Front. Comput. Sci., 2021, 15(2): 152501
https://doi.org/10.1007/s11704-019-9099-8

RESEARCH ARTICLE

Hierarchical data replication strategy to improve performance in cloud computing

Najme MANSOURI (✉)[1,2], Mohammad Masoud JAVIDI[1,2], Behnam Mohammad Hasani ZADE[1,2]

1 Department of Computer Science, Shahid Bahonar University of Kerman, Kerman 76169-14111, Iran
2 Mahani Mathematical Research Center, Shahid Bahonar University of Kerman, Kerman 76169-14111, Iran

Abstract Cloud computing environment is getting more interesting as a new trend of data management. Data replication has been widely applied to improve data access in distributed systems such as Grid and Cloud. However, due to the finite storage capacity of each site, copies that are useful for future jobs can be wastefully deleted and replaced with less valuable ones. Therefore, it is considerable to have appropriate replication strategy that can dynamically store the replicas while satisfying quality of service (QoS) requirements and storage capacity constraints. In this paper, we present a dynamic replication algorithm, named hierarchical data replication strategy (HDRS). HDRS consists of the replica creation that can adaptively increase replicas based on exponential growth or decay rate, the replica placement according to the access load and labeling technique, and finally the replica replacement based on the value of file in the future. We evaluate different dynamic data replication methods using CloudSim simulation. Experiments demonstrate that HDRS can reduce response time and bandwidth usage compared with other algorithms. It means that the HDRS can determine a popular file and replicates it to the best site. This method avoids useless replications and decreases access latency by balancing the load of sites.

Keywords cloud computing, data replication, multi-tier architecture, simulation, load balance

1 Introduction

Nowadays, cloud computing environment is a common computing model, which is a significant phase in the expansion of an increasing number of distributed applications [1–3]. Figure 1 shows the features of next-generation scientific study, the e-Science needs imposed by different characteristics, and main enabling technologies. Figure 1 presents that web services, workflow, semantic web, Grid computing, and Cloud computing (e.g., SaaS, PaaS, and IaaS) are some of the major enabling digital facilities for presenting useful e-Infrastructure and application-oriented platforms [4].

With increasing the requirements of users, the size of data files, and request to obtain knowledge from the huge amount of data, the utility of plain storage elements that are easy to control at a large scale increases. Data is a key element in the Cloud system, and so a suitable, well maintained, and efficient platform must be provided to guarantee the dependability and availability of data. Cassandra and Hive in Facebook, and HBase in Stream are the most popular examples of Cloud management systems [5–7]. The well-defined data management techniques have to present data reliability, availability, and durability at different rates. In addition, a platform with a data replication strategy can present better features and access time, which are serious from the perspective of the costumers [8].

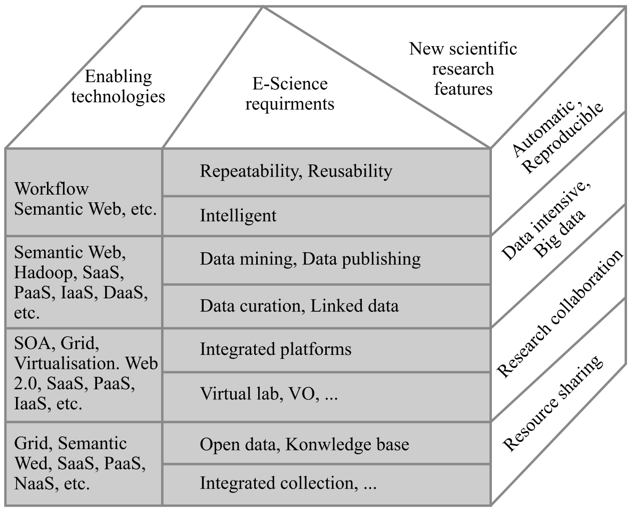
Enabling technologies
E-Science requirments
New scientific research features
Workflow Semantic Web, etc.
Repeatability, Reusability
Intelligent
Semantic Web, Hadoop, SaaS, PaaS, IaaS, DaaS, etc.
Data mining, Data publishing
Data curation, Linked data
SOA, Grid, Virtualisation. Web 2.0, SaaS, PaaS, IaaS, etc.
Integrated platforms
Virtual lab, VO, ...
Grid, Semantic Wed, SaaS, PaaS, NaaS, etc.
Open data, Konwledge base
Integrated collection, ...
Automatic, Reproducible
Data intensive, Big data
Research collaboration
Resource sharing

Fig. 1 A summary of e-Science requirements and important enabling technologies [4]

• **Data replication in cloud computing**

Replication in cloud computing can be introduced as the storing multiple replicas of data across different data centers. Replica management challenge is a hot research area in Cloud computing environment [9]. Replica management technique can reduce network delay and the bandwidth usage of remote data access, improve the load-balancing of network, and enhance the safety of data, by storing important replicas at suitable data centers.

Figure 2 indicates the generalized replication layer in distributed system [10]. In general, replication in distributed sys-

Received March 20, 2019; accepted November 6, 2019

E-mail: najme.mansouri@gmail.com

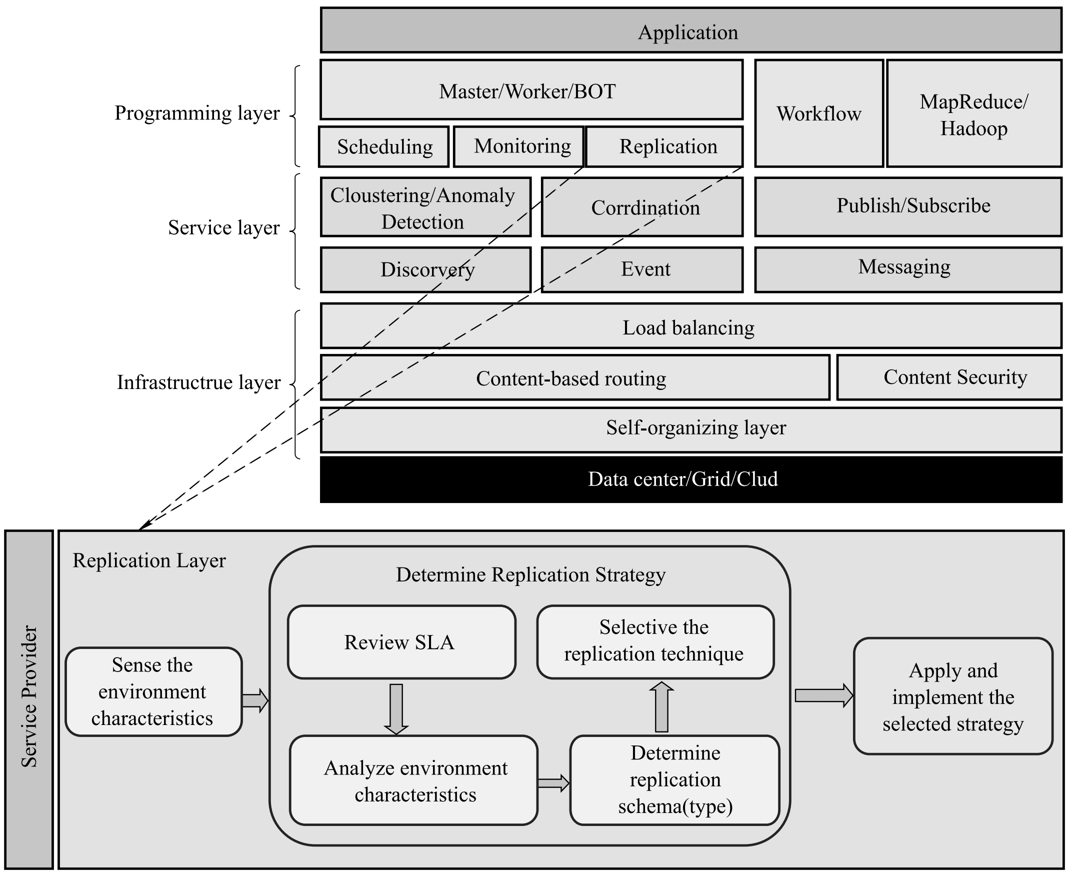

Fig. 2 Generalized replication layer [10]

tem consists of four components as users, provider of service, replication actor, and environment. The provider of service communicates with replication layer and environment to improve the quality of services. The replication actor updates and enhances the QoS factors based on the SLA by creating replicas as needed [11,12]. The main steps of replica management are replica creation, selection, placement, and replacement. According to the various creations, selections and replacement algorithms, replica management methods can be classified into the simple replication strategy, the hierarchical model based replication method, and the economic model based replication schema [13].

A common example of data replication is storing several replicas of a key document to recover in case of any failure or damage to the primary replica. Similarly, business communities, governments, and other companies copy the critical financial, personal, and legal data to assurance security, accessibility, and durability. The Google File Services (GFS) [14] and Amazon Simple Storage Service are the most common examples of data storage services that apply replication techniques [15].

The data replication strategies can generally be classified as static and dynamic [16]. In static replication, a replica stored until it is removed by users or its duration is finished [17,18]. The other type of replication is dynamic. In this type of replica generation, deletion and management are done automatically. Dynamic data replication algorithms can adapt to the variation in the user behavior [19–26].

Replicating data on a distributed platform is very difficult due to the dynamicity of sites that can compromise availability. Although, replica management has been previously investigated on the replica management problem, very few of the existing algorithms take a holistic view of the different steps and the benefits of replication. They commonly adopt data replication to present high availability, fast response, and high efficiency. These performance parameters obtain better as the number of replicas increase. However, the most significant thing they relinquished is that data replication does not come for free. Due to dynamic nature of Cloud, the selected location that stores replicas may not be the best location for fetching replicas in subsequent intervals. Hence, we propose a replica placement strategy that relocates replicas to the different locations if the performance parameter remarkably degrades. It is implemented using a new hierarchical labeling technique. Various benefits of the multi-tier structure for both search performance and scalable management advantages over uniform representations are highlighted via the results from simulation experiments.

The most previous replication methods focus on reducing data retrieval time. But the limited storage of site can store only the part of all data files since large scale data files are generated in Cloud environment. Data access patterns have been model based on different distribution and the popularity of file is introduced as how often users request a file. Ranganathan and Foster [27] discussed about the pattern of file requests and indicated different locality properties, including:

Temporal locality means that a currently accessed file has a high possibility to be accessed again.

Geographical locality means that a currently accessed file by a user has a high possibility to be accessed by nearby users.

Spatial Locality means that files near a currently accessed file has a high possibility to be accessed.

In [28] authors proposed Bandwidth Hierarchical Replication (BHR) algorithm. BHR strategy stores replicas as many times as possible within a region that has high bandwidth between sites.BHR strategy reduces network congestions in Grid to improve access time. Based on that work, the proposed strategy (HDRS) considers the network-level locality that shows the needed files are available in local Cloud site.

We list the main contributions as follows:

- Design the multi-tier structure for data replication in Cloud environment and provide flexible and scalable management for the large number of files. So, the network bandwidth will be efficiently applied. Since most of the data transfers only use local resources.
- Apply the labeling structure which is an interpolation of its parent label and its local order. Therefore, we can easily determine the precise physical location of site, the location of its parent, siblings, ancestor, load, and the number of hops.
- Determine the popular files based on the exponential decay to estimate the next number of access for data file in Cloud.
- Propose replica placement algorithm based on the network level locality. So, the resource usage and execution time are improved.
- Present replica replacement method according to the number of access for the file in future time and file size.
- Compare the proposed method with several data replication techniques in terms of average response time, effective network usage, replication frequency, load variance and storage usage.

The rest of this paper is organized as follows. Section 2 discusses the earlier works on data replication for Cloud environment. Section 3 describes the system model. Section 4 introduces the proposed data replication algorithm. Section 5 evaluates the performance of HDRS and current replication strategies in Cloud system. Finally, Section 6 concludes the paper and points out some future works.

2 Related works

Data replication approach in the Cloud system has currently received a lot of attention [29]. A handful of works are available in the literature for data replication algorithm in data Grid and Cloud computing environment. Here, the review of recent researches from these topics is provided. Early works mostly considered the performance parameter in data grid environment. Mansouri [9] presented a combined replica placement according to the different factors such as the probability of failure, latency, average service time, and load balancing. In addition, the proposed replication strategy replaced unnecessary replicas based on different parameters such as file availability, last access time, the size of file, the number of file access. But the main weakness is that it didn't consider the concept of network locality. But, in the proposed strategy (HDRS), we enhance local accesses with particular attention to the locality factor. Moreover, it stores replicas according to the access load and labeling technique.

Khanli et al. [30] introduced a new predictive hierarchical fast spread (PHFS) in hierarchical data Grid. PHFS algorithm enhances the CFS (common fast spread) by adding predictive technique and spatial locality attention. PHFS algorithm could replicate files in various layers of hierarchical Grid and improve local access and storage usage. But, this algorithm did not consider adaptive thresholds and time interval for various systems with different applications.

Mansouri and Dastghaibyfard [26] presented a new enhanced dynamic hierarchical replication (EDHR) strategy for data Grid environment. EDHR strategy determines the suitable location for replica placement according to the number of request and the time of last request. In addition, it replaces replicas by considering the economic model to provide sufficient space for new replicas. In the sequel, it chooses the appropriate replica location for the request of user based on the request queue and retrieval time. But the main drawback of EDHR strategy is that it didn't determine the popular file for replication.

In addition, data replication has been a well-known research issue in Cloud for reducing user waiting time. Sun et al. [31] presented a replication algorithm for hierarchal Cloud systems. Their work focused on the temporal locality [32]. Based on temporal locality, a more recently accessed data will be requested again in the future [33]. Therefore, a popular file is found by studying the users' access to the file. After the popularity of the data overpasses a dynamic threshold, the replica creation process will be done. Based on the system parameters such as availability and failure probability, the number of replicas will be set. A new replica will be stored on the nearby locations for site which triggers the most requests for that particular file. This replication strategy has been combined with the checkpoint technique to present a new method called DAFT (dynamic adaptive fault-tolerance) [34]. But they did not address load balancing issue.

Zhang et al. [35] proposed an intelligent optimization strategy called Plant Growth Simulation Algorithm (PGSA) for Cloud environment. PGSA takes in to account network status, the load of storage element and the historical information of file. It considers the plant growth environment as the problem solution space and specifies the corresponding plant morphactin concentration based on the goal function value of the problem solution [36]. PGSA models the replica's fitness to the user access with morphactin concentration and chooses the appropriate replica by adjusting and evaluating its morphactin concentration value. The simulation results with CloudSim presented that plant growth algorithm improves replica utilization and average access time. Further, they only emphasized on improvement of the overall performance and the authors did not investigate the fault tolerance in data centers.

Long et al. [37] presented a Multi-objective Optimized Replication Management algorithm, named MORM, for improving data availability in Cloud.MORM is based on the artificial immune method. It considers various system and data parameters such as average file unavailability, average service time, load variance, energy consumption, and average access latency to find the relationship among replica number, replica layout, and their performances. Replicas are located among data centers with respect to the five parameters. The number of replicas is found for each data file to reach the optimal objective value. They simulated the proposed strategy using the extended CloudSim and MATLAB software. The experimental results presented that the MORM strategy is able to improve file availability and system load balancing ability, reduce mean service time, and improve energy consumption in the Cloud environment.

Lou et al. [38] presented a data replication strategy based on individual QoS sensitivity constraints with widely consideration of the system load, transmission cost, user's QoS objective, and historical evaluation information. QoS based Replication Strategy (QRS) checks whether the local site has the needed file or not. If the needed file is available then QRS will be applied directly. Otherwise, the local site checks characteristics of the replica and the evaluation information is written to the list history of the replica site. According to user's QoS goal, it computes the standard weight *w* of availability, timeliness and reliability. Then, it obtains the similarity between the recent requesting environment and the historical replica credibility evaluated environment. Finally, it chooses the replica which has the maximum credibility similarity environment. Simulation results demonstrated that the selection approach can achieve to the user's objectives.

Kumar et al. [39] developed a workload-aware data replication strategy (SWORD) which improves the resource utilization of Cloud system. SWORD presents the expected workload with a hypergraph. Also, it uses a partitioning strategy that minimizes the mean query span, i.e., the average number of machines used in the answering of a request or a transaction. They empirically adjusted the use of query span as the parameter to optimize the transactional workloads. They also presented a replica placement method by showing connections for different well-studied graph theoretic notions. They used the fine-grained quorums to decrease the query spans and thus the overall throughput is enhanced. SWORD can seamlessly manage different workloads which is a fundamental requirement for Cloud systems. The experiment evaluation with two different types of workloads demonstrated the potency of the framework. The main problem of SWORD strategy is that it considers only average query span in replication process.

Some of the previous replication strategies created replicas from the perspective of access latency. For example, Tos et al. [40] presented PErformance and Profit oriented data Replication strategy, version 2 (PEPRv2) in Cloud environment. PEPRv2 calculates the response time for each query and determines whether it can guarantee the quality of service or not. If the predicated response time is higher than the response time of SLO, then it replicates the particular file. Moreover, PEPRv2 estimates the profit value based on storage and network usage. In addition, the cold replicas (i.e., that are not necessary) are removed to enhance the provider benefits. The analytical results with CloudSim demonstrated that PEPRv2 guarantees the predefined response time with low Cloud resource usage.

Wu et al. [41] proposed storage provider aggregating networked store (SPANStore) in Cloud environment. SPANStore presents an integrated view of storage services for different distributed data centers. The main goal of SPANStore is decreasing the cost of latency-sensitive application. Therefore, it takes into account the estimated workload, the latency of applications in SPANStore, the number of fails, consistency degree, and the cost model for storage services

Vulimiri et al. [42] presented a new discussion about Wide-Area Big Data (WABD) in executing query across different sites to reduce bandwidth consumption while satisfying the particular constraints. It considers network-centric methods for wide-area setting such as the optimization of joint query and classical query planning. The minimization of bandwidth consumption is the first goal of WABD and does not try to reduce the latency of execution.

Wei et al. [43] introduced a cost-based dynamic replication management method (CDRM). They can find the appropriate relation among the number of replicas and the level of availability that is defined by user. Then, they stored replicas in the site according to the capacity and the blocking probability of site. The experiments demonstrated that CDRM is cost-effective and outperforms the default replication management of HDFS in the load balancing of system. But, if storage does not have sufficient capacity, then CDRM does not use the replica replacement strategy.

Edwin et al. [44] proposed an efficient and improved multi-objective optimized replication management (EIMORM) that is based on MORM algorithm. EIMORM has two main differences compared with MORM algorithm. Firstly, it considers the cost of replication during replica placement process. Secondly, it assigns a weight to data files for determining popular files based on the last access time. The simulation results indicated that the proposed algorithm can reduce total cost especially for the large number of tasks. But, EIMORM strategy did not consider the replica replacement that is the crucial step when the storage space is full.

Azimi [45] introduced a Bee Colony based Data Replication (BCDR) algorithm for Cloud environment. BCDR stores the number of requests for each file in all sites. In calculation of fitness, the probability of existing files in sites is considered. Worker bees search all space for finding the best site based on the number of requests and determine a suitable site by considering its fitness for replica transferring. In other words, the site that has the highest number of requests for a file is more likely to save that file. The main weakness of this approach is that it ignores other important factors such as load during replication process.

Mansouri and Javidi [22] presented a data replication algorithm named prefetching-aware data replication (PDR). PDR tries to find the most related files based on the file access history and pre-fetches them. It discovers dependency among files by building the dependency matrix for all jobs and sites. Finally, PDR algorithm replaces replicas by designing fuzzy inference system. The simulation results with CloudSim demonstrated that PDR can reduce the number of communications and replication frequency. Neglecting the replica placement is the main disadvantage of the PDR strategy.

The main differences among the mentioned replication methods and the presented strategy can be summary as follows. 1) We apply a tree structure to show a hierarchical Cloud. The proposed method labels the structure sites using a labeling schema which is considering in XML databases processing [46]. It prevents inessential replication and provides reasonable performance by balancing the load of sites. 2) The proposed method considers the exponential growth and decay, and file age in determining files that must be replicated. 3) Due to the finite storage element, a powerful replica replacement strategy is essential to replace useless replicas with the vital replicas. HDRS deletes files that are the least likely to be required in the near future.

3 System model

Figure 3 indicates the tree structure that has three levels. Our structure is hierarchical and each link has a particular bandwidth. The Coordinator site that provides powerful computing resources is placed in Level 0. It controls all the regions and computes the distances between the site which requests a file, and the site (placed in various regions) that contains the requested file. The Level 2 are regions that are connected through low bandwidth. Each region comprises Regional Header (i.e., sites that have a huge storage capacity) and various sites (Level 3). The bandwidth capacity of intra-region is more abundant than the bandwidth of inter-region [40]. Figure 4 shows the main components of our model. They are described as following:

1) Resource Broker schedules jobs to the sites. The site can process the job by accessing the required files.

2) Replica Catalog determines the physical locations of data files. It has the databases that store the mapping between Logical File Names (LFN), Physical File Name (PFN), and labeled tree. Also Replica Information saves the other file characteristics such as file size.

3) Replica Manager has four important elements:

- Resource Manager allocates resources such as network and storage with respect to the reservations.
- Replica Selector finds a suitable replica based on the transfer time that is found based on:

$$Transfer\_Time_{ij}\,(Sec) = \frac{Replica\_Size\,(Mb)}{Bandwidth_{ij}\,(Mb/Sec)}, \quad (1)$$

  where *Size_Replica* shows the size of replica and *Bandwidth*$_{ij}$ indicates the bandwidth between site i and site j.
- Catalog Service cooperates with Replica Catalog to collect the knowledge from databases that has the mapping of LFNs and PFNs. In addition, database contains the labeling tree in which PFNs are assigned to labels.
- Replicator stores/replicates the replica in the appropriate location based on the particular data replication strategy.

Local Replica Catalog and Local Replica Manager are distributed on every sites. Region Headers contains Replica Manager and Replica Catalog.

4 Hierarchical data replication strategy (HDRS)

A replication algorithm consists of three major decisions to make: which file should be copied, where the new replica

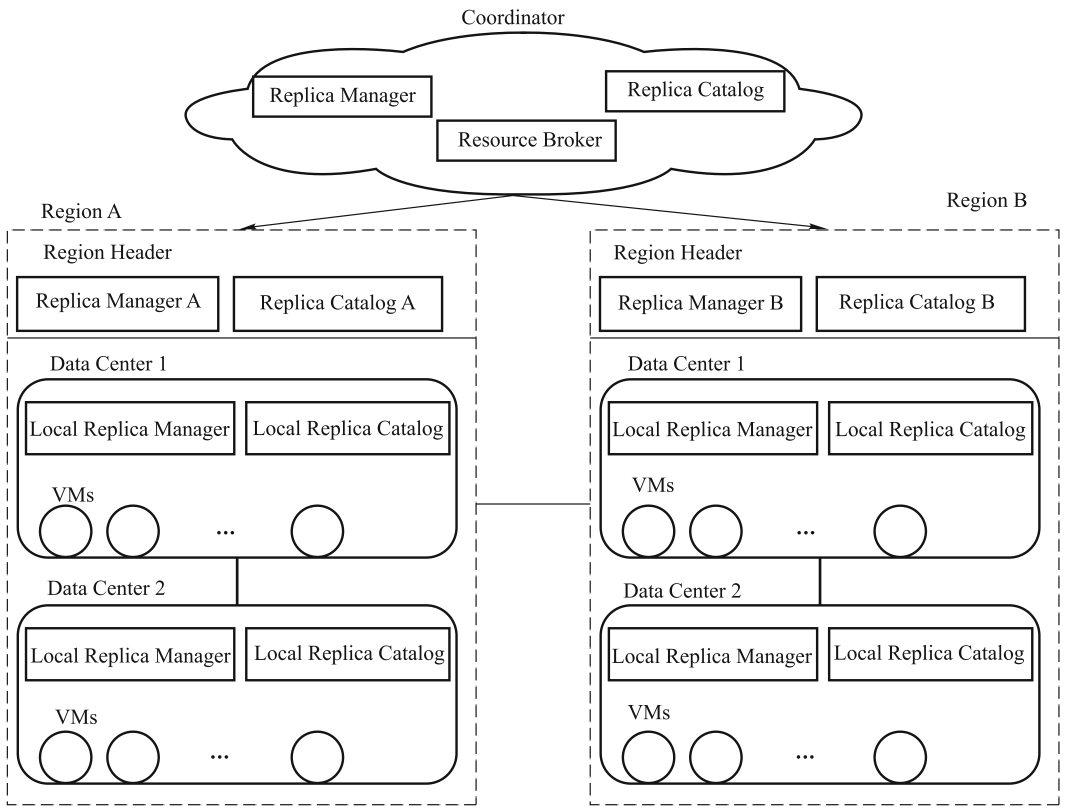

Fig. 3 Hierarchical cloud topology

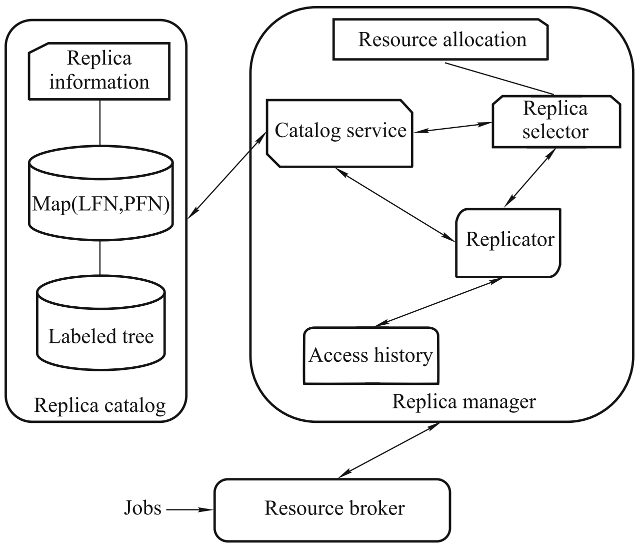

Fig. 4 Overview on different components

should be placed, and which file should be replaced. The answers to these questions lead to the different dynamic data replication methods.

4.1 Which file should be replicated?

When a job wants to execute but the required file is not available in the local site, replication must be triggered. Because of the finite storage capacity, replication decision should be adapted by users' needs. Therefore, the high requested files, named popular replicas, are efficiently kept and the replicas that are rarely needed are deleted. Moreover, the popularity is not stable over time. In general, the recent created files have the highest requests and as the time pass the popularity of file decreases. For instance, a new YouTube clip has most of the audiences. But, after the period of time it starts to lose popularity and the visitors [47].

Chang et al. [32] proposed a new dynamic replication named Last Access Largest Weight (LALW) strategy for Grid environment. The main goal of LALW method is to assign various weights to files having various ages. Madi and Hassan [48] improved LALW idea by considering an exponential growth/decay approach for an access number of files in access history. The main contribution of their strategy is that exponential decay base is adapted according to the file access rate. In other words, the rate of decay/growth for each file is not identical. Contrary to LALW strategy that considers exponential decay base equals to 1/2 for all files. Therefore, the rate of declension in weight will be slower. Madi and Hassan [48] and Khanli et al. [30] proved that the growth/decay of accesses is a more realistic and effective than the number of access in file popularity determination. Therefore, the authors introduced the exponential grow/decay rate of access to find the most popular files. To the best of our knowledge, the file accessing process in the distributed environment such as Grid and Cloud follows an exponential model [49].

We use the exponential decay to estimate the next number of access for data file. Note that we can model different real phenomena like bacteria and radioactive isotopes using relation that shows how things grow or decay as time passes. The exponential growth or decay approach is a growth in which the rate of growth is defined as proportional to the current size. The number of access for each files increases by the increase of access rate and vice versa. Therefore growth or decay change can be considered in the access history. For an access number of files in the access history, an exponential growth/decay pattern is defined.

If n_f^t shows the number of accesses for file f at time t and n_f^{t+1} indicates the number of accesses at time $t+1$ then the exponential growth/decay approach would be determined by:

$$n_f^{t+1} = n_f^t \times (1 + r), \tag{2}$$

where r indicates the rate of growth/decay and is defined as follows.

$$r = \frac{n_f^{t+1}}{n_f^t} - 1. \tag{3}$$

If t shows the intervals that are passed and n_f^t shows the number of accesses for file f at time interval t, then the sequence of the access numbers is obtained as follows:

$$n_f^0 \; n_f^1 \; n_f^2 \; \ldots \; n_f^{t-1} \; n_f^t. \tag{4}$$

So, we can write the following relations based on the growth/decay model.

$$r_0 = (\frac{n_f^1}{n_f^0}) - 1, \; r_1 = (\frac{n_f^2}{n_f^1}) - 1, \; \ldots, r_{t-1} = (\frac{n_f^t}{n_f^{t-1}}) - 1. \tag{5}$$

Finally, the average rate for all intervals is determined by Eq. (6):

$$r = \frac{\sum_{i=0}^{t-1} r_i}{t-1}. \tag{6}$$

For example, the numbers of access for file B during four intervals are 15, 20, 23, and 30, respectively.

Then, the average decay/growth rate for file B is determined.

$$r = \frac{0.33 + 0.15 + 0.3}{3} = 0.26.$$

Finally we estimate the next number of access for file B:

$$N_B^5 = 30 \times (1 + 0.26) = 31.2 \approx 31.$$

Finally, HDRS determines the most popular file based on this computation. File with the high number of access in the next time interval must be replicated.

4.2 Where the new replica should be placed?

The new labeling technique devotes to each site a label, which is an interpolation of its parent label and its local order [50]. As shown in Fig. 5, a label is a sequence of elements segregated by dots ("."), where the final element of the sequence indicates the local order of the site. The parent label is the sequence of elements excluding the final element. We assumed that number 1 shows the coordinator's label. In Fig. 5, the length of region headers labels is two and is presented by (1.1), (1.2), and (1.3). All of the sites that are placed at third level are determined as master sites. For example (1.1.1) indicates a master site label.

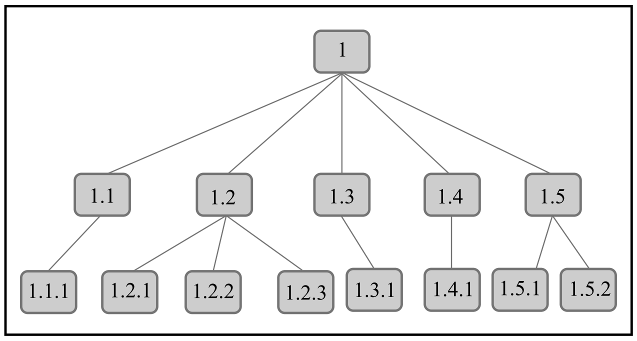

Fig. 5 Labeling structure

This labeling provides some advantages such as we can determine the precise physical the location of site, location of its parent, siblings, and ancestor. Moreover, it can find the number of hops between two sites and distance from the root site. It maps these labels to the physical file name.

Finding the best site for replica placement according to the network level locality is the first goal of our strategy [28]. In contrast to the strategy in Dogan [51], which only generates and stores replicas at the root of tree, in our schema replicas can be created and distributed to each site in the Level 3. The primary benefit of this schema is that we can avoid the bottleneck difficulty at the first level. In addition, replica servers have reasonable load in the most of time. Each requester that is answered by its parent imposes a particular access or storage load on that site. The whole load of server should not overstep its capability. The sum access loads of all children sites shows the access load of parent site. The access load of a site is the sum access loads of all its children sites. For example, if site x has five children then access load of site x will be the sum loads of five children.

After the popular file (P) is determined based on Eq. (6), we find a site where the file P has been accessed frequently based on the fact that it may require in the future. This site is named Requester. Now we must find the sites which contain the needed file (i.e., P). These sites are named as Provider. The path between the Provider sites and the Requester site is determined.

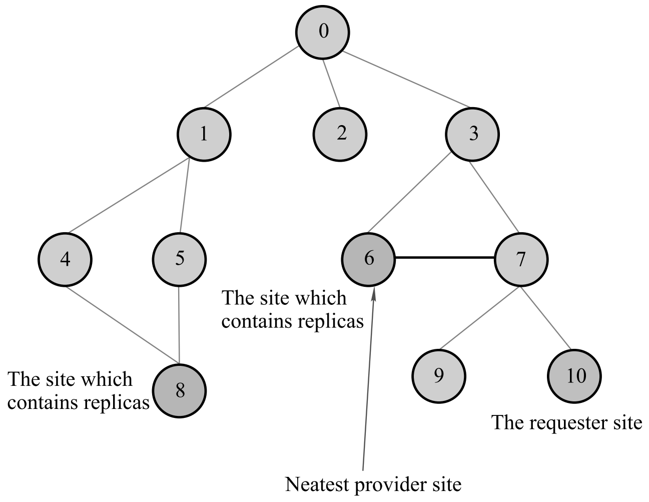
0
1
2
3
4
5
6
7
The site which contains replicas
The site which contains replicas
8
9
10
The requester site
Neatest provider site

Fig. 6 The provider and requester sites

Then, the nearest Provider site is selected. Figure 6 shows the Requester and Provider sites.

We have three cases based on the Requester location:

- The Requester site has the file P.
- The Requester and Provider are placed in the same region.
- The Requester and Provider are placed in the different regions.

4.2.1 Whenever the file P is in the Requester site, the replication is not triggered.

4.2.2 As it is shown in Fig. 7, if file P is not placed in the Requester site, this demand must be sent to the region header.

The master site determines which sites have the needed replica through its local replica manager and catalog. When the master site gets a file demand, it saves Physical File Name (PFN) of the requester site and Logical File Names (LFN) of that file request. Local replica catalog saves each request as (FileId, SiteId).

```
Algorithm 1   Replica placement when the requester (R) and provider (P) are placed in the same region
  if (file $P$ is in the site $R$)
     Replication is not triggered;
  else
{
     Determine Provider sites that are contained file $P$;
     Select the earnest Provider site;
     if (enough space available in site $R$)
       Store file $P$;
     else if ( $Value$ ($P$) > accumulative $Value$ loss by deleting the candidate files from site $R$) //see replacement
strategy
       {
         Delete candidate files from site $R$;
         Store file $P$;
       }
          else {
               Find the median site of the path;
               if (access load of the median site < its immediate sibling)
                  Store file $P$ in the median;
               else
                  Store file $P$ in its sibling;
               }
}
```

Fig. 7 Replica placement when the requester and provider are placed in the same region

4.2.2.1 Then, it sorts list L in the ascending order of *Value*. It candidates files from list L until enough space for file P is provided. The *Value* of file is obtained by:

$$Value = \frac{Num}{S}, \tag{7}$$

where *Num* is the number of accesses for the file in future time based on the exponential growth/decay. S shows the size of particular replica. Clearly, it is more critical to replace replicas with large size since it can decrease the number of replica replacement transactions. In addition, the mappings between physical location of sites and their labeling as (SiteId, Label) are stored in the database [52]. The Catalog Service returns the accurate physical location of sites that has the requested copy to the replicator. With the labeling service and the labeled tree, the labels of target sites are interacted to the replicator. Therefore, the replicator can find the physical file names and the label of sites that have the requested replica. Note that the region header's replica manager knows the locations of all sites in its region. It determines the path between the Requester and Provider sites based on their labels. Amongst sites that contain the file *P*, the earnest site is chosen. We have two cases:

1) If enough space is available in the Requester site then it stores file *P*.

2) If free space is insufficient, then the replacement process will be triggered. The file *P* is stored just in situation that the *Value* of storing file *P* be more than the expense of deleting the existed replicas. Firstly, it creates list *L* from files in the Requester site and for each file computes *Value* based on Eq. When HDRS decides to store the new replica *P* in a particular site that does not have enough storage, it must delete one or more files. The replacement will be done while, the *Value* for storing new replica *P* is higher than the *Value* of removing the candidate files. First, HDRS calculates *Value* (based on Eq. (7)) for each file in the Requester site. Then, it calculates the summation of all these *Value*. If the gained *Value* by replicating file *P*is greater than the accumulative *Value* loss in deletion of the candidate file from the Requester site, then HDRS deletes the candidate files and places the replica *P* in the Requester site.

• Else the median of this path is established. The median is the middle point of path between source and destination sites, in which half the numbers are above the median and half are below. If the access load of the median site is higher than its immediate sibling, then the strategy stores file *P*in its sibling. Figure 8 shows that the sibling site is selected.

4.2.3 As it is shown in Fig. 9, the coordinator queries from all of the headers and finds the label of sites which have the requested file. Amongst sites that contains the file *P*, the earnest site is chosen. We have two cases:

1) If enough space is available in the Requester site then it stores file *P*.

2) If space is insufficient then a group of files (that contains one or more files) must be deleted. Since in this step some valuable files are not available in the local region and may be needed in future. Therefore, such file unavailability will result in a high cost of transfer. It creates list *L* from files in the Requester site and for each file computes *Value* based on Eq. (7).

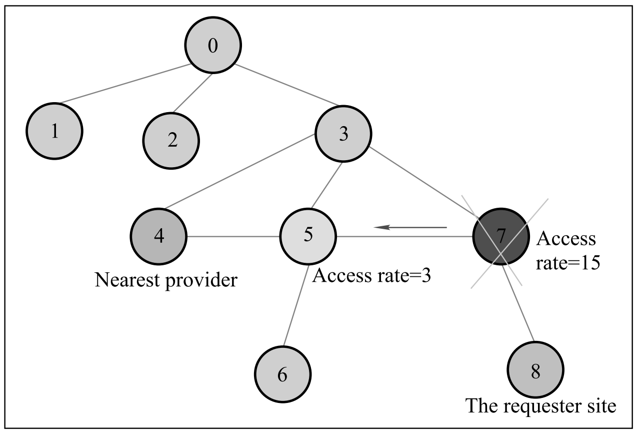

Fig. 8 The sibling site is selected

Algorithm 2 Replica placement when the requester (R) and provider (P) are placed in the different region

```
  if (file $P$ is in the site $R$)
      Replication is not triggered;
  else
{
    Determine Provider sites that are contained file $P$;
    Select the earnest Provider site;
    if (enough space available in site $R$)
      Store $P$;
    else
     {
     Create list L from the files in site $R$.
     Sort Lusing $Value$ in asecnding order.
     While (L is not empty && not enough space for file $P$)
        {
           Delete files file from list L in site $R$;
        }
    Store file $P$;
    }
}
```

Fig. 9 Replica placement when the requester and provider are placed in the different region

```
Algorithm 3   Replica replacement
// Isit possible to replicate file P in site S?
Create list L of files in the selected site (S);
For each file in list L compute Value (Equation 7);
Sort list L in ascending order of Value;
Sum = 0;
while (L is not empty && not enough space for file P) // Let P indicates the popular file
  {
          Select file $f_i$ from list L;
          Add $f_i$ in Candidate list;
        Sum = Sum + Value ($f_i$);
  }
if (Value (P) > Sum)
 {
   Delete files of Candidate list form site S;
   Store replica of P in site S;
 }
else
 Exit;
```

Fig. 10 Replica replacement

Then, it sorts list L in ascending order of *Value*. It deletes files from list L until enough space for file P is available. The main reason of this decision is that if the required file is located in the different regions, long time will be consumed for fetching the file. The replacement process is described in Fig. 10.

Before empirical comparing HDRS with the current replication strategies, we determine the time complexity of HDRS algorithm as follow.

Given a set of data nodes $D = \{D_1, \ldots, D_m\}$, a collection of replicas $R = \{fr_1, \ldots, fr_n\}$.

- The time complexity of Replica Creation would be $O(n)$. Since it finds the best file with the maximum number of accesses.
- The time complexity of Replica Selection would be $O(m)$. Since it finds the earnest site (based on their labels).
- The time complexity of Replica Placement would be $O(m)$. Since it finds the site with the highest number of accesses.
- The time complexity of Replica Replacement would be $O(nlog(n))$. Because it needs a sorting algorithm.

5 Performance evaluation

A cloud environment manages different issues such as network flow, load balancing, fault tolerance, bandwidth cost, interoperability, portability, reliability management and so on. Investigation in Cloud system usually emphasizes on these problems with varying importance. Experimenting new approaches in a real environment is not practically possible as such evaluations will provide the different systems and user QoS requirements such as reliability, cost, and speed. A suitable framework for experimental purposes is essential. One of the popular software is CloudSim toolkit. We have extended the classes of CloudSim toolkit to evaluate our strategy. Thus, it provides the comparison of various replication strategies.

The experiment of this section is divided into two parts. In the first part, the proposed method is compared with 8 other stated of the art methods (e.g., PGSA, MORM, EIMORM, CDRM, SWORD, QRS, BCDR, and PDR) in terms of average response time, effective network usage, replication frequency, load variance, and storage usage.

In the second part, the impact of four access patterns namely uniform, Zipf, geometric, and Growth/decay on the performance of proposed method in terms of number of replication and average response time is provided. In the both experiment parts, the simulation is run with 15 data centers and 100–1100 tasks which the computation workload of the task is from 2000 MI (Million Instruction) to 5000 MI. Table 1 shows the values of parameters in CloudSim. These values are set based on the existing studies [53] to realistically represent a typical Cloud environment. At the start of the simulator, the number of replicas of each data file is one and is randomly placed in different sites. Each task is send to the virtual machines according to the

Table 1 Values of parameters in CloudSim

Type	Parameters	Value
General	Number of regions	3
	Number of datacenters per region	5
	Number of VMs per datacenter	10
	Intra-region bandwidth	5 Gbit/s
	Inter-region bandwidth	1 Gbit/s
	Avg. Intra-region delay	25 ms
	Avg. Inter-region delay	100 ms
	Response time SLO	180 s
	Type of Manager	Space_shared
	Maximum simulation period	240 s
Virtual Machine	VM processing capability	1000–2000 MIPS
	VM storage capacity	10–20 GB
	Number of processing element per VM	2–5
	Type of Manager	Space_shared
Task	Total number of task	100–1000
	Length of task	2000–5000 MI
	Number of processing elements requirement	1–5
	Number of files for each task	5–25
	File size	500–1000 (MB)

Poisson distribution after the previous task and needs 5 to 25 data files [54,55]. In this study, we only assume that all data files are read-only and thus no data consistency technique is required.

5.1 Different data replication algorithms

The experiments set up five evaluation parameters: average response time, effective network usage, replication frequency, load variance, and storage usage. We consider 8 replication algorithms for evaluation as follows:

- Plant Growth Simulation Algorithm (PGSA) considers the status of network, storage load, and historical information in replica selection step.
- Multi-objective Optimized Replication Management (MORM) strategy stores new replicas based on the average file unavailability, average service time, load, latency, and energy consumption.
- Efficient and Improved Multi-objective Optimized Replication Management (EIMORM) algorithm considers file unavailability, average service time, load, latency, energy consumption, and cost during replica placement. Cost-Effective Dynamic Replication Management (CDRM) strategy places new replicas based on the site capability and blocking probability. It adapts the number of replicas and the location of replicas based on the load and capacity of site to provide a good load balancing system.
- Scalable Workload-Aware Data Placement (SWORD) strategy reduces mean query span and uses the partitioning technique. It models the expected workload based on the graph theoretic concepts to optimize resource usage.
- QoS based Replication Strategy (QRS) finds the similarity of recent requesting system and the replica history, and then selects the particular replica.
- Bee Colony based Data Replication (BCDR) algorithm finds the best site for storing replica based on the number of requests.
- Prefetching-aware Data Replication (PDR) algorithm determines the highly correlated files and replicates the most popular group of files.

5.1.1 Average response time

If response time is defined as the time duration among the sending of job and receiving of the answer, then the average response time is calculated by Eq. (8):

$$Average\ Response\ Time = \frac{\sum_{j=1}^{m}\sum_{k=1}^{m_j}(J_{jk}(rt) - J_{jk}(st))}{\sum_{j=1}^{m} m_j}, \tag{8}$$

where $J_{jk}(st)$ and $J_{jk}(rt)$ show the sending and receiving times of job k in user j, respectively. m_j is the number of jobs for user j and m indicates the number of users.

Figure 11 indicates the average response time of the seven dynamic replication algorithms. Accordingly, the average response time of MORM method is better than QRS. This is because MORM uses various criteria, i.e., the average file unavailability, average service time, load variance, energy consumption and mean access latency to find the relationship among replica number, replica layout, and their performances. The value of SWORD is lower than PGSA. Because, it minimizes the query span through data placement strategy. While, the lower flexibility of PGSA in load of replica access as well as the inability of dynamic adjustment leads to a higher average response time. The CDRM has lower response time about 6% in comparison with PGSA strategy for 1000 number of tasks. Since, it distributes replicas in a balance way by considering node capacity (CPU power, memory capacity, and network bandwidth). If the number of tasks is 1100 then the response time of PDR algorithm is about 25 second and the response time of BCDR algorithm is about 32 second. These results mean that the response efficiency of PDR algorithm is higher than other schemes especially when the number of tasks is high. Since PDR replicates the most popular group of files. HDRS replicates the desired file on the site which is placed nearby the user.

One of the key parameters that can minimize the Cloud site's job execution time is having their necessary files locally on their storage. Additionally, it determines the most popular file by analyzing the data access history. Therefore, HDRS has the lowest value of response time in comparison with PGSA, CDRM, SWORD, EIMORM, PDR, BCDR, QRS and MORM.

5.1.2 Effective network usage

Effective network usage shows the efficiency of network resource usage and is calculated by Eq. (9) [56]:

$$E_{enu} = \frac{N_{rfa} + N_{fa}}{N_{lfa}}, \tag{9}$$

where N_{rfa} indicates the number of access times that site reads a file from a remote site, N_{fa} shows the total number of file replication operations, and N_{lfa} indicates the number of times that site reads a file locally. The effective network changes between 0 and 1. Clearly, by decreasing the ENU, the usage efficiency of bandwidth was increased. In CloudSim, we define three variables as *totalRemoteReads*, *totalLocalReads*, and *replicas* with zero value. Then, we define a new method as *getStatistics()* that assigns appropriate values to the previous variables in cycle through all datacenters. Finally, we calculate Eq. (9).

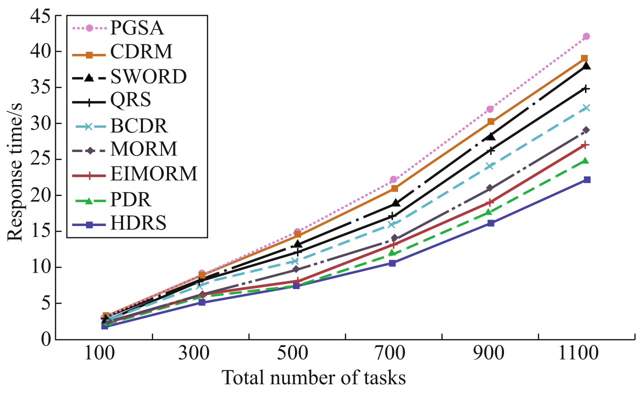

Fig. 11 Comparison of mean response time based on different number of tasks

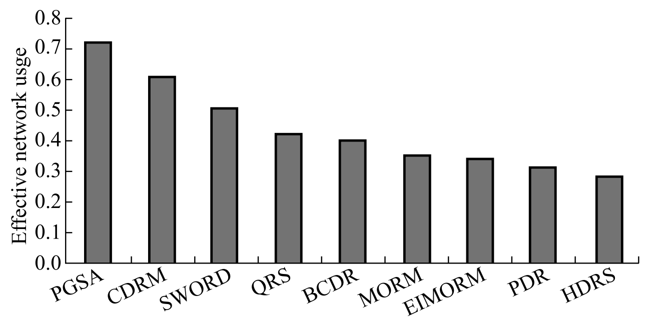

Fig. 12 Comparison of effective network usage for different data replication strategies

Despite of time and bandwidth consumption in data replication techniques, the application of this technique even in the simplest approach would be increased the network performance. Therefore, selection of efficient replication strategy can be decreased the future traffic. A lower value of ENU indicates the better performance on sorting data files in the proper locations. The effective network usage for different replication algorithms is shown in Fig. 12. The ENU of SWORD is better than PGSA because SWORD uses an incremental repartitioning approach that adapts replica placement in small phases without resorting to complete repartitioning.

In ENU, EIMORM strategy achieves better results (about 40%) compared with CDRM since EIMORM can balance the different optimization objectives (i.e., system availability, service time, load, and cost) by meta-heuristic technique. The ENU of HDRS is lower than about 30% compared with QRS strategy. This can be related to the ability of HDRS strategy to access locally the required data. Hence, the total replication number is diminished and enhanced the total number of local accesses. Accordingly, HDRS strategy could decrease the network traffic by minimizing the bandwidth consumption.

5.1.3 Replication frequency

By decreasing of the replication frequency, i.e., the ratio of how many replication is triggered per data access, the ability of replication algorithm to store data file in the appropriate sites is improved. The results of the replication frequency are presented in Fig. 13.

The replication creation is higher than 0.9 for PGSA algorithm, which shows that 0.9 replicas are created for a data access. The replication frequency of PGSA method is too high which renders it not feasible in the real environment. Figure 13 shows that MORM strategy has reasonable replica frequency, it is due to fact that it creates replicas on the basis of access load. The replication frequency of HDRS is less than 0.08, i.e., for successive 100 data access; eight replicas must be created. HDRS saves the valuable replicas while changing the environment. These characteristics can increase the availability beside of reducing unnecessary replications. Since HDRS transfers a small number of files therefore it does not effect on the network communication cost, considerably.

5.1.4 Load variance

Load variance is defined as the standard deviation of data node load of all data nodes in the Cloud which can be applied to indicate the degree of load balancing in the system [57]. The low value of load variance shows a good load balancing state.

Figure 14 shows that the load variance of HDRS replication strategy (with the lowest changes of load variance) is dramatically increased by increasing of file numbers. For example, HDRS yields 36% smaller load variance than PGSA and 28% than SWORD. The reason may be that HDRS balances the replica servers by using graph-based structure. It considers the sibling sites of replica servers on the structure and the access load on them. The load variance of MORM is about 8% better than BCDR, and better about 12% than QRS. This is due to MORM algorithm considers load during replication process.

If the load of system is light then PGSA strategy performs well for load balancing. The reason of that is the resource's capacities are high enough to avoid having a long job queue. When the load of system increases, the performance of PGSA deceases and becomes infeasible finally. This is due to PGSA deletes the large number of files in replacement process. Hence, the probability of fetching the files remotely will be increased; consequently triggering replication process will be increased as well.

5.1.5 Storage element usage

The capacity of replica servers effects on the performance of data replication strategy. Storage Element Usage (SEU) which is the percentage of available spaces that are used. Accordingly by increasing of SEU, the more storage resources can be resolved and studying of its percentage during simulation could be valuable. This can help in investigating a replication algorithm from two opposite points of view: on the one hand, the objective could be the minimization of storage usage, perhaps

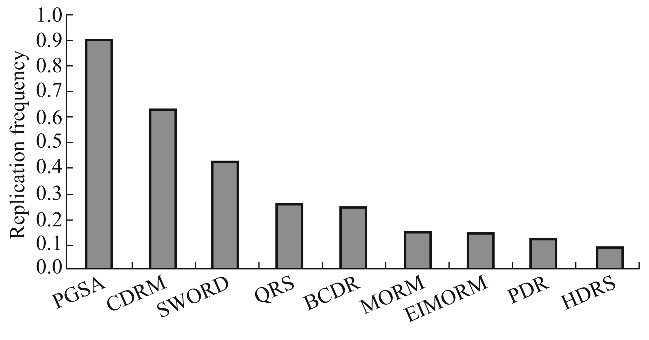

Fig. 13 Comparison of replication frequency for different data replication strategies

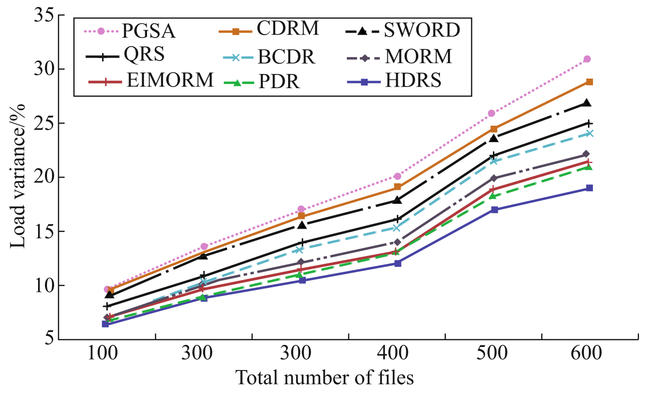

Fig. 14 Load variance for different data replication strategies

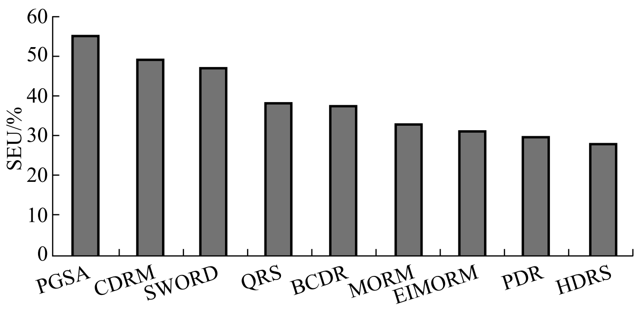

Fig. 15 Comparison of storage resources usage for different data replication strategies

because the resource expense is proportional to the amount being consumed; on the other hand, its expense might be fixed and one would then goal at maximizing the use of storage capacity.

Figure 15 compares the SEU of recent algorithms. Accordingly, HDRS works as a powerful strategy and places the replicas in the best site. Also, HDRS improves the temporal locality concept and finds the victim file based on the exponential growth/decay model. It invokes the replacement process only one time if there is a need to make a free space for the newly created replica. MORM uses the reasonable amount of network resources for the tested number of jobs because it is able to make a good decision in replica placement. However, if the available storage space is small, the benefit of MORM over HDRS decreases. The reason may be that MORM performs frequent replica replacement when the servers have small storage space. Consequently, the increased total number of replicas provides little advantage in terms of latency. The BCDR and QRS strategies use moderate storage space. Referring to the average storage usage, by using QRS, the storage usage is reduced by outperforming PGSA, 29%. Since, QRS considers user's QoS preference and the historical evaluation information in replication process.

5.2 Different file access patterns

In this section, the result of proposed replication algorithm when equipped with three different file access patterns (uniform, geometric, and Zipf) is discussed. Along with the growth/decay method, the following distributions are also used to predict the number of file access in the system.

(1) Geometric distribution: The probability of requests for the nth most popular file is obtained based on Eq. (10) [58].

$$p(n) = p(1-p)^{n-1}, \tag{10}$$

where n= 1, 2, … and $0 < p < 1$. The geometric distribution indicates that some files are accessed more times than others. When p has a larger value then a smaller fraction of data files is more requested.

(2) Zipf-like distribution: The number of requests for the nth most popular file is obtained by $n^{-\alpha}$ where α is a constant. Zipf distribution exists widely in the Internet world (e.g., Internet proxy, Internet traffic, Web server, and Gnutella) [59,60].

(3) Uniform distribution: In Zipf distribution if $\alpha = 0$ then the uniform distribution is appeared. The probability mass function of Zipf (α, n) distribution is defined by Eq. (11) [61]:

$$f(x) = \frac{1}{\sum_{i=1}^{n}\left(\frac{1}{i}\right)^{\alpha}} \quad x = 1, 2, \ldots, n. \tag{11}$$

If α=0 then we obtain the uniform approach:

$$f(x) = \frac{1}{\sum_{i=1}^{n}\left(\frac{1}{i}\right)^{0}} = \frac{1}{\sum_{i=1}^{n} 1} = \frac{1}{n}. \tag{12}$$

We investigate the results of proposed method with different access patterns in two perspectives (i.e., shapes of access patterns and the impact of access patterns on the number of replicas and response time).

5.2.1 Shape of file access patterns

In this perspective, the shapes of access patterns based on different files are investigated. Figure 16 shows the performance of proposed method when Zipf distribution is used for predicting the number of accesses to files in the next iteration. It can be seen from Fig. 16 that file with the highest popularity (i.e., file 1) also has the highest number of accesses. It is obvious that the shape of access pattern for the 10 most popular files with Zipf (α = 0.8) is exponential. Since the Zipf distribution is a member of the family of general exponential distributions and is used to model the ranks of randomly chosen files.

Figure 17 shows the performance of proposed method when the growth/decay approach is used for predicting the number of file accesses in the next interval. The shape of access pattern for the ten most popular files with growth approach is exponential. Therefore, the growth/decay approach almost has the same behavior compared with Zipf distribution. Adamic and Huberman [62] proved that the most accurate representation of the file requests on the modelling Internet traffic and the popularity of web pages is Zipf distribution [63]. Consequently, Zipf distribution and growth/decay can show reasonable results due to the conceptual similarity between web workloads and cloud file requests.

Figure 18 shows the performance of proposed method when uniform distribution is used. In the uniform distribution, all files have the same probability to be requested. So, the uniform approach unlike Zipf and growth/decay doesn't have a plan for

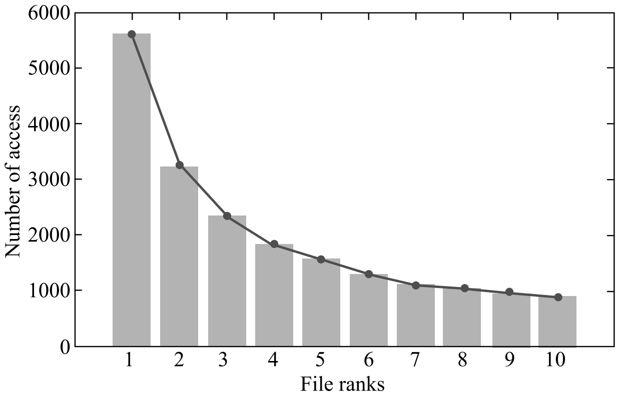

Fig. 16 Access pattern with Zipf

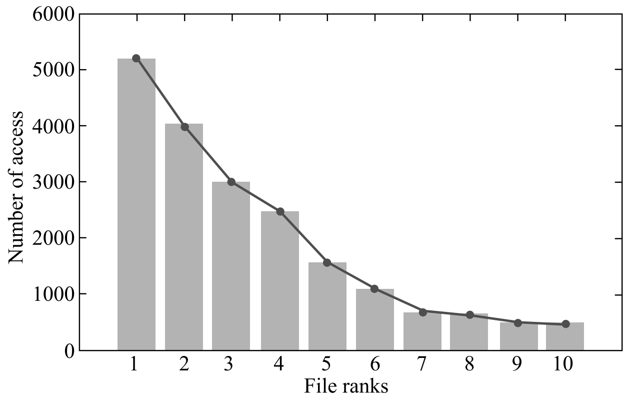

Fig. 17 Access pattern with growth/decay

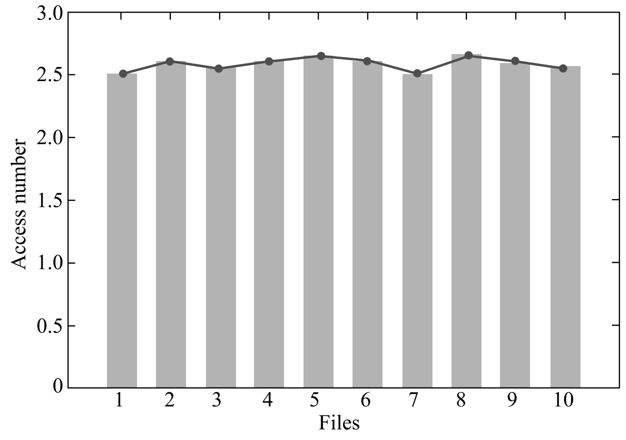

Fig. 18 Access pattern with uniform

determining the popular files. Therefore, all files have evenly chances to be requested in the next interval. The shape of access pattern for 10 files with uniform approach is like rectangle.

Figure 19 shows the performance of proposed method when geometric distribution is applied. In the geometric distribution, the probability of requesting popular files (p) is varying from [0.1-0.9] based on ranking files. It means that the popular files (files in early ranks) have more chances to be requested in early trials. It can observe that the requests for files are slowly decreased by decreasing the rank of files.

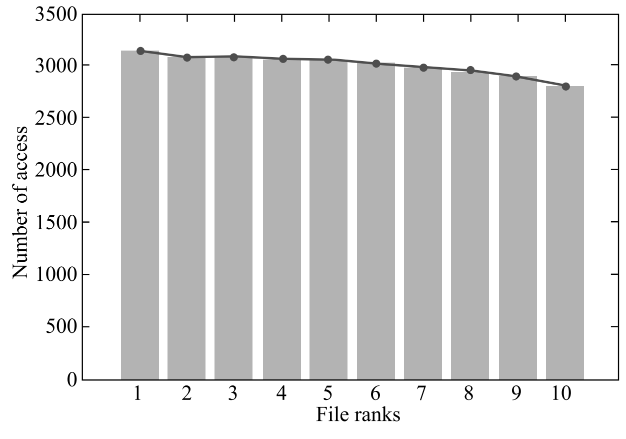

Fig. 19 Access pattern with geometric

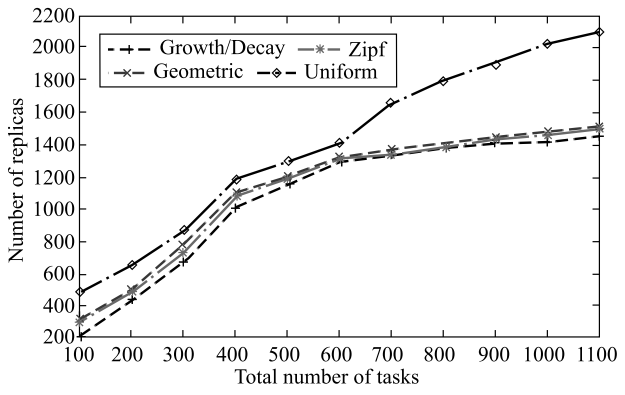

Fig. 20 Number of replicas for different access patterns

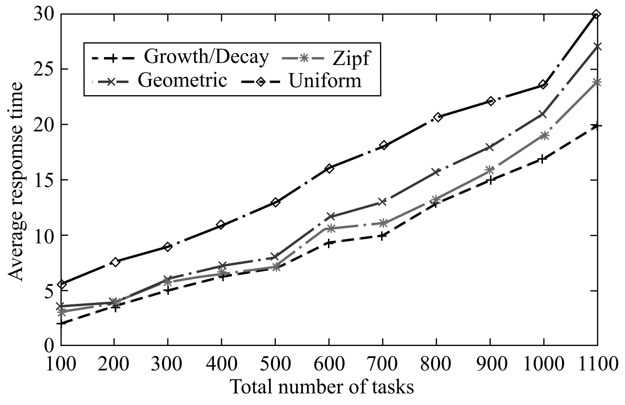

Fig. 21 Average response time for different access patterns

5.2.2 Impact of file access patterns on performance

Figures 20 and 21 present the performance of proposed method with different access patterns in terms of the number of replicas and response time, respectively. It can be observed that the proposed method with growth/decay approach performs much better than other distributions like uniform and reduces replicas by 11% and response time by 24% in average. Also, the proposed method with Zipf distribution outperforms geometric distribution in response time and the number of replicas (in average reduces the number of replicas and response time by 2% and 7%, respectively). Since the proposed method with Zipf can predict the appropriate file as popular one. Then, it replicates this popular file in the appropriate virtual machine and reduces response time.

Figures 22 and 23 show the result of proposed method with uniform distribution and growth/decay approach based on the different number of tasks and the number of files in terms of the number of replicas and response time, respectively. From both Figs. 22 and 23, it can interpret that growth/decay obtains better result compared with uniform (reduces the number of replicas by 22% and also reduces response time by 26% in average). Since the proposed method with growth/decay gives the high priority for the popular files which actually make up a small portion of the entire files.

Figures 24 and 25 show the performance of proposed method with Zipf distribution and growth/decay approach based on the different number of tasks and the number of files in terms of the number of replicas and response time, respectively. We can see that the proposed method with the growth/decay

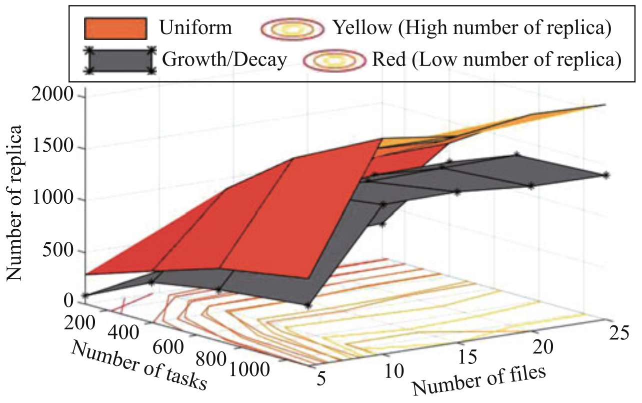

Fig. 22 Growth/decay vs. uniform for number of replicas

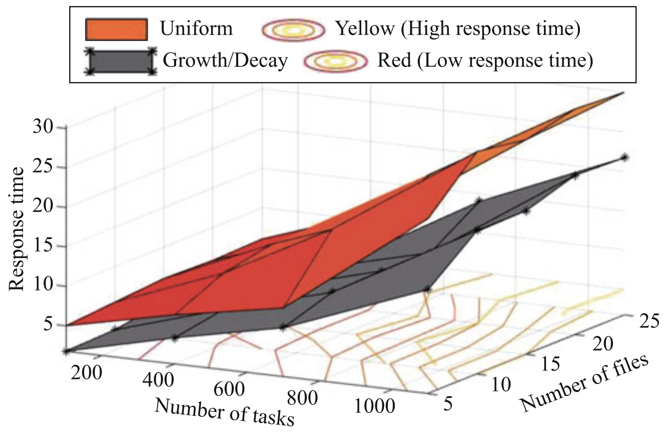

Fig. 23 Growth/decay vs. uniform for response time

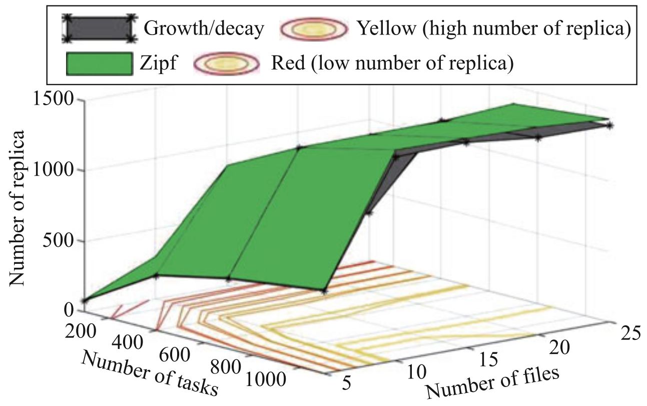

Fig. 24 Growth vs. Zipf for number of replicas

approach has almost similar behavior compared with Zipf distribution. This is because these two approaches can correctly predict the number of file accesses. The small superiority of growth approach (2% reduces replicas and 3% reduces response time compared with Zipf) is related to the weakness of Zipf in short interval. In short intervals, Zipf approach cannot precisely estimate the popular files. Since Zipf is a requested base approach. It means that the number of requests effect on Zipf prediction. For example, Zipf can make better decision when the number of requests is high (i.e., large intervals).

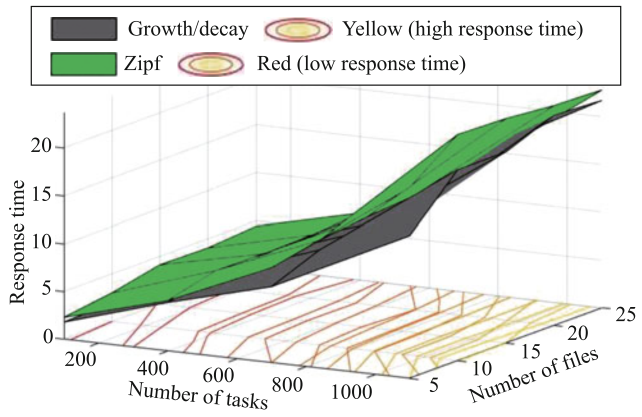

Fig. 25 Growth/decay vs. Zipf for response time

Figure 26 and Fig. 27 present the performance of proposed method with geometric distribution and growth/decay approach for different number of tasks and number of files in terms of the number of replicas and response time, respectively. It can observe that the proposed method with growth/decay has better performance compared with geometric approach. This is because in geometric approach the process of selecting popular file is highly depended on the probability of selecting (p). If the large value of p for file with rank 1 (i.e., the most accessed file in the previous interval) is selected then the probability of selection other files with high accesses as popular ones in next interval is decreased. Therefore, a few numbers of files is

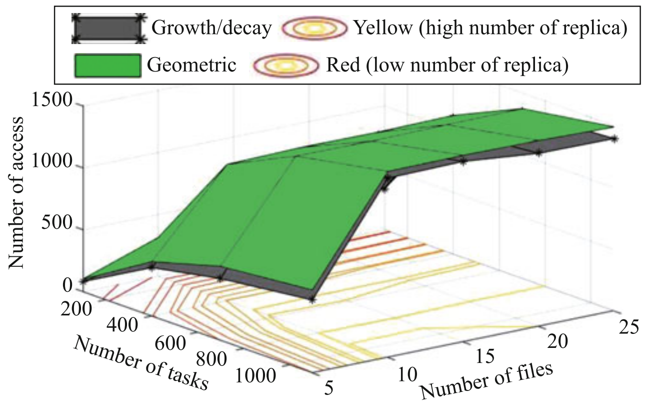

Fig. 26 Growth vs. geometric for number of replicas

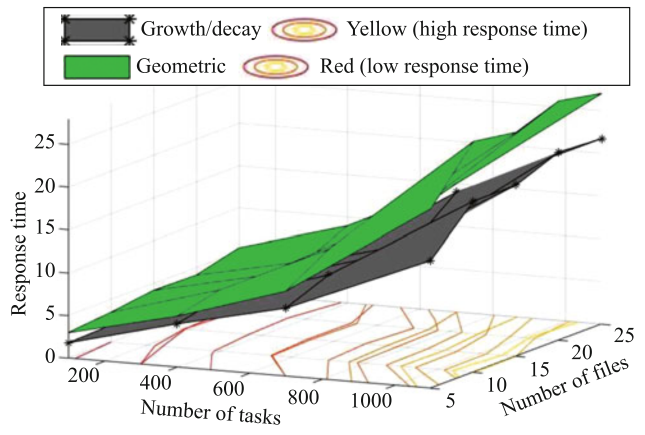

Fig. 27 Growth/decay vs. geometric for response time

considered as popular files for replication. On the other hand, if the small value of p for the file with rank 1 is selected then many files are considered as popular files and so the response time is increased.

6 Conclusion

Nowadays, the management of the large distributed resources efficiently around the wide area networks becomes a hot topic for both scientific study and commercial communication. In Cloud environment, data replication is one of the effective techniques affecting performance of the system. So, it is suggested that a powerful replication algorithm will improve data access and decrease job execution cost. We propose a Hierarchical Data Replication Strategy (HDRS) that offers high data availability and low bandwidth consumption. First, we find the popular file using the concept of exponential decay/growth. Then, the new replica is stored in the suitable locations sites so that each site's workload is balanced and a user could get the desired file quickly. Finally, HDRS replaces the replicas based on their importance. From the experiment results, it can be concluded that HDRS can achieve a good improvement of performance in terms of average response time, effective network usage, replication frequency, load variance, and storage usage over former similar works. As ongoing and future directions, we enrich the set of QoS parameters taken into account for data replication process consisting of service provider and client-related requirements with the business driven limitations. Also, we want to provide a reliable approach to determine how many copies are adequate for a file such that the file can be available against site failures.

References

1. Fu X, Chen J, Deng S, Wang J, Zhang L. Layered virtual machine migration algorithm for network resource balancing in cloud computing. Frontiers of Computer Science, 2018, 12(1): 75–85
2. Mansouri N, Javidi M M. A hybrid data replication strategy with fuzzy-based deletion for heterogeneous cloud data centers. The Journal of Supercomputing, 2018, 74(10): 5349–5372
3. Mansouri N, Javidi M M. A review of data replication based on meta-heuristics approach in cloud computing and data grid. Soft Computing, 2020
4. Yang X, Wallom D, Waddington S, Wang J, Shaon A, Matthews B, Wilson M, Guo Y, Guo L, Blower J D, Vasilakos A V, Liu K, Kershaw P. Cloud computing in e-Science: research challenges and opportunities. The Journal of Supercomputing, 2014, 70: 1453–1471
5. Shi Y, Meng X, Zhao J, Hu X, Liu B, Wang H. Benchmarking cloud-based data management systems. In: Proceedings of the 2nd International CIKM Workshop on Cloud Data Management. 2010
6. Thusoo A, Sarma J, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R. Hive-a warehousing solution over a MapReduce framework. Proceedings of the VLDB Endowment, 2009, 2(2): 1626–1629
7. Kuhlenkamp J, Klems M, Röss O. Benchmarking scalability and elasticity of distributed database systems. Proceedings of the VLDB Endowment, 2014, 7(12): 1219–1230
8. Loukopoulos T, Ahmad I, Papadias D. An overview of data replication on the internet. In: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN.02). 2002, 27–32
9. Mansouri N. Adaptive data replication strategy in cloud computing for performance improvement. Frontiers of Computer Science, 2016, 10(5): 925–935
10. ElYamany H F, Mohamed M F, Grolinger K, Capretz M A. A generalized service replication process in distributed environments. In: Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER). 2015, 20–22
11. Kim H, Parashar M, Foran D J, Yang L. Investigating the use of cloud-bursts for high-throughput medical image registration. In: Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (GRID). 2009
12. Mohamed M F. Service replication taxonomy in distributed environments. Service Oriented Computing and Applications, 2016, 10(3): 317–336
13. Zhong H, Zhang Z, Zhang X. A dynamic replica management strategy based on data grid. In: Proceedings of the 9th International Conference on Grid and Cloud Computing. 2010, 18–23
14. Ghemawat S, Gobioff H, Leung S T. The Google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles. 2003, 29–43
15. Wang Y, Wang J. An optimized replica distribution method in cloud storage system. Journal of Control Science and Engineering, 2017, 11: 1–8
16. Milani B A, Navimipour N J. A comprehensive review of the data replication techniques in the cloud environments: major trends and future directions. Journal of Network and Computer Applications, 2016, 64: 229–238
17. Tabet K, Mokadem R, Laouar M R, Eom S. Data replication in cloud systems: a survey. International Journal of Systems and Social Change, 2017, 8(3): 1–17
18. Shvachko K, Hairong K, Radia S, Chansler R. The Hadoop distributed file system. In: Proceedings of the 26th Symposium on Mass Storage Systems and Technologies, Incline Village, NV. 2010, 1–10
19. Mansouri N, Dastghaibyfard G H. Job scheduling and dynamic data replication in data grid environment. The Journal of Supercomputing, 2013, 64: 204–225
20. Tos U, Mokadem R, Hameurlain A, Ayav T, Bora S. Dynamic replication strategies in data grid systems: a survey. The Journal of Supercomputing, 2015, 71(11): 4116–4140
21. Jianjin J, Guangwen Y. An optimal replication strategy for data grid systems. Frontiers of Computer Science, 2007, 1(3): 338–348
22. Mansouri N, Javidi M M. A new prefetching-aware data replication to decrease access latency in cloud environment. Journal of Systems and Software, 2018, 144: 197–215
23. Gopinath S, Sherly E. A dynamic replica factor calculator for weighted dynamic replication management in cloud storage systems. Procedia Computer Science, 2018, 132: 1771–1780
24. Mansouri N, Dastghaibyfard G H, Mansouri E. Combination of data replication and scheduling algorithm for improving data availability in data grids. Journal of Network and Computer Applications, 2013, 36: 711–722
25. Dabas C, Aggarwal J. An intensive review of data replication algorithms for cloud systems. In: Shetty N, Pathaik L, Nagaraj H, Hamsavath P, Nalini N, eds. Emerging Research in Computing, Information, Communication and Applications. Springer, Singapore, 2019, 25–39
26. Mansouri N, Dastghaibyfard G H. Enhanced dynamic hierarchical replication and weighted scheduling strategy in data grid. Journal of Parallel and Distributed Computing, 2013, 73(4): 534–543
27. Ranganathan K, Foster I. Identifying dynamic replication strategies for a high performance data grid. In: Proceedings of International Workshop on Grid Computing. 2001, 75–86
28. Park S M, Kim J H, Ko Y B, Yoon W S. Dynamic data grid replication strategy based on Internet hierarchy. In: Proceedings of International Conference on Grid and Cooperative Computing. 2003, 838–846
29. Myint J, Hunger A. Comparative analysis of adaptive file replication algorithms for cloud data storage. In: Proceedings of International Conference on Future Internet of Things and Cloud. 2014
30. Khanli L M, Isazadeh A, Shishavan T N. PHFS: a dynamic replication method, to decrease access latency in the multi-tier data grid. Future Gen-

eration Computer Systems, 2011, 27(3): 233–244
31. Sun D W, Chang G R, Gao S, Jin L Z, Wang X W. Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. Journal of Computer Science and Technology, 2012, 27: 256–272
32. Chang R S, Chang H P. A dynamic data replication strategy using access-weights in data grids. Journal of Supercomputing, 2008, 45(3): 277–295
33. Kim Y H, Jung M J, Lee C H. Energy-aware real-time task scheduling exploiting temporal locality. IEICE Transactions on Information and Systems, 2010, 93(5): 1147–1153
34. Sun D W, Chang G R, Miao C, Jin L Z, Wang X W. Analyzing modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments. The Journal of Supercomputing, 2013, 66: 193–228
35. Zhang B, Wang X, Huang M. A PGSA based data replica selection scheme for accessing cloud storage system. Advanced Computer Architecture, 2014, 451: 140–151
36. Ding X, You J. Plant Growth Simulation Algorithm. Shanghai People's Publishing House, 2011, 1–59
37. Long S Q, Zhao Y L, Chen W. MORM: a multi-objective optimized replication management strategy for cloud storage cluster. Journal of Systems Architecture, 2014, 60(2): 234–244
38. Lou C, Zheng M, Liu X, Li X. Replica selection strategy based on individual QoS sensitivity constraints in cloud environment. Pervasive Computing and the Networked World, 2014, 8351: 393–399
39. Kumar K A, Quamar A, Deshpande A, Khuller S. SWORD: workload-aware data placement and replica selection for cloud data management systems. The VLDB Journal, 2014, 23(6): 845–870
40. Tos U, Mokadem R, Hameurlain A, Ayav T, Bora S. Ensuring performance and provider profit through data replication in cloud systems. Cluster Computing, 2018, 21(3): 1479–1492
41. Wu Z, Butkiewicz M, Perkins D, Katz-Basset E, Madhyastha H V. Spanstore: cost-effective geo-replicated storage spanning multiple cloud services. In: Proceedings of the 24th ACM Symposium on Operating Systems Principles. 2013, 292–308
42. Vulimiri A, Curino C, Godfrey B, Padhye J, Varghese G. Global analytics in the face of bandwidth and regulatory constraints. In: Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation. 2015, 323–336
43. Wei Q, Veeravalli B, Gong B, Zeng L, Feng D. CDRM: a cost-effective dynamic replication management scheme for cloud storage cluster. In: Proceedings of IEEE International Conference on Cluster Computing. 2010, 188–196
44. Edwin E B, Umamaheswari P, Thanka M R. An efficient and improved multi-objective optimized replication management with dynamic and cost aware strategies in cloud computing data center. Cluster Computing, 2019, 22: 11119–11128
45. Azimi S K. A Bee Colony (Beehive) based approach for data replication in cloud environments. In: Montaser Kouhsari S, eds. Fundamental Research in Electrical Engineering. Springer, Singapore, 2018, 1039–1052
46. Tatarinov I, Viglas S D, Beyer K S, Shanmugasundaram J, Shekita E J, Zhang C. Storing and querying ordered XML using a relational database system. In: Proceedings of the 2002 ACMSIGMOD International Conference on Management of Data. 2002, 204–215
47. Cheng X, Dale C, Liu J. Statistics and social network of YouTube videos. In: Proceedings of the 16th International Workshop on Quality of Service. 2008, 229–238
48. Madi M K, Hassan S. Dynamic replication algorithm in data grid: survey. In: Proceedings of International Conference on Network Applications, Protocols and Services. 2008
49. Madi M, Hassan S, Yusof Y. A dynamic replication strategy based on exponential growth/decay rate. In: Proceedings of International Conference on Computing and Informatics. 2009
50. Xu L, Ling T W, Wu H, Bao Z. DDE: from dewey to a fully dynamic XML labeling scheme. In: Proceedings of SIGMOD Conference. 2009, 719–730
51. Dogan A. A study on performance of dynamic file replication algorithms for real-time file access in data grids. Future Generation Computer Systems, 2009, 25(8): 829–839
52. Rahmani A M, Fadaie Z, Chronopoulos A T. Data placement using dewey encoding in a hierarchical data grid. Journal of Network and Computer Applications, 2015, 49: 88–98
53. Barroso L A, Clidaras J, Holzle U. The Datacenter As a Computer: an Introduction to the Design of Warehouse-scale Machines. 2nd ed. Morgan and Claypool Publishers, 2013
54. Murugesan R, Elango C, Kannan S. Cloud computing networks with poisson arrival process dynamic resource allocation. IOSR Journal of Computer Engineering, 2014, 16(5): 124–129
55. Mosleh M A S, Radhamani G, Hasan S H. Adaptive cost-based task scheduling in cloud environment. Scientific Programming, 2016
56. Cameron D G, Carvajal-schiaffino R, Paul Millar A, Nicholson C, Stockinger K, Zini F. UK Grid Simulation with OptorSim. UK e-Science All Hands Meeting, 2003
57. Lee L W, Scheuermann P, Vingralek R. File assignment in parallel I/O systems with minimal variance of service time. IEEE Transactions on Computers, 2000, 49(2): 127–140
58. Ranganathan K, Foster I. Decoupling computation and data scheduling in distributed data intensive applications. In: Proceedings of International Symposium for High Performance Distributed Computing. 2002
59. Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web caching and Zipf-like distributions: evidence and implications. In: Proceedings of IEEE INFOCOM'99, Conference on Computer Communications. 1999, 126–134
60. Iamnitchi A, Ripeanu M, Foster I. Locating data in (small-world?) peer-to-peer scientific collaborations. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems. 2002, 232–241
61. Visser M. Zipf's law, power laws and maximum entropy. New Journal of Physics, 2013, 15(4): 1–13
62. Adamic L, Huberman B. Zipf's law and the Internet. Glottometrics, 2002, 3(1): 143–150
63. Tos U, Mokadem R, Hameurlain A, Ayav T, Bora S. Dynamic replication strategies in data grid systems: a survey. The Journal of Supercomputing, 2015, 21(11): 4116–4140

Najme Mansouri is currently a faculty of Computer Science at Shahid Bahonar University of Kerman, Iran. She received her PhD in Computer Science from Shahid Bahonar University of Kerman, Iran and MSc in software engineering at Department of Computer Science & Engineering, College of Electrical & Computer Engineering, Shiraz University, Iran. She received her BSc (Honor Student) in Computer Science from Shahid Bahonar University of Kerman, Iran in 2009. She has published more than 40 scientific papers in the field of high-performance, parallel processing, grid and cloud computing, and contributed to more than 20 research and development programs.

Mohammad Masoud Javidi is currently an associate professor of Computer Science in Department of Computer Science, Shahid Bahonar University of Kerman, Iran. His research interests include distributed systems, and cloud computing. He has published more than 50 articles in journals and conferences.

Behnam Mohammad Hasani Zade received MSc degree in computer science from Shahid Bahonar University of Kerman, Kerman, Iran in 2018. His researches interests are in the areas of evolutionary algorithms, swarm intelligence, multi-objective optimization, machine learning and cloud computing.