{Size_{F_k}} = \frac{\#AVG\_Request_{S_i,F_k}}{BW_{(S_i,S_j)F_k} \cdot P_{S_j}}$$

$$= \frac{\#Request_{S_i,F_k}}{(T_{Current} - T_{FirstRequest_{S_i,F_k}} + 1) \cdot BW_{(S_i,S_j)F_k} \cdot P_{S_j}}$$

Thus, it is clear that the ratio defined in the expression 3.3 has a low value when the value of $\#Request_{S_i,F_k}$ is low and the value of $(T_{Current} - T_{FirstRequest_{S_i,F_k}} + 1)$ is high.

In addition, we have:

- In the LFU strategy, the least accessed file in the local storage is deleted. In our proposal, this is also the case for each file F_k having a low value of $\#Request_{S_i,F_k}$.
- In the LRU strategy, the oldest replica in the local storage is removed. In our case, this is done for each the file F_k having a high value of $(T_{Current} - T_{FirstRequest_{S_i,F_k}} + 1)$.

4. Computational complexity of DPRSKP. In this section, we evaluate the theoretical computational complexity in the worst case of the two algorithms described in Sect. 3.

4.1. Complexity of the algorithm determining the candidate files for replication. Let m be the number of requested files by the site S_i. The complexity in the worst case of Algorithm 3.1.1 (*cf.* page 107) is function of this parameter.

- The lines 2, 3 and 4 are simple initializations and thus their complexity is in $O(1)$.

- The loop, between line 5 and line 8, has a complexity of $O(m)$.
- The line 9 has a complexity of $O(1)$.
- The loop, between line 10 and line 12, has a complexity of $O(m)$.

Thus, the complexity in the worst case of Algorithm 3.1.1 is $O(1) + O(m) + O(1) + O(m) = O(m)$.

4.2. Complexity of the files replication algorithm. Let n be the cardinality of the set $\mathcal{F} \cup \mathcal{F}_{ToReplicate}$. The complexity in the worst case of Algorithm 3.1.3 is function of this parameter.

- The block of instructions, between line 2 and line 4, has a complexity of $O(n)$.
- The loop, between line 6 and line 7, has a complexity of $O(n)$.
- The instruction of line 8 has a complexity of $O(n \cdot log(n))$.
- The instruction of line 9 has a complexity of $O(1)$.
- The loop, between line 10 and line 15, has a complexity of $O(n)$.
- The block of instructions, between line 5 and line 15, has thus a complexity of $O(n) + O(n \cdot log(n)) + O(1) + O(n) = O(n \cdot log(n))$.

Therefore, the complexity in the worst case of Algorithm 3.1.3 is $Max(O(n)\ , O(n \cdot log(n))) = O(n \cdot log(n))$.

It is however important to note that the majority of replication strategies have a polynomial-time complexity as shown in [1]. The experiments presented in the next section prove the effectiveness of our strategy DPRSKP.

5. Experiment setup and results. We use the OptorSim simulator [11] to evaluate and compare our strategy DPRSKP with the DR2 [44] and Best Client [37, 38] strategies. We compare our strategy with Best Client and DR2 since these strategies can be applied in P2P grids and are periodic strategies. In addition, Best Client and DR2 are two decentralized replication strategies that are based on the requests number of each file to choose the files needed to be replicated. Moreover and in order to show that our replacement strategy is better than LRU and LFU strategies, we compare DPRSKP to Best Client and DR2 which use LRU and LFU as replacement strategies. We note that we have made extensions to the OptorSim simulator in order to model dynamic grids. Indeed, we have added new classes and methods in the simulator to satisfy our needs. Moreover, we have made some changes to the configuration files to support the grid dynamicity.

5.1. Simulation environment. OptorSim is a simulator written in Java to evaluate the performance of replication strategies in data grids. It adopts the model of EU DataGrid Project[6] for the grid structure [12].

In OptorSim, a grid is composed of a User, a Resource Broker (RB) and several sites. A site contains a Computing Element (CE), a Replica Manager (RM) and a Storage Element (SE). These components are described as follows:

- User: the user submits jobs to the RB according to a pattern.
- Resource Broker: the RB handles the scheduling of the jobs to sites.
- Storage Element: each SE has a size in Mb and contains some files.
- Computing Element: each CE contains a given number of "worker nodes" with a given processing power and having for task to run jobs.
- Replica Manager: the RM performs the movement of data associated with jobs between sites. Moreover, the RM contains a Replica Optimiser (RO) responsible for both replica selection and the creation and deletion of replicas.

In OptorSim, the execution of a simulation is given as below:

1- Initially, the master files are placed in the SEs of the sites as specified in the configuration file. In our strategy, we assume the existence in the grid of a specific site called "Master Site" containing the master files. The Master Site is always connected to the grid.

2- The SEs register these master files in the Replica Catalog (RC).

3- The RB schedules each job to the appropriate site according to a scheduling strategy.

4- The CE of the chosen site in the previous step runs the job and selects the best replicas if the requested file is not present in the SE of the site.

5- The CE saves the transfer history of replicas.

[6]The European DataGrid Project. http://www.edg.org

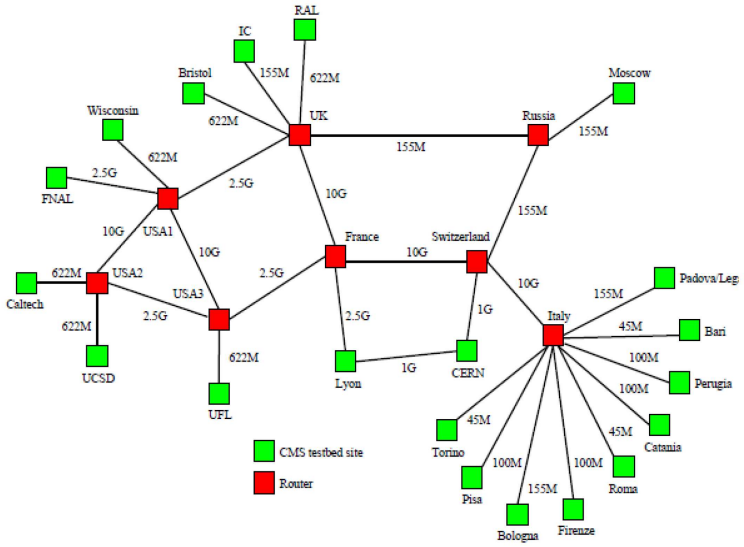

FIG. 5.1. *Testbed grid topology*

6- After a period (number of jobs), each site determines and replicates the candidate files for replication.
7- Each site updates the RC.

5.2. Grid topology. We base all simulation studies on the testbed used during a large scale production effort for the high energy physics experiment CMS [12]. The grid topology (see Fig. 5.1) consists of 20 sites in Europe and the USA. CERN and FNAL have a storage capacity of 100 Gb. Every other site has a CE and initially an empty storage element of a capacity of 50 Gb. CERN stores a master copy of each file. Table 5.1 describes the parameters used in the simulation experiments.

TABLE 5.1
Grid and jobs configurations

Parameter	Value
Size of a single file	1 Gb
Number of files	97
Access pattern	Sequential
Scheduling	access cost for current job + all queued jobs

5.3. Experimental results.

5.3.1. Performance evaluation metrics. In OptorSim, the experiments consist in submitting a variable number of jobs to the RB. At the end of each simulation, we get the response time, the number of remote file accesses, the number of local accesses and the ENU (the Effective Network Usage). This latter parameter [11, 21] is measured as follows:

$$ENU = \frac{N_{remote\ file\ accesses} + N_{file\ replications}}{N_{remote\ file\ accesses} + N_{local\ file\ accesses}}$$

ENU defines the ratio of files transferred to files requested. The parameter ENU is an important parameter in quantifying the effectiveness of any replication strategy in data grids. A low value indicates the efficiency of the adopted replication strategy [11, 24]. In addition, the response time is a good measure for evaluating the effectiveness of a replication strategy. Indeed, each job requests a set of files. If a file is present in the requesting

site, file transfer time is assumed to be zero, otherwise the file must be transferred from the best site containing it. Thus, the response time incorporates the time required to transfer files in addition to the time needed to process jobs. The best replication strategy will have the lowest response time. On the other hand, the number of remote file accesses and the number of local file accesses are two measures to evaluate the effectiveness of a replication strategy. Indeed, the number of remote file accesses represents the number of times where the sites read the necessary files from another one. Thus, the best replication strategy will have the lowest number of remote file accesses. However, the number of local accesses represents the number of times where the sites read the necessary files locally. Thus, the best replication strategy will have the greatest number of local file accesses.

5.3.2. Choice of the period. As mentioned above, DPRSKP is a decentralized periodic replication strategy, *i.e.*, DPRSKP is triggered at each period. The choice of this period is important as proven in [6, 7]. Indeed, the period greatly affects the efficiency of any periodic replication strategy. In this respect, a dynamic period, *i.e.* whose value changes according to the grid behavior, is shown to be much better than a static period, *i.e.* of constant value, whatever the adopted replication strategy. Thus, in our experiments, we use the model of dynamic period.

Now, we briefly sketch the formula used for computing the period (for more details, interested readers are referred to [6, 7]) . To take into account the replicas placement, the $(n+1)^{th}$ period T_{n+1} is defined based on the replications number (denoted $\#Replica_{T_n}$) made during the previous period T_n.

The period T_{n+1} is indeed given by the following formula:

$$T_{n+1} = \frac{T_n}{(\#Replica_{T_n}/T_n) + 1}$$

The next subsection depicts the obtained results which will be discussed in subsection 5.3.4.

5.3.3. Simulation results. This part presents the obtained results for the different metrics.

• *Evaluation of response time:* Table 5.2 shows the response time in milliseconds of each strategy and for different numbers of jobs. The last column indicates the percentage gain of our strategy compared to the other ones and is calculated as follows:

$$Gain\ in\ \% = \frac{Min(Value_{Best\ Client}, Value_{DR2}) - Value_{\text{DPRSKP}}}{Min(Value_{Best\ Client}, Value_{DR2})} \cdot 100$$

The same experiments of Table 5.2 are represented by histograms in Fig. 5.2.

• *Evaluation of the ENU parameter:* Table 5.3 and Fig. 5.3 give the ENU evaluation for the three strategies under the same conditions. The last column of Table 5.3 represents the percentage gain of DPRSKP compared to the other strategies and is calculated according to expression 5.3.3.

• *Evaluation of the number of remote file accesses:* Table 5.4 sketches the number of remote file accesses for each strategy and for different numbers of jobs. The last column computes the gain percentage of DPRSKP compared to Best Client and DR2 according to expression 5.3.3. The results of Table 5.4 are also represented through histograms in Fig. 5.4.

• *Evaluation of the number of local file accesses:* Table 5.5 and Fig. 5.5 show the number of local file accesses of each strategy and for different numbers of jobs. The last column in Table 5.5 indicates the percentage gain of DPRSKP compared to Best Client and DR2. This percentage is computed as follows:

$$Gain\ in\ \% = \frac{Value_{\text{DPRSKP}} - Max(Value_{Best\ Client}, Value_{DR2})}{Max(Value_{Best\ Client}, Value_{DR2})} \cdot 100$$

5.3.4. Discussion. It is clear from the obtained experimental results that the proposed DPRSKP strategy aims at increasing files availability, improving response time and reducing bandwidth consumption. Indeed, we notice, from Table 5.2 and Fig. 5.2, that the response time of our strategy is always better than that of the other strategies. In addition, our strategy offers an ENU much better than Best Client and DR2 whatever the

TABLE 5.2
Response time in milliseconds and associated percentage gain of DPRSKP *compared to Best Client and DR2*

Number of jobs	Best Client	DR2	DPRSKP	Gain (%)
100	3 580	3 070	2 989	2.64
500	8 790	8 623	6 417	25.58
1 000	18 067	16 847	14 897	11.57
1 500	25 922	23 663	19 138	19.12
2 000	41 650	49 504	32 653	21.60
2 500	54 396	76 713	50 203	7.71
3 000	69 500	72 406	56 687	18.44

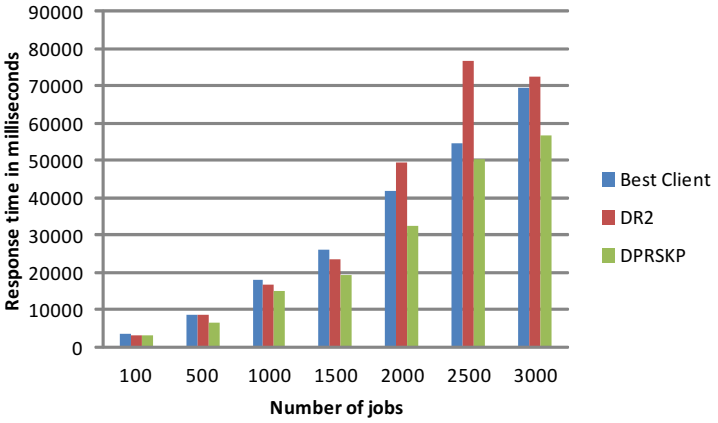

FIG. 5.2. *Response time for the different replication strategies*

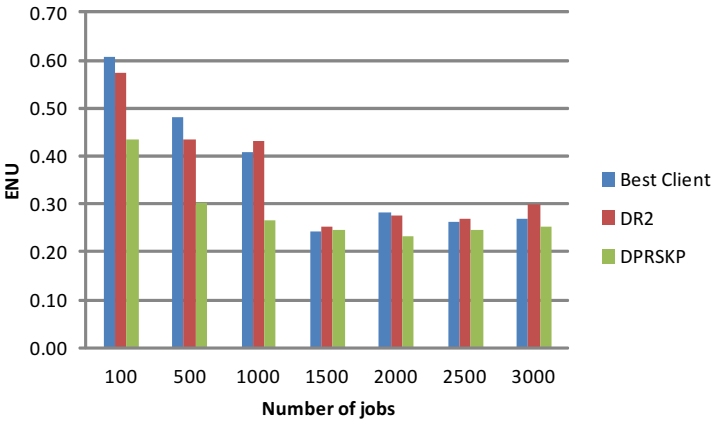

FIG. 5.3. *ENU for the different replication strategies*

number of jobs. Indeed, the obtained average gain of our strategy compared to both others is about 20.74% (*cf.* Table 5.3). On the other hand, we remark from Table 5.4 that the number of remote file accesses of DPRSKP is always lower than Best Client and DR2 strategies. We notice, also from Table 5.5 and Fig. 5.3, that the number of local file accesses of our strategy is always higher than the other strategies. All these obtained results are explained by the fact that the files deletion in the case of a shortage space is not automatic in our approach.

TABLE 5.3
ENU and associated percentage gain of DPRSKP *compared to Best Client and DR2*

Number of jobs	Best Client	DR2	DPRSKP	Gain (%)
100	0.61	0.58	0.44	24.17
500	0.48	0.43	0.30	30.68
1 000	0.41	0.43	0.27	35.02
1 500	0.24	0.25	0.25	-1.38
2 000	0.28	0.27	0.23	15.20
2 500	0.26	0.27	0.25	6.60
3 000	0.27	0.30	0.25	6.22

TABLE 5.4
Number of remote file accesses and associated percentage gain of DPRSKP *compared to Best Client and DR2*

Number of jobs	Best Client	DR2	DPRSKP	Gain (%)
100	801	599	502	16.19
500	3 057	2 437	1 710	29.83
1 000	4 973	5 048	3 291	1.26
1 500	4 509	4 656	4 452	18.33
2 000	6 776	6 281	5 534	11.89
2 500	8 042	8 103	7 328	8.88
3 000	10 041	9 681	9 233	4.63

Indeed, contrary to DR2 and Best Client strategies which always removes files in the case there is a shortage in the storage space at the selected site to replicate a new file, DPRSKP cannot replicate candidate files if their added values are less than those existing in a site and then it preserves those existing files in the site. In other words, DPRSKP deletes files from a given site only if they are qualified as not beneficial in the future and replaces them with more beneficial files. All these results confirm the efficiency of our strategy.

6. Conclusions and future work. Data grids are large scale systems that connect a collection of several machines and storage resources distributed around the world. In such systems, an optimized management and access of distributed resources is the primary purpose. Data grids aggregate a collection of distributed resources placed around the world to enable data intensive applications to share data and resources. As the data is the most important resource in data grids, their quick access time and efficient management have become challenging tasks. Replication is a technique used to ensure these tasks. In this paper, we proposed a new decentralized periodic replication strategy, called DPRSKP, formulated according to the Knapsack problem with storage constraints. Our strategy takes into account the dynamicity of grid sites. Indeed, the dynamicity of sites is an important challenge in grids. We designed first an algorithm which selects the best candidate files for replication based on the requests number and the copies number of each file. We then proposed an algorithm which chooses the candidate files for deletion when there is not enough space to accommodate new files. This algorithm introduces the file efficiency of a file F_k, $Efficiency_{F_k}$, which is equal to $\frac{\#AVG\_Request_{S_i,F_k}}{BW_{(S_i,S_j)F_k} \cdot P_{S_j}}$. As a result, the file efficiency does not depend on the file size. Thus, contrary to the majority of work in the literature, all large and small files have the same chance to be placed in the site. We believe that this notion of file efficiency is at the root of the good performances of our strategy.

In future work, we plan to validate our strategy by deploying the replication algorithm in a real grid environment. In addition, in practice, the files are required simultaneously with others, *i.e.*, there exists a

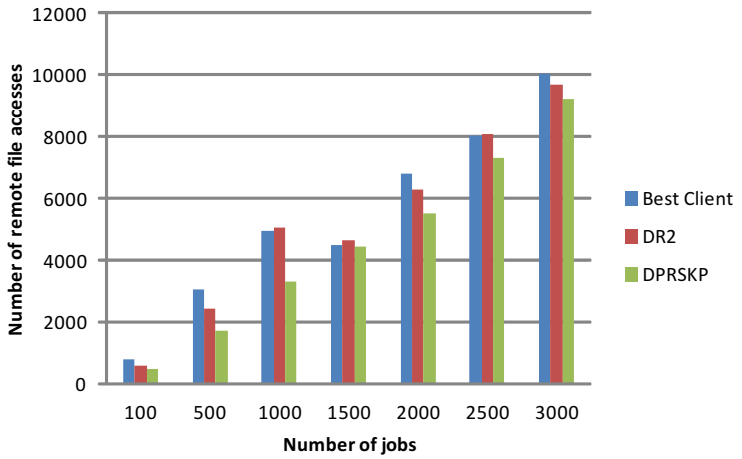

FIG. 5.4. *Number of remote file accesses for the different replication strategies*

TABLE 5.5
Number of local file accesses and associated percentage gain of DPRSKP *compared to Best Client and DR2*

Number of jobs	Best Client	DR2	DPRSKP	Gain (%)
100	596	564	823	38.09
500	3 467	3 374	4 387	26.54
1 000	7 501	6 846	9 721	29.60
1 500	14 964	14 801	15 527	3.76
2 000	18 361	18 636	19 431	4.27
2 500	23 864	23 886	23 957	0.30
3 000	28 641	28 309	28 808	0.58

strong correlation between requested files. Thus, it is possible to investigate replication strategies which consider as granularity a set of correlated files instead of a single file. Moreover, the grid files can be modified by some applications. Since these data files are replicated in grid sites, it is necessary to update them to ensure maintaining coherent copies. It is then important to look for an efficient way for extending the replication strategies proposed in the literature in order to be able to take into consideration file consistency.

REFERENCES

[1] T. AMJAD, M. SHER, AND A. DAU, *A survey of dynamic replication strategies for improving data availability in data grids*, Future Generation Computer Systems, 28 (2012), pp. 337–349.

[2] M. BEIGREZAEI, A. T. HAGHIGHAT AND A. KHADEMZADEH, *A new fuzzy based dynamic data replication algorithm in data grids*, Journal of Basic and Applied Scientific Research, 3 (2013), pp. 350–358.

[3] W. BELL, D. CAMERON, L. CAPOZZA, A. MILLAR, K. STOCKINGER, AND F. ZINI, *Simulation of dynamic grid replication strategies in OptorSim*, Lecture Notes in Computer Science, 2536 (2002), pp. 46–57.

[4] ———, *OptorSim - a grid simulator for studying dynamic data replication strategies*, International Journal of High Performance Computing Applications, 17 (2003), pp. 403–416.

[5] F. BEN CHARRADA, H. CHETTAOUI, AND R. MILI, *A new replication strategy for data grids*, in Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Networks, 2011, pp. 184–189.

[6] F. BEN CHARRADA, H. OUNELLI, AND H. CHETTAOUI, *Dynamic period vs static period in data grid replication*, in Proceedings of the 3rd International Workshop on Simulation and Modeling of Emergent Computational Systems, 2010, pp. 565–568.

[7] ———, *A model for dynamic period in data grid replication*, International Journal of Modeling and Optimization, 3 (2013), pp. 163–166.

[8] ———, *An efficient replication strategy for dynamic data grids*, in Proceedings of the 5th International Conference on P2P,

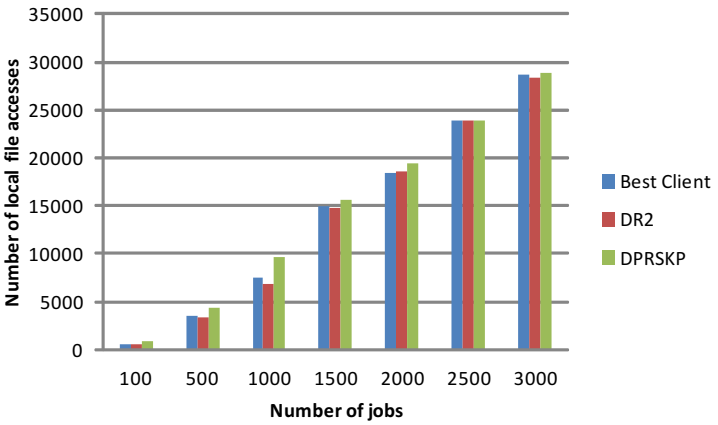

FIG. 5.5. *Number of local file accesses for the different replication strategies*

Parallel, Grid, Cloud and Internet Computing, 2010, pp. 50–54.

[9] ———, *An efficient replica placement strategy in highly dynamic data grids*, International Journal of Grid and Utility Computing, 2 (2011), pp. 156–163.

[10] M. BSOUL, A. AL-KHASAWNEH, Y. KILANI, AND I. OBEIDAT, *A threshold-based dynamic data replication strategy*, The Journal of Supercomputing, (2010).

[11] D. CAMERON, R. CARVAJAL-SCHIAFFINO, J. FERGUSON, A. MILLAR, C. NICHOLSON, K. STOCKINGER, AND F. ZINI, *OptorSim v2.1 installation and user guide*, tech. report, 2006.

[12] D. CAMERON, A. MILLAR, C. NICHOLSON, R. CARVAJAL- SCHIAFFINO, F. ZINI, AND K. STOCKINGER, *Analysis of scheduling and replica optimisation strategies for data grids using OptorSim*, Journal of Grid Computing, 2 (2004), pp. 57–69.

[13] R. CHANG, H. CHANG, AND Y. WANG, *A dynamic weighted data replication strategy in data grids*, in Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp. 414–421.

[14] H. CHETTAOUI AND F. BEN CHARRADA, *A Decentralized Periodic Replication Strategy based on Knapsack Problem*, in Proceedings of the 13th ACM/IEEE International Conference on Grid Computing, 2012, pp. 3–11.

[15] U. CIBEJ, B. SLIVNIK, AND B. ROBIC, *The complexity of static data replication in data grids*, Parallel Computing, 31 (2005), pp. 900–912.

[16] T.H. CORMEN, C.E. LEICERSON, R.L. RIVEST, AND C. STEIN, *Introduction to Algorithms*, Third Edition, MIT Press, 2009.

[17] Z. CUI, D. ZUO, AND Z. ZHANG, *Based on support and confidence dynamic replication algorithm in multi-tier data grids*, Journal of Computational Information Systems, 9 (2013), pp. 3909–3918.

[18] N. DANG, S. HWANG, AND S. LIM, *Improving job scheduling performance with dynamic replication strategy in data grids*, Lecture Notes in Computer Science, 4671 (2007), pp. 194–199.

[19] N. DANG AND S. LIM, *Combination of replication and scheduling in data grids*, International Journal of Computer Science and Network Security, 7 (2007), pp. 304–308.

[20] D. DULLMANN, W. HOSCHEK, J. JAEN-MARTINEZ, B. SEGAL, A. SAMAR, H. STOCKINGER, AND K. STOCKINGER, *Models for replica synchronisation and consistency in a data grid*, in Proceedings of IEEE International Symposium on High Performance Distributed Computing, 2001, pp. 67–75.

[21] H. E. AL MISTARIHI AND C. YONG, *Replica management in data grid*, International Journal of Computer Science and Network Security, 8 (2008), pp. 22–32.

[22] H. KELLERER, U. PFERSCHY, AND D. PISINGER, *Knapsack problems*, Springer, 2004.

[23] H. LAMEHAMEDI, Z. SHENTU, AND B. SZYMANSKI, *Simulation of dynamic data replication strategies in data grids*, in Proceedings of the Heterogeneous Computing Workshop, 2003.

[24] H. LAMEHAMEDI AND B. SZYMANSKI, *Decentralized data management framework for data grids*, Future Generation Computer Systems, 23 (2007), pp. 109–115.

[25] M. LEE, F. LEU, AND Y. CHEN, PFRF*: An adaptive data replication algorithm based on star-topology data grids*, Future Generation Computer Systems, 28 (2012), pp. 1045–1057.

[26] M. LEI AND S. VRBSKY, *A data replication strategy to increase data availability in data grids*, in Proceedings of the International Conference on Grid Computing and Applications, 2006, pp. 221–227.

[27] M. LEI, S. VRBSKY, AND X. HANG, *An on-line replication strategy to increase availability in data grids*, Future Generation Computer Systems, 24 (2008), pp. 85–98.

[28] N. MANSOURI AND GH. DASTGHAIBYFARD, *A dynamic replica management strategy in data grid*, Journal of Network and Computer Applications, 35 (2012), pp. 1297–1303.

[29] ———, *Improving data grids performance by using modified dynamic hierarchical replication strategy*, Iranian Journal of Electrical & Electronic Engineering, 10 (2014), pp. 27–37.

[30] S. MARTELLO AND P. TOTH, *Knapsack problems: Algorithms and computer implementations*, John Wiley & Sons, 1990.

[31] M. MEHRABAN, A. KHADEMZADEH, AND M. SALEHNAMADI, *A prediction-based replica replacement strategy in data grid*,

Journal of Basic and Applied Scientific Research, 2 (2013), pp. 928–939.
[32] J. Montagnat, V. Breton, and I. Magnin, *Using grid technologies to face medical image analysis challenges*, in Proceedings of the Workshop on Biomedical Computations on the Grid, 2003, pp. 588–593.
[33] S. Park, J. Kim, Y. Ko, and W. Yoon, *Dynamic data grid replication strategy based on internet hierarchy*, in Proceedings of the International Workshop on Grid and Cooperative Computing, 2003, pp. 838–846.
[34] R. Rahman, K. Barker, and R. Alhajj, *Replica placement in data grid: Considering utility and risk*, in Proceedings of the International Conference on Information Technology: Coding and Computing, 2005, pp. 354–359.
[35] ———, *Replica placement design with static optimality and dynamic maintainability*, in Proceedings of the IEEE International Symposium on Cluster Computing and Grid, 2006, pp. 434–437.
[36] ———, *Replica placement strategies in data grid*, Journal of Grid Computing, 6 (2008), pp. 103–123.
[37] K. Ranganathan and I. Foster, *Design and evaluation of dynamic replication strategies for a high-performance data grid*, in Proceedings of the International Conference on Computing in High Energy and Nuclear Physics, 2001.
[38] ———, *Identifying dynamic replication strategies for a high performance data grid*, in Proceedings of the Second International Workshop on Grid Computing, 2001, pp. 75–86.
[39] K. Ranganathan, A. Iamnitchi, and I. Foster, *Improving data availability through dynamic model-driven replication in large peer-to-peer communities*, in Proceedings of Global and Peer-to-Peer Computing in Large Scale Distributed Systems Workshop, 2002, pp. 376–381.
[40] Q. Rasool, J. Li, G. Oreku, and E. Munir, *Fair-share replication in data grid*, Information Technology Journal, 7 (2008), pp. 776–782.
[41] M. Sakarovitch, *Optimisation Combinatoire*, Hermann Edition, 1984.
[42] K. Sashi and A. Thanamani, *Dynamic replication in a data grid using a modified* BHR *region based algorithm*, Future Generation Computer Systems, 27 (2011), pp. 202–210.
[43] Y. Shi, A. Shortridge, and J. Bartholic, *Grid computing for real-time distributed collaborative geoprocessing*, in Proceedings of the Symposium on Geospatial Theory, Processing and Applications, 2002.
[44] P. Suri and M. Singh, DR2*: A two-stage dynamic replication strategy for data grid*, International Journal of Recent Trends in Engineering, 2 (2009), pp. 201–203.
[45] H. Takeuchi, T. Kondo, Y. Koyama, and J. Nakajima, *Vlbi@home-vlbi correlator by GRID computing system*, in Proceedings of the International VLBI Service for Geodesy and Astrometry, 2004, pp. 200–204.
[46] P. Vashisht, R. Kumar and A. Sharma, *Efficient dynamic replication algorithm using agent for data grid*, The Scientific World Journal Volume (2014), 10 pages, http://dx.doi.org/10.1155/2014/767016
[47] G. von Laszewski, M. Su, J. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Thiebaux, M. Rivers, S. Wang, B. Tieman, and I. McNulty, *Real-time analysis, visualization, and steering of microtomography experiments at photon sources*, in Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing, 1999.
[48] W. Zhao, X. Xu, Z. Wang, Y. Zhang, and S. He, *Improve the performance of data grids by value-based replication strategy*, in Proceeding of the International Conference on Semantics, Knowledge and Grids, 2010, pp. 313–316.

Edited by: Dana Petcu
Received: Feb 9, 2014
Accepted: Mar 11, 2014