Soft Computing (2020) 24:7335–7360
https://doi.org/10.1007/s00500-019-04357-w

METHODOLOGIES AND APPLICATION

Using data mining techniques to improve replica management in cloud environment

N. Mansouri[1,2] · M. M. Javidi[1,2] · B. Mohammad Hasani Zade[1,2]

Published online: 17 September 2019

Abstract
Effective data management is a crucial problem in distributed systems such as data grid and cloud. This can be achieved by replicating file in a wise manner, which reduces data access time, increases data availability, reliability and system load balancing. Determining a reasonable number and appropriate location of replicas is essential decision in cloud computing. In this paper, a new dynamic replication strategy called Data Mining-based Data Replication (DMDR) is proposed, which determines the correlation of the data files accessed using the file access history. We focus particularly on how extracted knowledge with maximal frequent correlated pattern mining improves data replication. We can group files with high dependency in the same replica set. Through the DMDR strategy, replicas can be stored in the suitable locations, with reduced access latency according to the centrality factor. In addition, due to the finite storage space of each node, replicas that are useful for future tasks can be wastefully deleted and replaced with less beneficial ones. Results of simulation using CloudSim indicate that DMDR strategy has a relative advantage in effective network usage, average response time, hit ratio in comparison with current methods. It can be concluded from this investigation that data mining technique is effective and helpful in the finding of users' future access behavior in cloud environment.

Keywords Cloud computing · Data replication · Data mining · Simulation

1 Introduction

A data-intensive system, encountered in some grid and cloud computing applications, consists of jobs that perform analysis or large-scale transfer data. Cloud computing system is getting more and more considerations as a new trend of data management (Peer Mohamed and Swarnammal 2017; Gupta et al. 2018; Mansouri et al. 2019). However, with the increasing requirement of users, size of data files, and request to obtain more knowledge from the huge amount of data, the utility of plain storage elements that are easy to control at a large scale increases (Mansouri and Javidi 2018). In cloud computing, data is one of the main elements and so a suitable, well-maintained, and efficient platform must be provided to guarantee dependability and availability of data. Some popular examples of cloud management systems are Cassandra (Cassandra 2011) and Hive (Thusoo et al. 2009) in Facebook, and HBase (HBase 2016). Well-defined data management techniques must have to present data reliability, availability, durability, and fault tolerance at different levels. Moreover, a platform where data replication is applied can present better aforementioned features and response time, which are serious from the perspective of the consumers (Rehman Malik et al. 2016).

1.1 Cloud Computing

Cloud computing provides various computing services as a utility service similar to power and water (Mansouri and Javidi 2018; Qi et al. 2017; Park et al. 2017; Mell and Grance 2009). Cloud computing is usually divided into the classes of: (a) Software as a Service, which offers applications as services in real time to different users, (b) Platform as a Service, which presents powerful environment

Communicated by V. Loia.

✉ N. Mansouri
najme.mansouri@gmail.com

1 Department of Computer Science, Shahid Bahonar University of Kerman, Box 76135-133, Kerman, Iran

2 Mahani Mathematical Research Center, Shahid Bahonar University of Kerman, Kerman, Iran

where developers can customize their applications, and (c) Infrastructure as a Service (IaaS), which presents virtual machine servers (Al-Asaly et al. 2019; Bojanova and Samba 2011). Each category has a specific purpose that provides various facilities to researchers and individuals. These categories with examples of each service provider are shown in Fig. 1. The possible variations of cloud computing paradigms are indicated in Fig. 2 (Abouzeid et al. 2009; Cooper et al. 2009).

1.2 Data replication in cloud computing

Replication in cloud computing can be introduced as storing multiple replicas of data across different data center. A cloud data replication service is shown in Fig. 3.

Replica management challenge is a hot research area in cloud computing environment (Mukundan et al. 2014). Replica management technique can reduce network delay and bandwidth usage of remote data access, improve load-balance of network, and enhance the safety of data by storing important replicas at suitable data centers (Torres-Franco et al. 2015). The main steps of replica management are replica creation, selection, placement, and replacement. According to various creations, selections and replacement algorithms, replica management methods can be classified into simple replication strategy, hierarchical model-based replication method and economic model-based replication schema (Zhong et al. 2010). Dynamic replication can take advantage of discovered relation through data mining approaches (Settouti et al. 2016; Jalil et al. 2016; Croda et al. 2019). Recently, there has been more attention on data mining approach in grid environment (Hamrouni et al. 2015). The first aim of this emerging field is to investigate and monitor history of data access in order to efficiently found new meaningful knowledge (Sánchez et al. 2008). The knowledge discovered can enhance overall performance in many fields such as data transfer (Grace and Manimegalai 2014), failure detection (Duan et al. 2006), scheduling (Bouyer et al. 2009), and resource management (Khanli et al. 2011).

1.3 Background of data mining

Han et al. (2011) and Zaki and Meira (2014) introduced data mining as the automated technique of extracting beneficial information and knowledge consisting of associations, changes, trends, patterns and structures from huge data sets. A typical knowledge discovery process may consist of the following steps: developing and understanding the domain of application, selecting and creating set of data, preprocessing and cleansing, selecting the data mining method, finding evaluation. Data mining as a core of knowledge discovery process is shown in Fig. 4. The main data mining approaches relied on distributed environment are association analysis, classification, and clustering (Kou et al. 2014, 2019). The author can study the issues of other data mining tasks such as outlier detection more deeply in (Moradi and Mokhatab Rafiei 2019; Kou et al. 2012; Peng et al. 2008, 2011; Ahmed et al. 2018).

As we can see that while grid and cloud focus on the distributed system architecture and protocols, data mining is combined basically with replication and scheduling

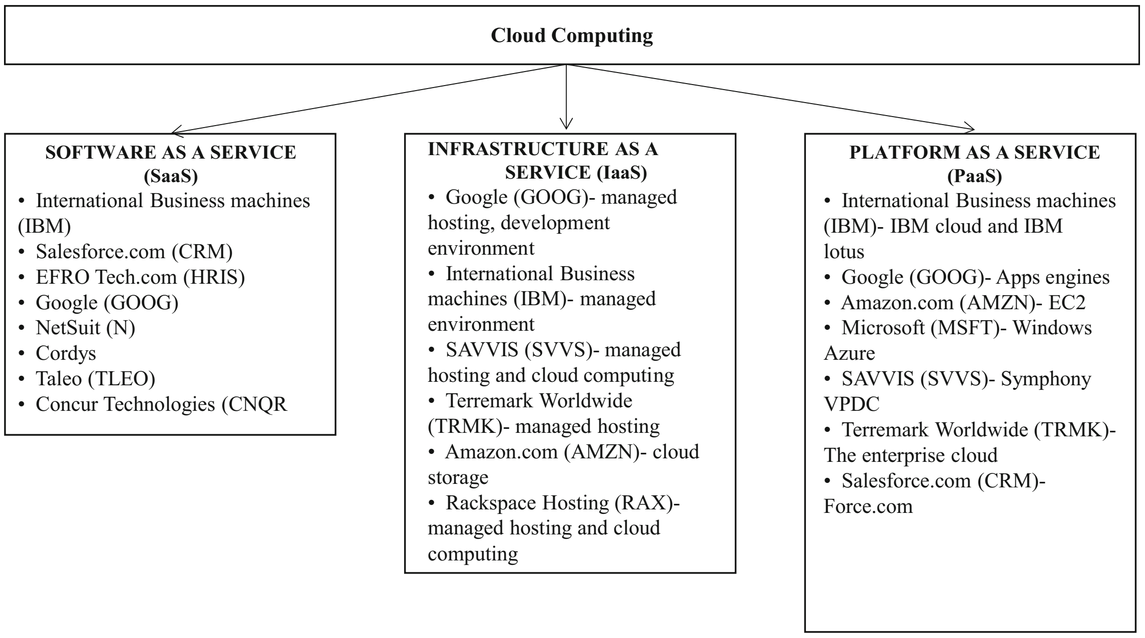

Cloud Computing
SOFTWARE AS A SERVICE (SaaS)
• International Business machines (IBM)
• Salesforce.com (CRM)
• EFRO Tech.com (HRIS)
• Google (GOOG)
• NetSuit (N)
• Cordys
• Taleo (TLEO)
• Concur Technologies (CNQR
INFRASTRUCTURE AS A SERVICE (IaaS)
• Google (GOOG)- managed hosting, development environment
• International Business machines (IBM)- managed environment
• SAVVIS (SVVS)- managed hosting and cloud computing
• Terremark Worldwide (TRMK)- managed hosting
• Amazon.com (AMZN)- cloud storage
• Rackspace Hosting (RAX)- managed hosting and cloud computing
PLATFORM AS A SERVICE (PaaS)
• International Business machines (IBM)- IBM cloud and IBM lotus
• Google (GOOG)- Apps engines
• Amazon.com (AMZN)- EC2
• Microsoft (MSFT)- Windows Azure
• SAVVIS (SVVS)- Symphony VPDC
• Terremark Worldwide (TRMK)- The enterprise cloud
• Salesforce.com (CRM)- Force.com

Fig. 1 Cloud computing service models and examples

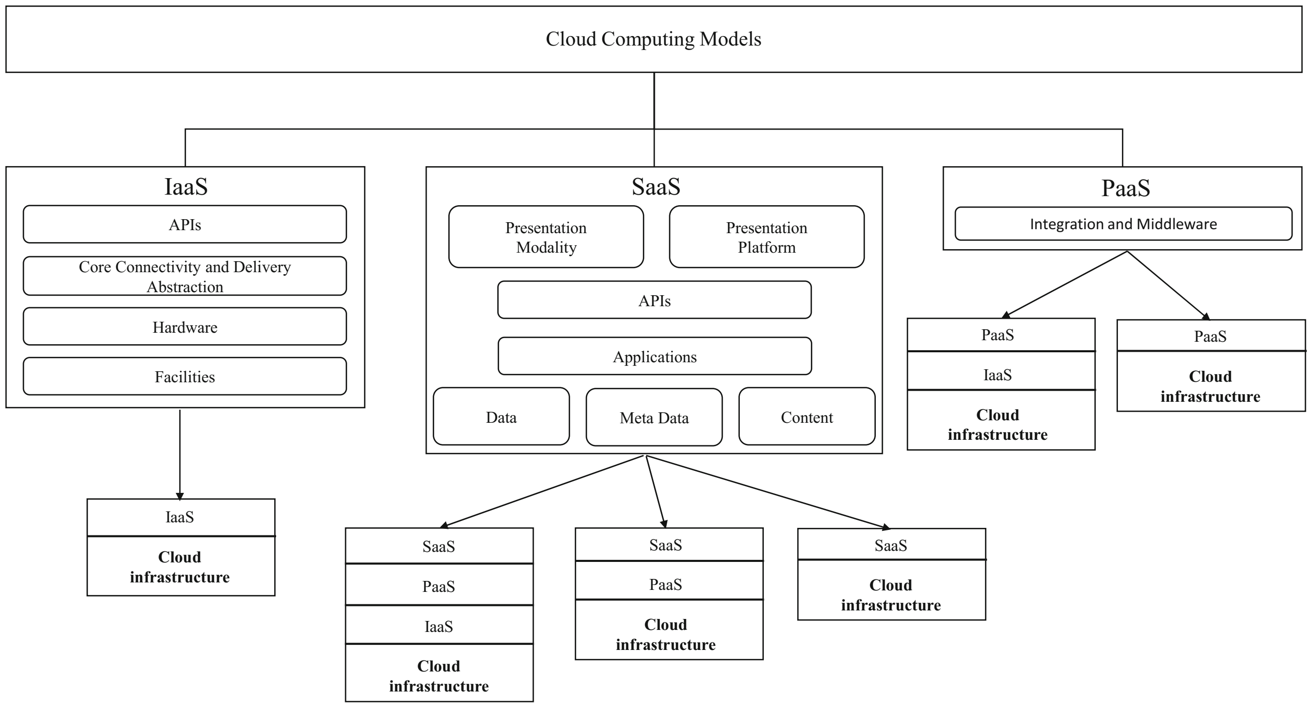


Fig. 2 Cloud computing service models variations

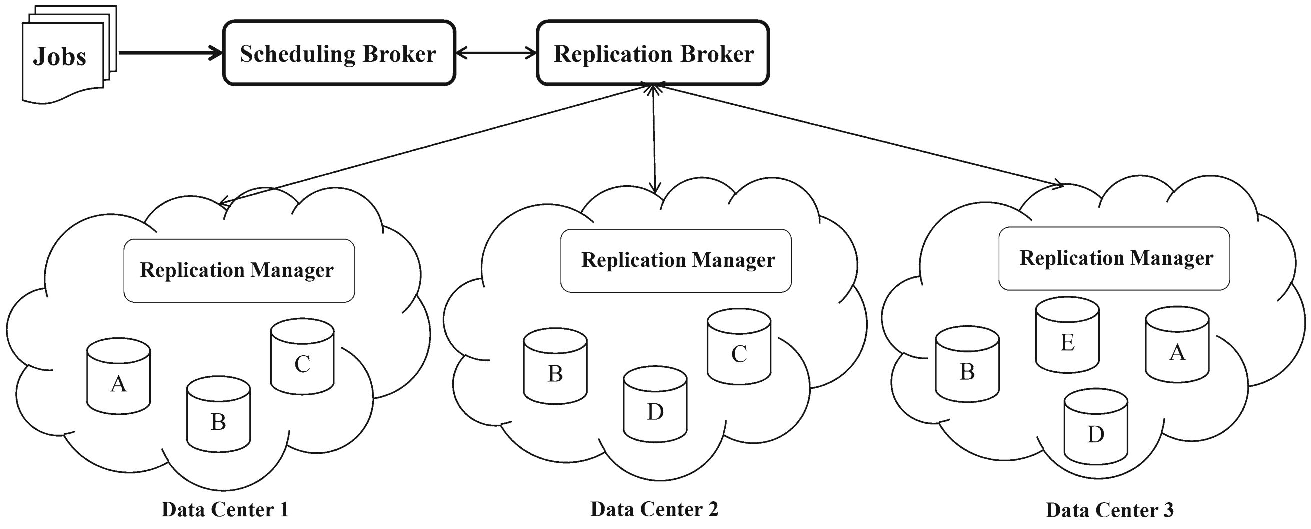


Fig. 3 The cloud data replication architecture

methods for discovering knowledge from data. Nowadays, the emergence of cloud approach, as well as the increasingly complex nature of data mining applications, has led to a new cooperation of cloud and data mining. On the one side, cloud environment provides powerful computational system for distributed data mining applications. On the other side, mining cloud data is emerging as a novel type of data mining application. The interesting task is to monitor access pattern from the cloud based on data mining techniques to extract new useful knowledge and enhance cloud performance.

Generally some data files tend to have correlations in same jobs, applications or users. In many real data-intensive applications such as high-energy physics, some files are requested together. We can group files with strong correlations in the same replica group to improve the data replication strategy. For example, if correlated files are stored in the same or nearby sites then the number of distributed transactions, which need to access to several sites, is reduced.

In this paper, we propose a Data Mining-based Data Replication (DMDR), which determines the correlation of

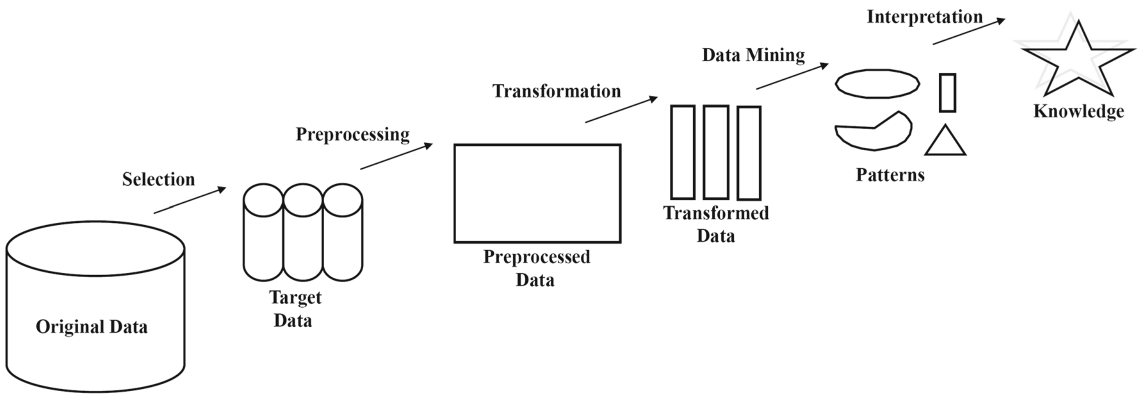


Fig. 4 Knowledge discovery process

the data files accessed using the history of access. An efficient replica placement strategy can result in good performance gains. In this regard, DMDR selects the best site that is most central and high number of accesses. Finally, DMDR uses another paradigm based on replica deletion instead of unnecessary replication to further improve the efficiency of storage resource utilization. DMDR removes data files that are the least likely to be needed in the near future. For evaluation purposes, the DMDR is simulated using the CloudSim simulator (Calheiros et al. 2011). The experimental results show that the DMDR is able to reduce access time, enhance hit ratio, total number of communications, and improve effective network usage in the cloud environment.

The rest of the paper is organized as follows. Section 2 shows a brief introduction of the related works on data replication. Section 3 introduces Data Mining-based Data Replication (DMDR) Algorithm. We explain DMDR process using a case study in Sect. 4. Section 5 evaluates the performance of effective and current replication strategies in cloud system. In Sect. 6, we illustrate some real examples to see the application of the proposed method. Finally, we conclude the paper and point out some future works in Sect. 7.

2 Related works

Data replication is one of the most challenging issues in distributed systems (Manjula et al. 2016; Nivetha and Vijayakumar 2016; Khalili 2019). This issue has attracted considerable international attention from the researchers in the last few years. In the following, we present a review of most relevant prior investigations.

Long et al. (2014) designed a Multi-objective Optimized Replication Management (MORM) strategy by balancing the trade-offs among the different parameters in cloud environment. They proposed formula that is combination of various optimization factors such as data availability, load, latency, service time, energy consumption, probability of failure. Users can set weight of variables based on the requirements. For example, assigning a higher value on their performance objectives makes replication strategy adaptable. The experimental results using the extended CloudSim showed that MORM strategy enables to enhance load balancing of system and reduce the energy consumption. This algorithm does not pay attention to the replica replacement that is the crucial step when the storage space is full. But our proposed algorithm (DMDR) tries to deletes files with the least likely to be required in the future.

Casas et al. (2017) presented a Balanced and file Reuse-Replication Scheduling (BaRRS) strategy that schedules applications in cloud environment. BaRRS splits large workflows into several small workflows to improve system utilization using paralleling technique. It also uses data replication algorithm to data access at run-time. BaRRS considers important characteristics of workflows, such as execution time, dependency patterns, and file size. They executed four scientific workflows based on various dependency patterns and file size. The obtained results confirmed that the workflow dependency patterns have a significant effect on performance of scheduling strategy. The main weakness of this approach is that it ignores dynamic variation of file access patterns. But, our proposed replication algorithm (DMDR) handles monitors historical data of data centers based on data mining techniques to find important information and improve response time.

Tos et al. (2018) introduced Performance and Profit Oriented Data Replication (PEPR) method that provides SLA guarantees such as availability and performance as well as improves the economic benefit of the cloud provider. When an estimated response time is higher than the SLO response time threshold, PEPR stores replicas in data centers. PEPR triggers replication only when the response time and profitability of the provider are satisfied. It stores replicas in the site that is closest to the most amounts of requests. The number of replicas is continually determined based on satisfying the SLA objectives. Simulation results confirmed that PEPR satisfied the SLA and profitability of the cloud provider. Neglecting the hidden association relationships in file access is the main disadvantage of the PEPR strategy, while our proposed replication algorithm (DMDR) focuses on pre-fetching replicas within the cloud computing that is not considered in Tos et al. (2018).

Mansouri et al. (2017) developed a Dynamic Popularity aware Replication Strategy (DPRS) that is based on the leveraging data access behavior. It dynamically calculates data popularity based on the variation of users' access behaviors and then stores most frequently requested data to the best locations. It finds the best locations according to the number of requests, free storage space, and site centrality. DPRS applies the parallel downloading approach to assemble data fragments. The weakness of strategy is neglecting the effect of file access pattern in replica decision. But our proposed replication algorithm (DMDR) replicates the related files based on historical behavior.

By consideration of transmission cost, evaluation information of history, system load and users QoS preference as criteria, Lou et al. (2014) proposed a strategy on the base of individual QoS sensitivity restrictions. This strategy answers to this question: Does appropriate file exist in local node? If yes, used directly. Else, feature of replica and historical evaluation information must be checked. QoS preference is able to estimate the availability level, timeline and reliability. By doing so, it is responsible to estimate the similarities between current demanding environments with the historical replica. Finally, the replica with the highest credibility similarity environment is selected. The analytical results indicated that the proposed strategy enables increase data availability due to that replication in an adaptive manner by adjusting the reliability parameters in the environment model. Nevertheless, the scheme focuses only on the replica selection. Our proposed replication algorithm (DMDR) considers the issue of replica placement and replica replacement simultaneously.

Elango and Kuppusamy (2016) proposed a data replication strategy based on the Fuzzy-FP-Tree-based frequent pattern mining algorithm. They applied FP-growth strategy along with the support and confidence is being determined based on the membership function in the fuzzy logic to extract the frequent patterns. In the first step, they converted quantitative values in transactions into linguistic terms according to the on Hong et al.'s (1999) idea. Experiment results demonstrated that it delivered better data access time without much delay while comparing the available Apriori_based method. Its shortcoming is that it didn't determine which file to be replaced whenever the storage is full. In a real large-scale cloud environment, storage is a critical element. Therefore, the proposed strategy (DMDR) improves the temporal locality property by considering information such as data access frequency, size of file, access time to determine the victim file.

Alghamdi et al. (2017) formulated problem of file replication inside data center, with the goal of reducing the energy consumption of data file access in cloud system. They proposed Profit-Based Replication (PBR) algorithm, which minimizes the energy consumption by at least half of that, obtained by an optimal solution. Profit concept is defined as combination of the file size and energy consumption. The intuition behind the profit is that storing a replica in a data center is more profitable if this decrease more energy usage for file access and the file has a smaller size. Simulations using CloudSim toolkit indicated that under Zipf distribution, the performance difference between Profit and Local is even larger. But the file access pattern is not addressed in this work. Our proposed strategy (DMDR) takes into account file correlations to pre-fetch the frequently access files in the suitable location based on centrality.

According to literatures, different replication strategies developed to copy the critical file at reasonable time. Table 1 summarized these strategies based on different options:

- Bandwidth Consumption determines whether the bandwidth consumption is considered in replication decision.
- Response Time shows whether the replication strategy takes into account the response time for replication.
- Load Balancing specifies whether the workload is balanced on different data centers.
- Energy Efficiency option determines whether replication considers the energy consumption in cloud data centers.
- Consistency management guarantees that the multiple replicas of a particular file are kept consistent in the presence of concurrent updates.
- Security option determines whether the replication algorithm takes into account the security factors in the replication process.
- Data Mining indicates whether the data mining techniques is used in replica decision or not.
- Simulator used factor specifies the simulator that is used to test the proposed strategy.

Results of the experimental evaluation show that while a data replication strategy is necessary to satisfy performance requirements, using traditional algorithms are not enough for the cloud provider to take a holistic view of the different steps and benefits of replication. Traditional strategies have done that, such as providing better response time,

Table 1 Data replication factors considered by existing replication algorithms in cloud

Mechanisms	Year	Main idea	Bandwidth consumption	Response time	Load balancing	Energy efficiency	Consistency	Security	Data mining	Simulator
Long et al. (2014)	2014	Modeling multi-objective optimized replication management	+	+	+	+	–	–	–	CloudSim
Casas et al. (2017)	2017	Analyzing workflow features	+	+	+	–	–	–	–	Real environment
Tos et al. (2018)	2016	Estimating the response time	+	+	+	–	–	–	–	CloudSim
Mansouri et al. (2017)	2017	Considering the access popularity	+	+	–	–	–	–	–	CloudSim
Lou et al. (2014)	2014	Proposing multi-objective offline optimization approach	+	+	–	–	–	–	–	CloudSim
Elango and Kuppusamy (2016)	2016	Using fuzzy-FP tree	–	+	–	–	–	–	+	JAVA platform
Alghamdi et al. (2017)	2017	Using energy-efficient heuristic	–	+	–	+	–	–	–	CloudSim

higher network usage, they did not pay attend to relevant knowledge like file access pattern. To the best of our knowledge, comparatively less works have been attempted for proposing replication algorithms based on the data mining techniques for cloud systems, instead most of the works exist for grid environments.

1. In many real data-intensive applications like high-energy physics (Doraimani 2007; Ko et al. 2007), some files tend to have correlations. As a consequence, it is necessary to discover files that are accessed together. Therefore, we apply a maximal frequent correlated pattern mining strategy to find sets of highly related files.
2. Data retrieval time is a key factor in determining the throughput and the quality of service of cloud. So, the proposed method considers centrality and number of replica access in placement step to reduce retrieval time. The central data center can disseminate files faster.
3. With explosive growth of data, storing several replicas in the limited storage is another problem and hence a powerful replica replacement strategy is essential to replace useless replicas with vital replicas. DMDR strategy removes replicas based on the last time the replica was accessed, access number and size of file.

3 Data Mining-based Data Replication (DMDR) algorithm

3.1 Main definitions

In this section, we introduce how each data replication concept is mapped to suit the data mining tasks.

Item: The proposed strategy assumes an item is a logical value (zero or one) that indicates whether or not the file is required enough by a particular job.

Transaction: The proposed strategy determines each transaction as a set of files required by a particular job.

Extraction context: Generally, data mining approach defines an extraction context based on a triplet $K = (T, I, R)$, where T shows the set of transactions/object, I indicates the set of attributes/items and R determines a binary relation (i.e., $R \subset T \times I$). In other words, each couple $(t, i) \in R$ shows that the object $t \in T$ consists of the item $i \in I$. In our case, items are the accessed files and transaction is the required files of each job.

Pattern: A pattern/itemset is determined as a set of items.

Frequent pattern: The proposed strategy defines a frequent pattern as a set of files that exist simultaneously in the history of file access with a higher support than predefined support value.

Correlation measure: There are different metrics for correlation such as *lift* and $\chi 2$ (Brin et al. 1997), *all-confidence*, *any-confidence* and *bond* (Omiecinski 2003), *coherence* (Lee et al. 2003), and *cosine* (Wu et al. 2010). The *all-confidence* correlation measure is selected for our replication case. The *all-confidence* value for an itemset *Y* is found based on Eq. (1):

$$All{-}Confidance\,(Y) = \frac{Support(\wedge Y)}{Max\{Support(\wedge i)|i \in Y\}} \quad (1)$$

Correlated pattern: If the value of correlation measure for a pattern is higher than or equal to the predefined threshold of correlation measure then pattern is found as a correlated pattern.

Frequent correlated pattern: In our case, a set of files whose correlation value overpasses the predefined threshold (i.e., *MinCorrelation*) and the number of simultaneous occurrences in the history of file access overpasses the predefined threshold (i.e., *MinSupport*) shows a frequent correlated pattern.

Maximal frequent correlated pattern: If a frequent correlated pattern has no frequent correlated pattern as a superset then it found as a maximal frequent correlated pattern. We have four main steps as follows:

Step 1: Each site saves the history of accesses performed by the jobs assigned on it for all remote files and local ones. In the first step, we extract the history of file access. Access pattern specifies the order of requested file by the job. Let us consider, a job J_1 needs three files F_1, F_2, and F_3. History of file access creates couple (J_1, F_1) which is shows the number of times F_1 is accessed by a J_1 in a particular period.

Step 2: Second step converts the history of file access to logical file access history, named as binary context. We creates binary context in a form of table which is contains the logical values for required files of each job.

Step 3: Third step found the hidden correlations among files according to the maximal frequent correlated pattern mining strategy. In other word, we group the closely related files.

Step 4: Fourth step uses the maximal frequent patterns determined in the third phase as input of the replication strategy. It found the best site for replica placement based on the centrality and number of access. If enough storage is available to store the set of correlated files in the best site, then the replication will trigger. Otherwise, replacement process should be take place. We assigned value to each file based on important parameters such as the last time the replica was accessed, access number and size of file. If aggregation value of the candidate files for deletion is lower than the value of the group of files to replicate, then candidate files are removed. Otherwise, the replication process will be abandoned.

3.2 Constructing a binary context based on the file access history

Each site creates history of file access periodically. Assume a matrix $A_{n \times m}$ indicates history of file access on site *Sitei*, where number of jobs are scheduled to the site *Sitei* in a particular period is shown by *n* and number of files requested by these jobs is determined by *m*. In other words, *Aj,k* = *NumAccess* (*Jj*, *Fk*) shows the number of times the job *Jj* accessed the file *Fk*. Now we transform the history of file access to a binary context consisting logical values (zero or one). For this purpose, we consider the popularity of file. If number of requests of *Fk* by jobs executed in *Sitei* is higher than average number of accesses *Fk*, then *Fk* is popular. The average number of accesses of each file, i.e., *AverageAcc* (*Fk*) is determined as Eq. (2):

$$AverageAcc\,(F_k) = \frac{\sum_{k=1}^{N_j} NumAccess(J_j, F_k)}{N_j} \quad (2)$$

where *Nj* is the number of jobs assigned to *Sitei* that request the file *Fk*.

Firstly, for each file *Fk*, the algorithm computes *AverageAcc* (*Fk*). Then for total number of jobs, if *NumAccess* (*Jj*, *Fk*) > *AverageAcc* (*Fk*), i.e., *Jj* frequently requests *Fk*, The value of (*Jj*, *Fk*) cell is set as 1. Otherwise, the value 0 will be stored

Algorithm 1. MFCP()

```
1  Input: Binary context, MinSupport, Min-All-Confidence
2  Output: Mfcp
3    k=1;
4    $Fcp_1 = \{i \in T \mid Support(i) > MinSupport)\}$;
5    Mfcp= Fcp1;
6    While ($Fcp_k \neq \phi$)
7     {
8        Fcp_{k+1} = Gen_Next_Fcp (Fcp_k, MinSupport, Min-All-Confidence);
9        For each ($X_{k+1} \in Fcp_{k+1}$) do
10         if ($\exists X_k \subset X_{k+1} \mid (X_k \in Mfcp)$) then
11               Delete X_k from Mfcp;
12         $Mfcp = Mfcp \cup Fcp_{k+1}$;
13         k= k+1;
14    }
15   Return Mfcp
```

Algorithm 2. Gen_Next_Fcp ()

```
1  Input: Fcp_k, MinSupport, Min-All-Confidence
2  Output: $Fcp_{k+1}$
3    $Fcp_{k+1} \neq \phi$;
4    $Con_{k+1}$= Apriori ($Fcp_k$);
5    For each ($X_{k+1}$ in $Con_{k+1}$) do
6      {
7        If $(\exists x, y \in X_{k+1} \mid \frac{Support(\wedge x)}{Support(\wedge y)} < MinAllConfidence)$
8          Stop;
9        If $(\exists X_k \subset X_{k+1} \mid (X_k \notin Fcp_k))$
10         Stop;
11       If $(Support(X_k) \geq MinSupport)$
12         {
13           $AllMax(X_{k+1}) = \max\{Support(i) \mid i \in X_{k+1}\}$;
14           $AllConfidence(X_{k+1}) = \frac{Support(X_{k+1})}{AllMax(X_{k+1})}$;
15           If $(AllConfidence(X_{k+1}) \geq Min-all-confidence)$
16               $Fcp_{k+1} = Fcp_{k+1} \cup X_{k+1}$;
17         }
18     }
19 Return $Fcp_{k+1}$;
```

.

3.3 Algorithm of maximal frequent correlated pattern mining

Algorithm of maximal frequent correlated pattern mining needs binary context, *MinSupport* threshold and *Min-All-Confidence* threshold as inputs. Definition of parameters is presented in Table 2. Algorithms 1 and 2 indicate the Maximal Frequent Correlated Pattern (MFCP) algorithm.

1. It compares support of each item with *MinSupport* threshold. Then, the initial set of maximal frequent correlated patterns is determined by set of frequent items. In addition, the all-confidence value of an item is set as 1.
2. We do the following steps in each iteration:

   - Frequent correlated patterns of size *k*-1 constructs the frequent correlated patterns of size *k*.
   - *Mfcp* contains the set of $Fcpk + 1$. Moreover, line 10 to line 11 in the strategy is used to guarantee each element of *Mfcp* has no subset in this same set.
   - The algorithm is terminated when the new pattern isn't generated.

Table 2 Parameter description

Parameter	Description
X_k	A pattern with size *k*
Con_k	Candidate patterns with size *k*
Fcp_k	Frequent correlated pattern with size *k*
Mfcp	Maximal frequent correlated patterns set

3.4 Replica placement

One of the serious concerns in cloud with high-speed growth of data is finding the best site for replica placement. We store replicas based on the centrality and number of replica access. According to the temporal locality (recently accessed file are likely to be accessed again), number of replica access has a main role in replica placement decision (Saleh et al. 2015).

Moreover, the relative importance of a site in the system can be determined by the centrality of a node in a graph. Our strategy considers the centrality to reduce retrieval time. There are different centrality metrics such as closeness centrality, degree centrality, between centrality, and eccentricity centrality (Newman 2009). We only consider the closeness metric in replica placement process. A site is set as closeness in a network, if it has the lowest value for the summation of the distances from all of the other sites. The lower the sum of distances from the other sites, the more centrality has the site. The closeness centrality value for site *v* can be defined as the following (Newman 2009):

$$Centrality(a) = \frac{N-1}{\sum_{a \neq b} d(a,b)} \tag{3}$$

The parameter *N* is used to indicate total number of data centers in the system and *d*(*a*,*b*) shows the distance between data center *a* and data center *b*.

In the sequel, replica is stored in a site with the highest value of *Merit* by Eq. (4).

$$Merit(s) = w_1 \times TotalAccess + w_2 \times Centrality \tag{4}$$

where *TotalAccess* indicates total number of access for files in *Mfcp* list by site *s*, *Centrality* is centrality value of site *s*, w_1 and w_2 are the proportion weights corresponding to the above two main parameters. We know that the scale of above parameters is different. Therefore, it is necessary to transform value of parameters into a scale of 0–1 before applying them in Eq. (4). We assume that the scale of normalization is uniform.

Maximal frequent correlated patterns that are found are candidates for replication. Now, we have three important phases:

- Maximal frequent correlated patterns are arranged based on the number of items in descending order.
- For each pattern, if the enough storage space is available in the selected site to copy all the files of the correlated pattern, then DMDR strategy stores them.

- Otherwise, one or more replicas must to be candidate for replacement. For this purpose, DMDR strategy considers key factors such as the last time the replica was accessed, access number and size of file. The number of access and the last time the replica was requested identify the probability of requesting the file again. In addition, it is logical to replace files with large size and can decrease the number of replica replacement. Firstly, it creates list L from files in selected site and for each file compute *Value* based on Eq. (5). Then it sorts list L in ascending order of *Value*. It selects candidate files from list L until enough space is available. *Value* of file is obtained by:

$$Value(f) = \frac{NAcc_f}{(CT - LT_f) + Size_f} \tag{5}$$

where CT is the current time, LT_f is the last request time of file f, $Size_f$ shows size of file f, and $NAcc_f$ is the number of access of file f

```
Algorithm 3. Replace ()
Input: Selected site (S)
Output: Specify which files should be deleted
  Create list L of files in the selected site ($S$);
  For each file in list L compute $Value$ based on Eq. (3);
  Sort list L in ascending order of $Value$,
  Sum =0;
  While (L is not empty && not enough space for files of Mfcp list )
  {
     Select file $f$ from list L;
     Add $f$ in Candidate List (CL),
     $AvgDel = \frac{1}{|CL|} \times \sum_{f \in CL} Value(f)$;
  }
 $AvgMfcp = \frac{1}{|Mfcp|} \times \sum_{f \in Mfcp} Value(f)$;
 If ($AvgMfcp > AvgDel$)
  {
     Delete files of Candidate List (CL) from site $S$;
     Store all files of Mfcp list in site $S$;
  }
 Else
     Exit;
```

.

Then, it computes two average value criteria: the first for the set of data files to replicate and is named *AvgMfcp*, while the second is related to the candidate files for deletion and is called *AvgDel*. Finally, if the *Value* gained by replicating files (*AvgMfcp*) is greater than the accumulative *Value* loss in deletion of the candidate file (*AvgDel*), then DMDR deletes the candidate files and places new correlated group of files. Replacement strategy is shown in Algorithm 3. Figure 5 shows flowchart of the DMDR algorithm.

4 Case study

In this example, there are four sites *Site0*, *Site1*, *Site2* and *Site3* as shown in Fig. 6. Let us consider nine master files are stored in *Site0* with size 30 GB and the storage size of other sites is 15 GB. We assume that the proposed algorithm select *Site1* for replication according to the *Merit* value. In addition, eight jobs run on the *Site1* in the studied period. Table 3 indicates the history of file access.

Phase 1: *Constructing a binary context based on the file access history*

DMDR strategy determines for each file *Fk* its average number of accesses as indicated in Table 4. For example, average number of accesses of F_1 is 55. For each job *Jj* and a given file *Fk*, if *NumAccess* (*Jj*, *Fk*) ≥ *AvgerageAcc* (*Fk*) then it set *NumAccess* (*Jj*, *Fk*) by value 1. Otherwise, the value of *NumAccess* (*Jj*, *Fk*) is set by 0. In this case, we replace the *NumAccess* (*Jj*, *Fk*) value by 1 in the binary context; otherwise, the value 0 is saved. Table 5 indicates the binary context.

Phase 2: *Determining the maximal frequent correlated pattern*

DMDR strategy uses algorithm of maximal frequent correlated pattern mining (Sect. 3.3) based on the value of *MinSupport* is 0.3 and *Min-All-Confidence is 0.4.*

For frequent correlated with size one, only F_3 is infrequent. Then it finds the frequent correlated patterns with size two according to the frequent correlated patterns with size one. For each pattern, it identifies whether it is frequent. If pattern is frequent, then it compares its *all-confidence* measure with *Min-All-Confidence*. Else, it prunes this pattern and it doesn't calculate its correlation value.

In this case, frequent correlated patterns with size 2 are F_1F_2, F_1F_5, F_1F_6, F_1F_4, F_2F_4, F_2F_5, F_2F_6, F_4F_5, F_4F_6, F_4F7, F_4F_8, F_4F_9, F_5F_6, F_6F7, F_6F_9, F_6F_8, $F7F_8$, $F7F_9$ *and* F_8F_9. Then, it determines the maximal frequent correlated patterns with size 3 and so on based on the same method. $F_1F_2F_4F_5F_6$ and $F_4F_6F7F_8F_9$ are maximal frequent correlated patterns.

Phase 3: *Replication process*

The file characteristics such as number of accesses, size of file, and last time access are presented in Table 6. DMDR strategy determines two patterns ($F_1F_2F_4F_5F_6$ and $F_4F_6F7F_8F_9$) for replication. $Site_1$ contains the F_1 and F_2 for first pattern. The free storage of $Site_1$ is 8 GB (15 GB − 7 GB = 8 GB). For the first pattern, available storage of $Site_1$ is enough for storing F_4, F_5 and F_6. After replication, $Site_1$ has F_1, F_2, F_3, F_4, F_5 and F_6.

In another example, we want to place $F_4F_6F7F_8F_9$ in $Site_1$ that contains F_1, F_2, F_3. The free storage of $Site_1$ is 8 GB. Therefore replacement process is triggered. For this case, DMDR strategy assigns *Value* to the files based on

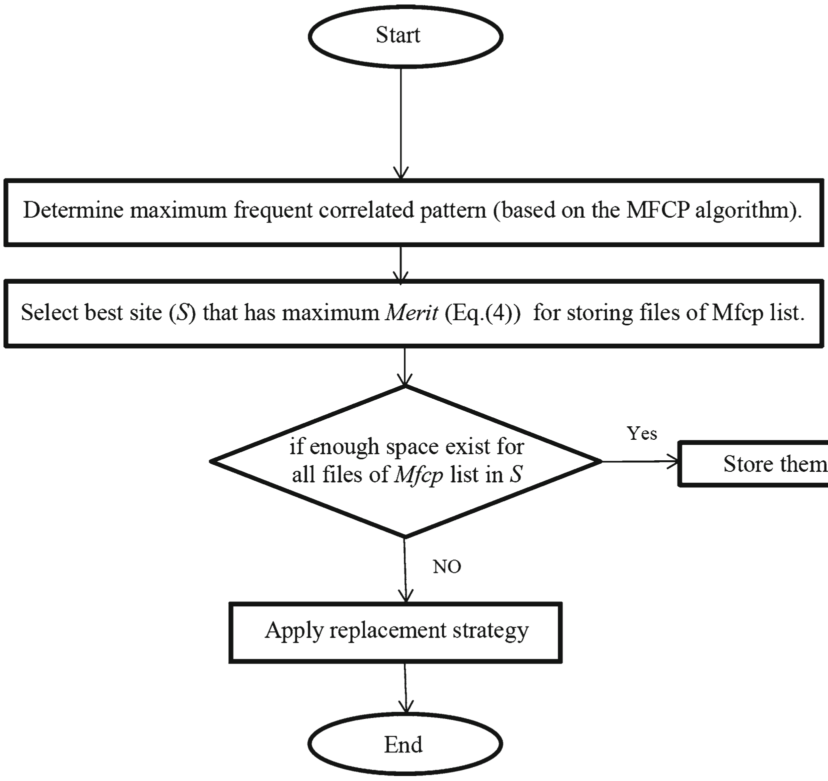

Start
Determine maximum frequent correlated pattern (based on the MFCP algorithm).
Select best site (S) that has maximum Merit (Eq.(4)) for storing files of Mfcp list.
if enough space exist for all files of Mfcp list in S
Yes
Store them in S
NO
Apply replacement strategy
End

Fig. 5 Flowchart of DMDR algorithm

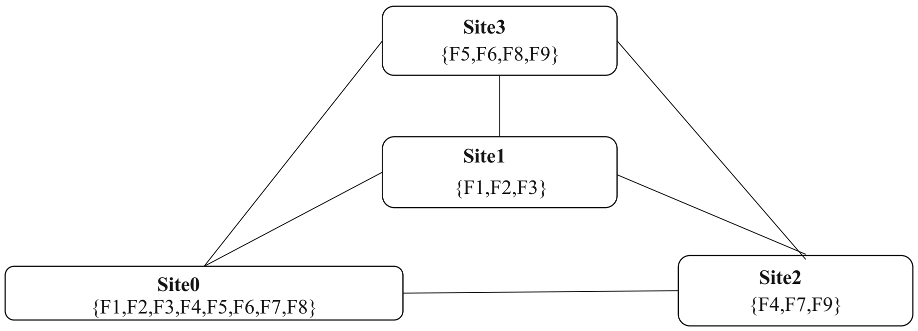


Fig. 6 Distribution of files before executing the DMDR algorithm

Table 3 File access history for *Site₁*

Job	File								
	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
J_1	60	160	50	195	200	100	0	20	0
J_2	0	20	0	195	50	100	100	100	130
J_3	60	150	0	195	260	100	0	80	130
J_4	60	150	0	195	200	100	0	0	0
J_5	0	0	0	195	0	100	100	100	130
J_6	0	0	0	195	0	100	100	100	130
J_7	60	160	0	195	190	100	0	0	0
J_8	0	0	0	195	0	100	100	100	130

Table 4 Average access of each file for *Site₁*

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
AvgerageAcc	30	80	6.25	195	101.2	100	50	83.3	81.2

Eq. (5) as shown in Table 7. Therefore, it selects F_1 and F_3 as the candidate files for replacement.

As a result, the average value of candidate files for deletion (i.e., F_1 and F_3) and the average value of candidate files for replication (i.e., $F_4F_6F7F_8F_9$) are computed as 0.04 and 1.02, respectively (i.e., $AvgDel = 0.04$ and $AvgMfcp = 1.02$). Since the average value of candidate files for deletion is lower than the average value of candidate files for replication, the files F_1 and F_3 will be

Table 5 Binary context for *Site₁*

Job	File								
	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
J_1	1	1	1	1	1	1	0	0	0
J_2	0	0	0	1	0	1	1	1	1
J_3	1	1	0	1	1	1	0	0	1
J_4	1	1	0	1	1	1	0	0	0
J_5	0	0	0	1	0	1	1	1	1
J_6	0	0	0	1	0	1	1	1	1
J_7	1	1	0	1	1	1	0	0	0
J_8	0	0	0	1	0	1	1	1	1

Table 6 File characteristics

F_i	NumAccess (F_i)	Size (F_i)	CT-LT (F_i)
F_1	240	2	5
F_2	640	3	3
F_3	50	3	5
F_4	1560	2	2
F_5	900	2	3
F_6	800	2	1
F_7	400	3	4
F_8	580	1	4
F_9	520	3	3

Table 7 File *Value*

F_i	NumAccess (F_i)	Size (F_i)	CT-LT (F_i)	Value (F_i)
F_1	0.12	0.5	1	0.08
F_2	0.39	0.1	0.4	0.78
F_3	0	0.5	1	0
F_4	1	0.25	0.2	2.2
F_5	0.56	0.25	0.4	0.86
F_6	0.49	0.25	0	1.96
F_7	0.23	1	0.6	0.14
F_8	0.35	0	0.6	0.58
F_9	0.31	1	0.4	0.22

removed from *Site₁*, while *F7*, F_8 and F_9 will be stored in *Site₁*. As a consequence, Fig. 7 indicates the new distribution of files in the cloud site for two previous examples.

5 Evaluation results and analysis

5.1 Simulation scenarios

We have performed extensive experiments with three main categories as follows.

- Replication algorithms comparison: Performance evaluation of the proposed replication algorithm by comparing it with other related methods based on different metrics.
- Effect of thresholds of the proposed algorithm on efficiency: Investigation of MinSupport and Min-All-Confidence thresholds in response time and effective network usage.
- Network topology architecture: Study of the parameters which effect on topology network.

5.1.1 Replication algorithms comparison

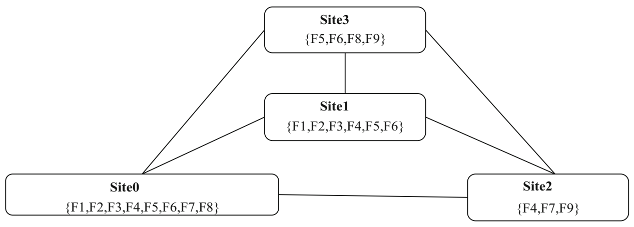
We have extended different classes of CloudSim toolkit to test replication algorithms. Table 8 shows the general parameters' setting of cloud simulator. The parameter of simulation is set based on the existing studies (Barroso et al. 2013) to represent a typical cloud environment realistically. We have performed an intensive set of simulation studies for showing the performance power of proposed method. To test the effectiveness of the proposed strategy, we consider Average Response Time, Storage Usage, Number of Communications, Replication Frequency, and Hit ratio as performance metrics for analytical evaluation. At the beginning of simulation, we placed the primary copy of each data file is randomly sites.

5.1.1.1 Average response time If the response time is defined as the time duration among the sending of job and receiving of the answer, then the average response time calculated from Eq. (6):

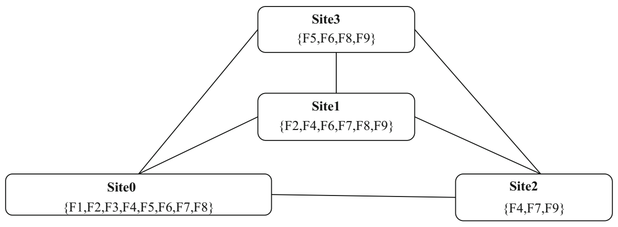
$$Average\,Response\,Time = \frac{\sum_{j=1}^{m}\sum_{k=1}^{m_j}\left(J_{jk}(rt) - J_{jk}(st)\right)}{\sum_{j=1}^{m} M_j} \tag{6}$$

where $J_{jk}(st)$ and $J_{jk}(rt)$ shows the sending and receiving times of job k in user j, respectively. m_j is the number of the jobs for user j. We have performed four different of tests based on different number of tasks, different number of data centers, different file size and different number of files for average response time.

Tables 9–12 show the simulation structure for test numbers 1–4, respectively. Figure 8 compares the average response time of the replication strategies for the uniform distribution and Zipf distribution. We can see that MORM



(a) Replicating $F_1F_2F_4F_5F_6$ in Site1.



(b) Replicating $F_4F_6F_7F_8F_9$ in Site1.

Fig. 7 Distribution of files after executing the DMDR algorithm

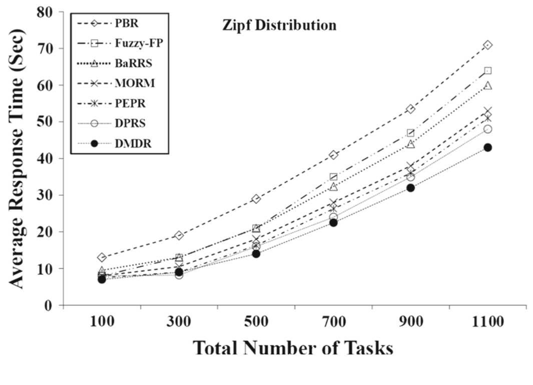
Table 8 Parameters setting of cloud simulator

Type	Parameters	Value
Data center	Number of host	3–10
	Type of manager	Space_shared
		Time_shared
Virtual Machine	MIPS of processing element	500–300
	Number of processing element per VM	2–5
	VM memory (RAM)	5–20 (GB)
Task	Length of task	100–500 (MI)
	Number of processing elements requirement	1–4

Test number 1

Table 9. Response time simulation variables: Total number of tasks.

Parameter	Value
Total Number of tasks	**[100-1100]**
Total_Number of data centers	60
Number of files	20
Size of files	1500 Mb



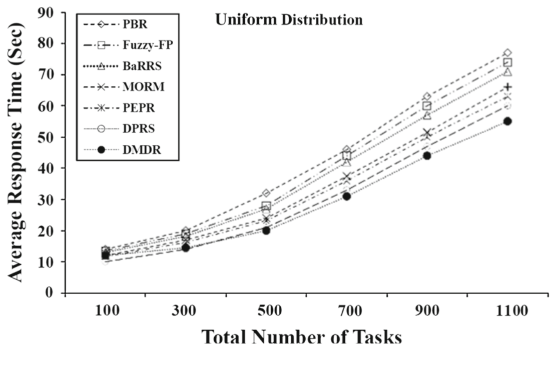


Fig. 8 Average response time based on varying number of tasks

strategy outperforms the BaRRS algorithm by up to 11% when number of task is 1000. Since MORM strategy replicates based on the file unavailability, service time, load variance, energy usage and latency to determine the relationship among replica number, replica layout and these performances. But BaRRS achieves better average response time compared with Fuzzy-FP due to creating a much greater number of copies in the cloud nodes. DPRS executes tasks on a data center with the most required files and has slightly better average response time in comparison with PEPR strategy. DMDR applies pattern mining to find subsequent file requests when a file is requested in data center, and this hence reduces average response time and latency.

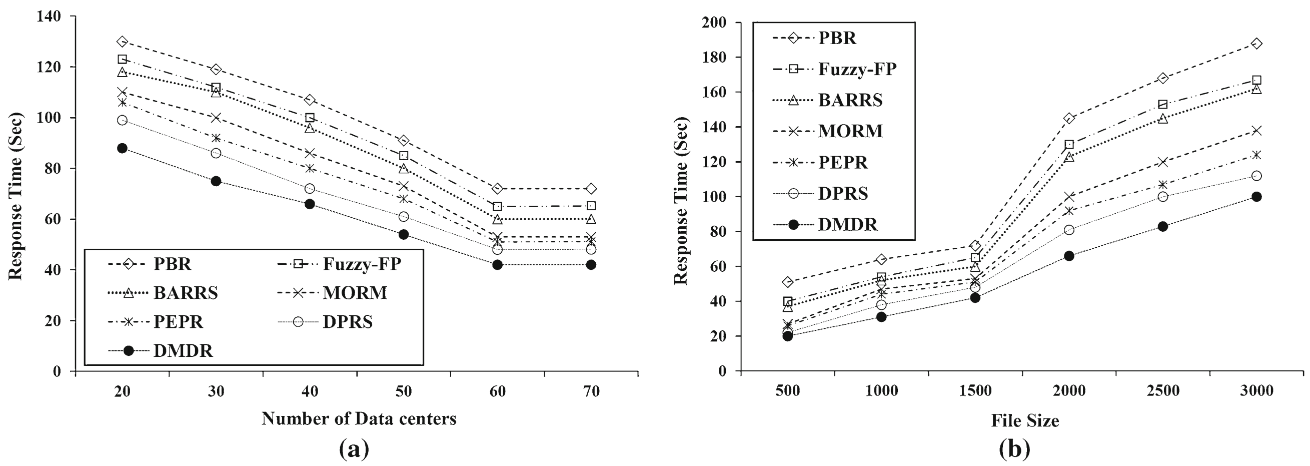


Fig. 9 Average response time based on **a** varying number of data centers. **b** different file size

Figure 9a compares the average response time for different number of data centers. From Fig. 9a, it is clear that as the number of data centers increases, the replication algorithms are able to process the tasks in the low response time with high efficiency because tasks can be distributed on different data centers and the task waiting queue in each data center has small size. Tasks for execution in each data center require several files. In difficult conditions such as low number of data centers and resource constraints, the advantage of the proposed replication method will be more prominent than other algorithms. DMDR considers a proper mechanism to place copy of files in appropriate data center. It can be seen from Fig. 9a, MORM algorithm achieves lower response time compared with fuzzy-FP method (about 13.5% in average) because in MORM algorithm the load variance of data centers is considered as a parameter for replica placement.

Figure 9b presents the results that are obtained when the size of files changes. The size of files is another evaluation factor that can show the advantage of one replication algorithm compared with others. So the replication algorithm that considers appropriate way to identify popular files obtained lower response time in comparison with other methods. It can be seen from Fig. 9b that DPRS obtains lower response time compared with MORM algorithm (about 21%) even in high level of file size, because it uses a dynamic process to find popular file and replicates those files.

We can observe from Fig. 10 that as the number of files increases, the replication algorithms process the tasks in the high response time. The difference of replication algorithms is appeared in difficult conditions such as high number of files. In this situation, the advantage of the proposed method will be more distinguished than other algorithms; the proposed method executes tasks in average 27% faster than other methods.

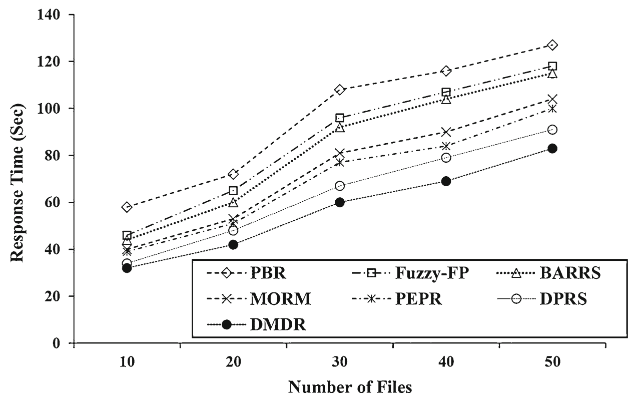


Fig. 10 Average response time based on different number of files

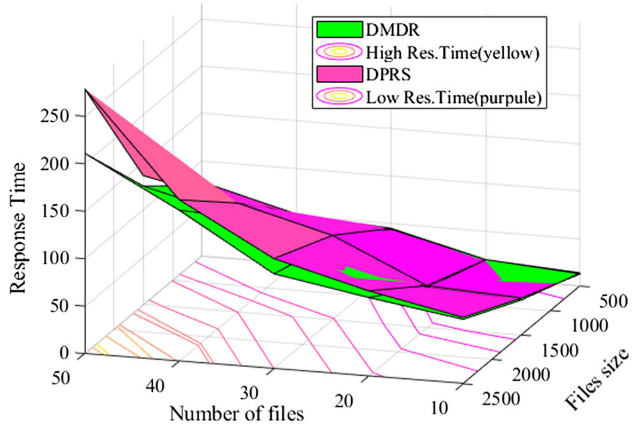


Fig. 11 Average response time for different number of files and file size

Figure 11 shows average response time for the proposed method (DMDR) and DPRS in terms of file size and number of files. We can see from Fig. 11 that in low number of files and file size these two methods have close performance ; DPRS has better performance due to considering an intelligent data placement process for balancing of the system. As the file size and number of files are increased, the response time for the proposed method (DMDR) and DPRS also increased. In high number of files and file size, DMDR method achieves lower response time compared with DPRS because it considers centrality as a parameter for selecting data centers and also it applies a good strategy for replacing replicas.

5.1.1.2 Storage usage We know that various methods proposed for data replication may lead to different storage usage, affecting the storage capacity planning. The storage usage for replicas by replication schema can be expressed as in Eq. (7):

$$Storage\ Usage = \frac{Filled\_Space\_Available}{Space} \tag{7}$$

Storage is undoubtedly one of the key elements in cloud environment, so beneficial information can be provided by monitoring the use of storage resources. This can be valuable in proposing an efficient replication methodology from two important points of view: on the one hand, the objective could be the minimization of storage consumption, perhaps because the resource cost is proportional to the amount being used; on the other hand, its cost might be constant and one would then goal at maximizing the use of storage space.

We have performed four different of tests based on different number of tasks, different number of data centers, different file size and different number of files for storage usage.

Tables 13–16 show the simulation structure for test numbers 1–4, respectively. From Figs. 12, 13, 14a, b, we can observe that MORM achieves lower storage usage compared with PBR, Fuzzy-FP, and BARRS. Because in MORM instead of storing replicas in many sites, they can be placed in the best locations so that the storage usage can be reduced. From Fig. 12 and referring to the resource consumption, by using DPRS, the storage usage is decreased by outperforming PEPR, 23%. This is because DPRS only keeps frequently accessed files that are a small part of the whole data set, thus decreasing the storage usage of replication. DMDR strategy consumed almost half of the storage capacity used by PBR, while PBR method filled more than half of the entire storage capacity available in the cloud environment. This is because DMDR only replicates the important files based on the *Value* assignment to each file. When set of files are candidate for replication in the selected site and there is no sufficient space, the DMDR compares the aggregated *Value*. If their summation of *Value* for is “higher,” these files will be replicated. Otherwise, the file will be accessed remotely.

We can see from Fig. 14a, as the size of file is increased the storage usage by different replication method is also increased. The proposed algorithm (DMDR) compared with BARRS and Fuzzy-FP can reduce storage usage by 35% and 39%, respectively. Since using an appropriate replacing strategy is one of the main factors especially when sufficient space is not available for storing a new replica.

Figure 15 shows storage usage for the proposed method (DMDR) and DPRS in terms of file size and number of files. We can observe form Fig. 15, in low number of files and file size, these two methods have close performance. In high number of files and file size, the proposed method achieves lower storage usage in comparison with DPRS since it deletes replicas with low value and only keeps the valuable files.

5.1.1.3 Number of communications The next evaluation is planned to study the number of communication for DMDR in comparison with other replication methods. It is crucial to reduce the total number of communications for decreasing the access latency and the bandwidth

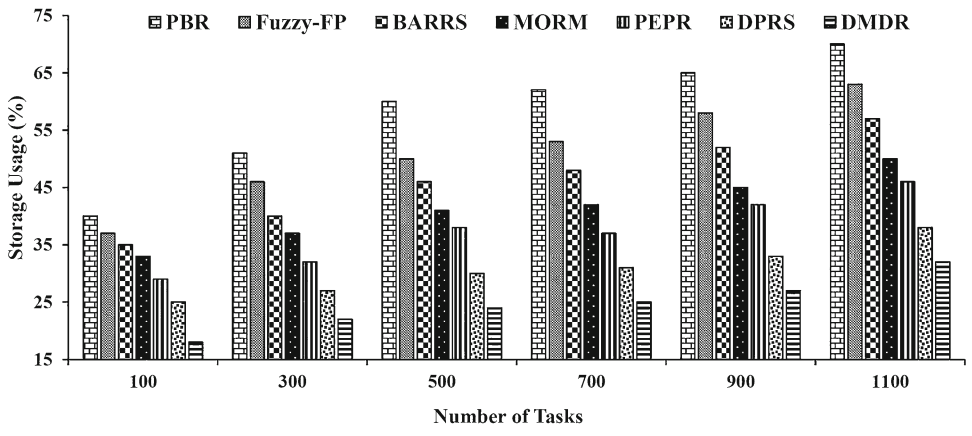


Fig. 12 Storage usage based on varying number of tasks

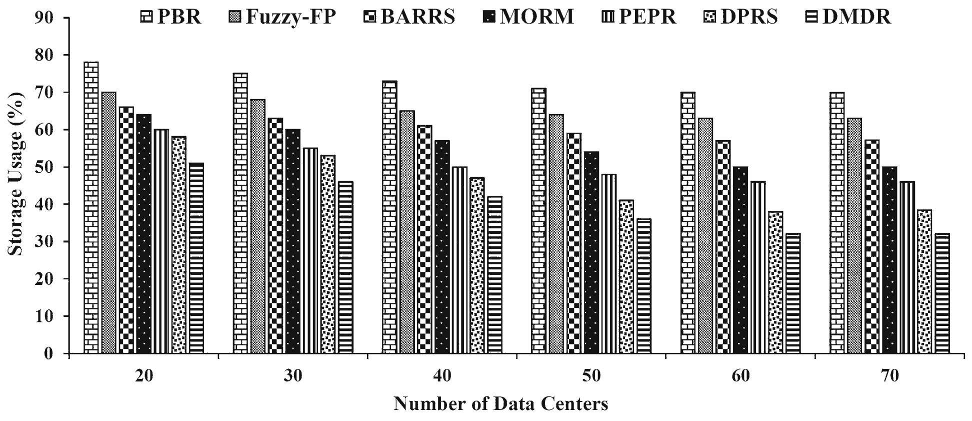


Fig. 13 Storage usage based on varying number of data centers

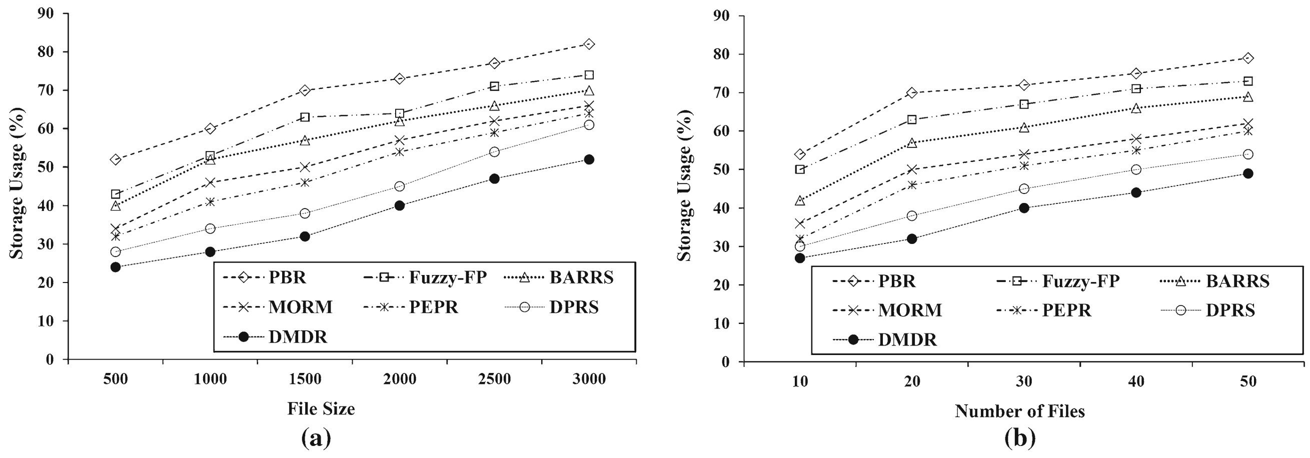


Fig. 14 Storage usage based on different **a** file size, **b** number of files

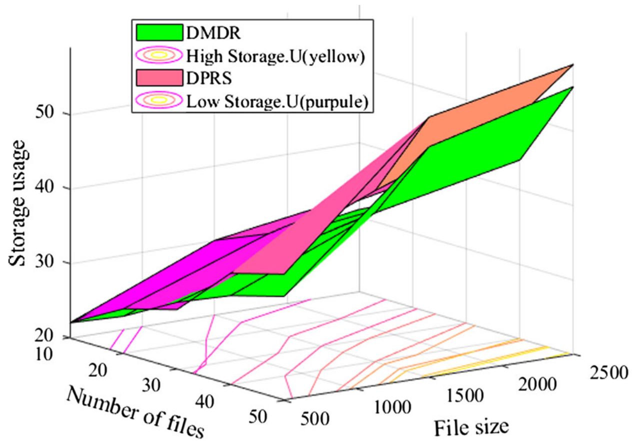


Fig. 15 Storage usage for different number of files and file size

congestion. We have performed three different of tests based on different number of tasks, different number of data centers, and different number of files.

Tables 17–19 show simulation structure for test numbers 1–3, respectively. In Fig. 16, it is observed that DMDR outperforms by 9% over DPRS and by 15% over PEPR. Instead of storing individual files, an advanced DMDR replication strategy replicates the most related files. In addition, DMDR stores replicas in the best site (i.e., best site that has the great number of access and most central) based on the temporal and geographical locality concepts. Consequently, it can decrease total communications between data centers.

Based on the results of Fig. 17 and Fig. 18, we can infer that the number of communications in our approach is fewer than other approach. This optimization has a big impact, since the number of communications increases rapidly with increasing the number of files. When the number of data centers is low, more number of tasks are assigned to data centers thus more number of files are available for tasks in local location of tasks, and as a result number of communications are low but average response time is incredibly increased by considering the lower number of data centers for systems. We can see from Fig. 18, PEPR has lower number of communication compared with PBR algorithm because (about 18%) PEPR considers storage space and computational load of the data centers for replica placement.

Fig. 16 Number of communications based on varying number of tasks

Test number 1

Table 17. Communication simulation variables: number of tasks.

Parameter	Value
Total number of tasks	**[100-1100]**
Number of data centers	60
Number of files	20
Size of files	1500 Mb

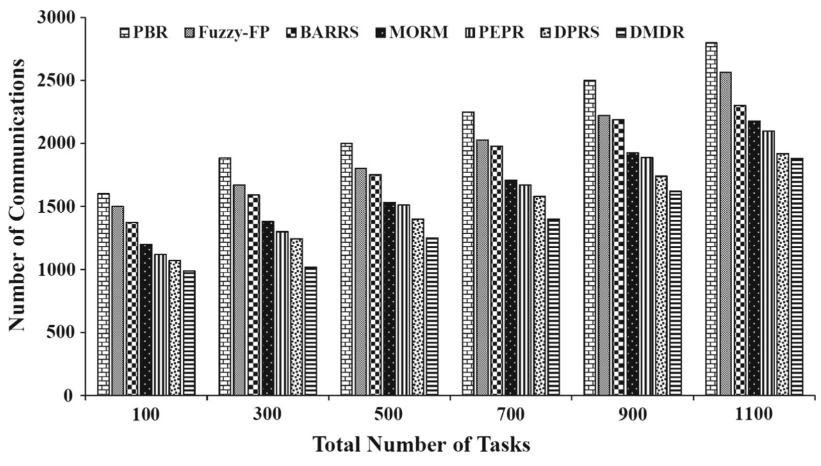
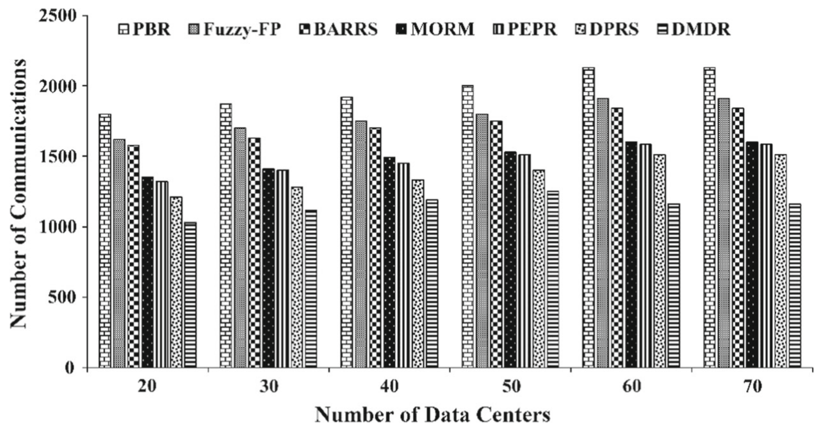


Fig. 17 Number of communications based on varying number of data centers

Test number 2

Table 18. Communication simulation variables: number of data centers.

Parameter	Value
Number of tasks	1100
Number of data centers	**[20-70]**
Number of files	20
Size of files	1500 Mb



5.1.1.4 Effective network usage (ENU) The effective network usage indicates the ratio of files transferred to files requested. So a low value of ENU demonstrates that the data replication method is successful in storing files in the proper locations. As follow equation, N_{rfa} is the number of access times that local site gets a file from remote sites, N_{fa} is the total number of file replication operation, and N_{lfa} is the number of times that computing element uses a file locally (Cameron et al. 2003).

$$E_{enu} = \frac{N_{rfa} + N_{fa}}{N_{lfa}} \qquad (8)$$

The range of effective network usage is from zero to one. A lower value of ENU demonstrates that the network bandwidth is utilized more efficiently. Figure 19 shows effective network usage by different replication algorithms, when based on Table 20. It is obvious that data replication is a time- and cost-consuming process. But, performing no

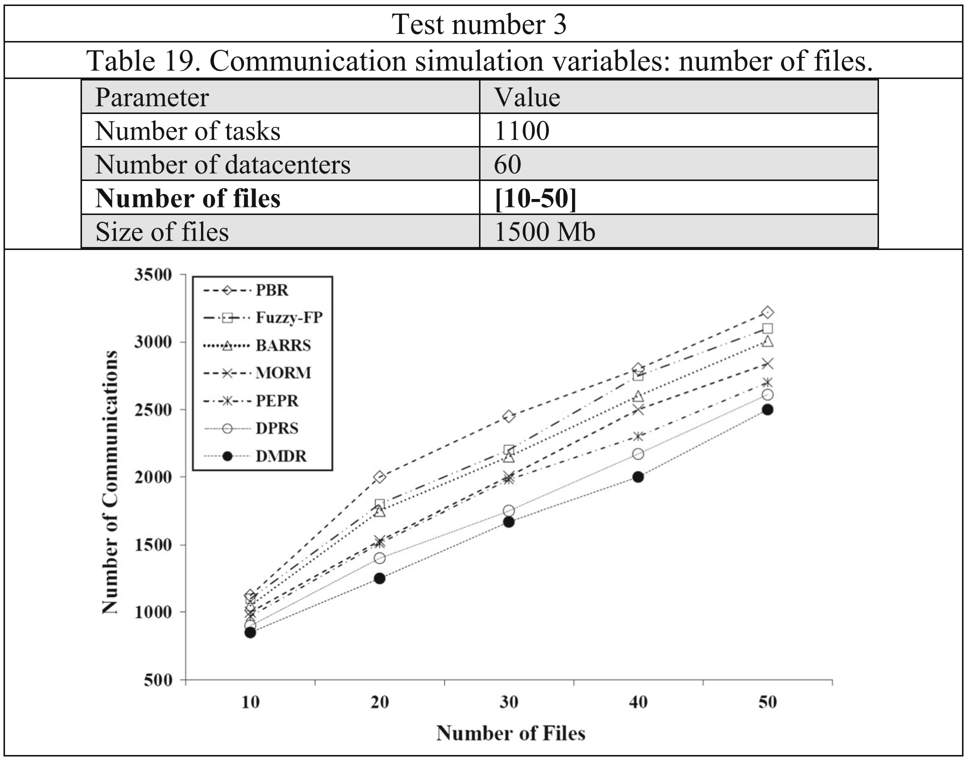

Test number 3

Table 19. Communication simulation variables: number of files.

Parameter	Value
Number of tasks	1100
Number of datacenters	60
Number of files	**[10-50]**
Size of files	1500 Mb

Fig. 18 Number of communications based on different number of files

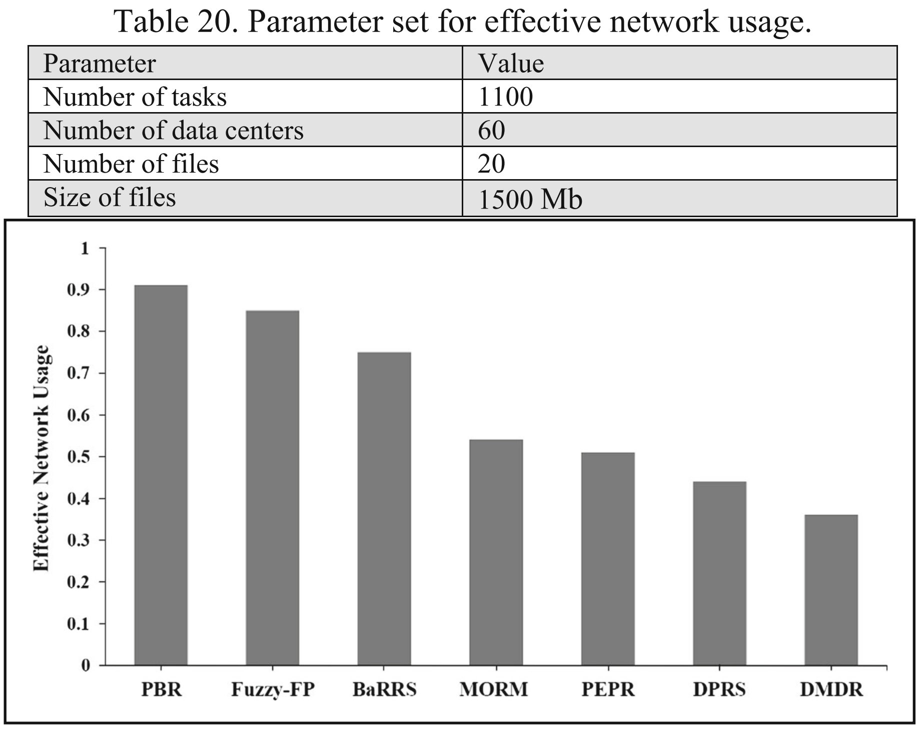


Table 20. Parameter set for effective network usage.

Parameter	Value
Number of tasks	1100
Number of data centers	60
Number of files	20
Size of files	1500 Mb

Fig. 19 Effective network usage for different data replication strategies

replication method has been proved to be improper in comparison with the simplest replication strategy. The effective network usage of DPRS is better in comparison with MORM and PEPR strategies because of considering access cost in replica placement. As depicted in Fig. 19, the ENU of PEPR is lower by 38% compared to PBR algorithm.

This is because PEPR transfers most of files at the intra-data center level using replicas that are variable in the other sites in the local region with cheaper bandwidth. If accessing a file from remote site is more profitable than storing a new replica in local site, PEPR perform such an action to improve overall performance. When compared to the PBR, Fuzzy-FP, BaRRS, MORM, PEPR, and DPRS algorithms, the DMDR algorithm performs better because of considering centrality and number of accesses in replica placement. Most of the time replica of necessary files is available in the local data center. We can conclude from this evaluation that knowledge discovery process is beneficial in the determination of users' future access behavior in cloud environment.

5.1.1.5 Hit ratio Hit ratio is as the proportion of total number of local file accesses to all accesses (i.e., local file accesses, total number of replications and total number of remote file accesses). Figure 20 explains the hit ratio among different replication algorithms, when based on Table 21. It is easy to see through the results of Fig. 20, DMDR has the highest value of hit ratio in comparison with other replication algorithms. In DMDR strategy, total number of local accesses has been increased by storing replica in appropriate location and avoiding unnecessary replication. When a site requests for a file and replication is triggered for it.

Adjacent files, which are extracted by mining the history of file accesses, if these are beneficial, are also pre-fetched. Therefore, total number of replications and remote accesses has been decreased and consequently hit ratio has been increased. As a result, prediction of future file requests according to the results of the data mining process has a great impact on enhancing the effectiveness of the replication algorithm by planning for a replication a priori, more dynamic and more adaptable to the user behavior.

5.1.1.6 Replication frequency By decreasing of the replication frequency, i.e., the ratio of how many replication trigger per data access, the ability of replication algorithm to sort data file in the appropriate sites is improved. The results of the replication frequency are presented in Fig. 21 when based on Table 22. The replication creation are higher than 0.73 for PBR algorithm, which shows that at least 0.73 replicas are created for a data access. The replication frequency of PBR method is too high which renders it not feasible in the real environment. Figure 21 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 DMDR is less than 0.21, i.e., for successive 100 data access; 21 replicas must be created. The ability of DMDR is to save valuable replicas during the change of environment. These characteristics can increase the availability beside of reducing unnecessary replications.

Fig. 20 Hit ratio for different data replication algorithms

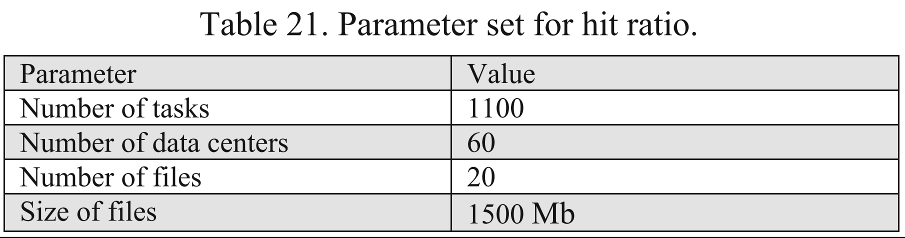
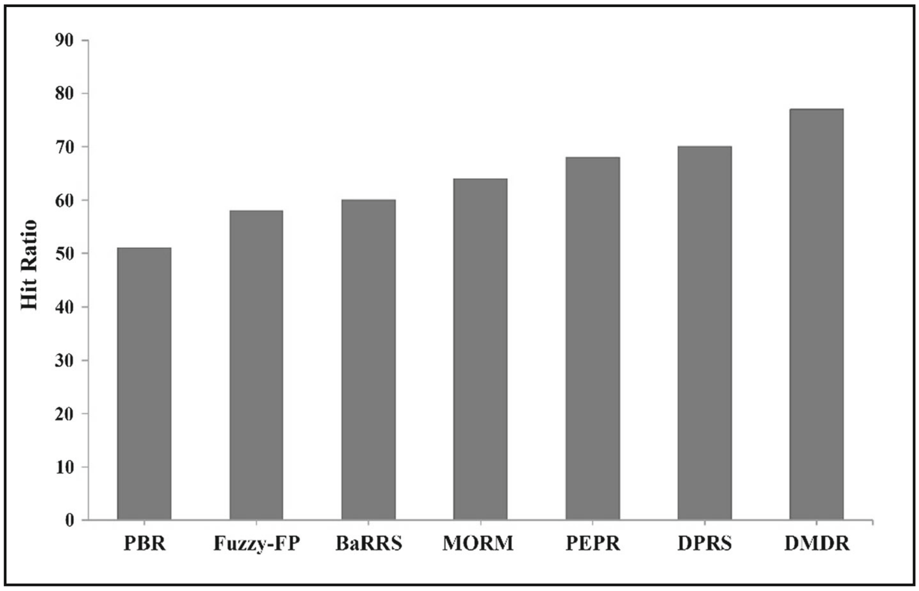


Table 21. Parameter set for hit ratio.

Parameter	Value
Number of tasks	1100
Number of data centers	60
Number of files	20
Size of files	1500 Mb



5.1.2 Effect of thresholds

5.1.2.1 Impact of *MinSupport* threshold We investigate the impact of *MinSupport* on the ENU and average response time. The simulation results for the different *MinSupport* and fixed threshold of *Min-All-Confidence* are shown in Figs. 22 and 23. It is obvious that DMDR strategy gives reasonable average response time in low *MinSupport* value (from 0 to 0.5), which shows that it is efficient even if it does not take into account only high correlated files.

In high value of *MinSupport*, the performance is deteriorated. The main reason is that replication strategy extracts only few frequent patterns.

5.1.2.2 Impact of *Min-All-Confidence* threshold Now we study the impact of variation *Min-All-Confidence* threshold for fixed values of the *MinSupport* in term of average response time and ENU. As illustrated in Figs. 24 and 25, optimum value of *Min-All-Confidence* equal to 0.5. Moreover, a rapid degradation of performance metrics is observed for a *Min-All-Confidence* value larger than 0.5.

Comparing Figs. 22, 23, 24 and 25, we can see that DMDR strategy replicates efficiently when appropriate number of frequent correlated patterns is determined.

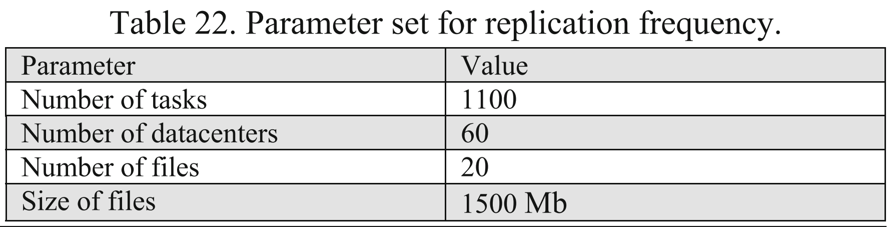


Table 22. Parameter set for replication frequency.

Parameter	Value
Number of tasks	1100
Number of datacenters	60
Number of files	20
Size of files	1500 Mb

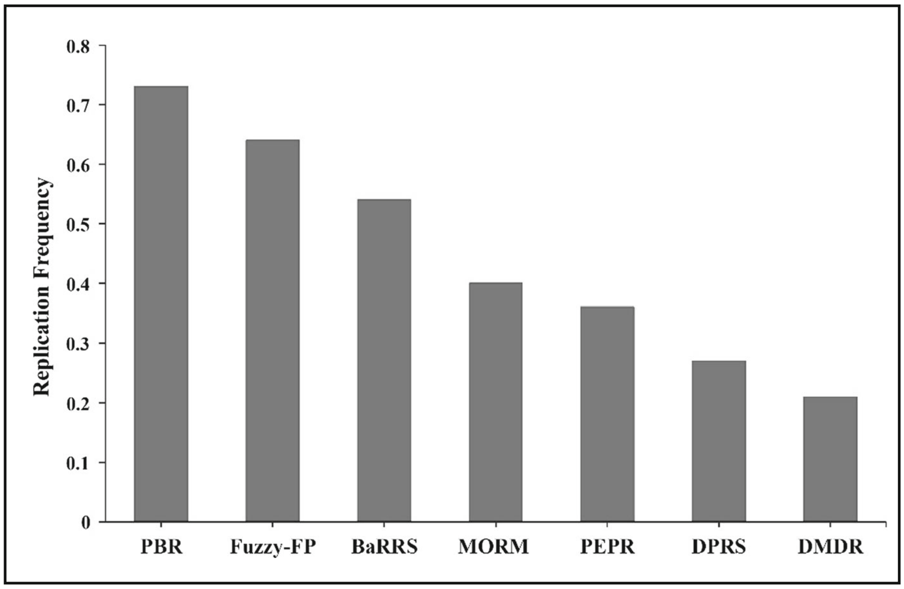


Fig. 21 Replication frequency for different data replication strategies

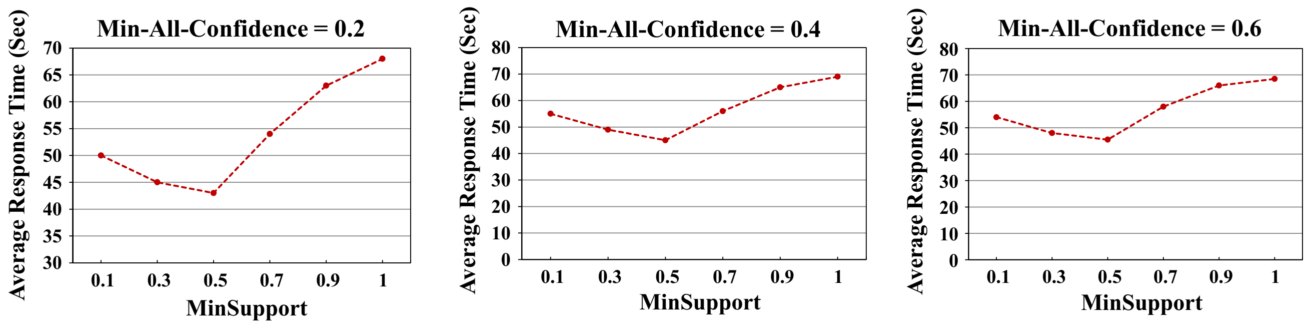


Fig. 22 Average response time in different *MinSupport* value

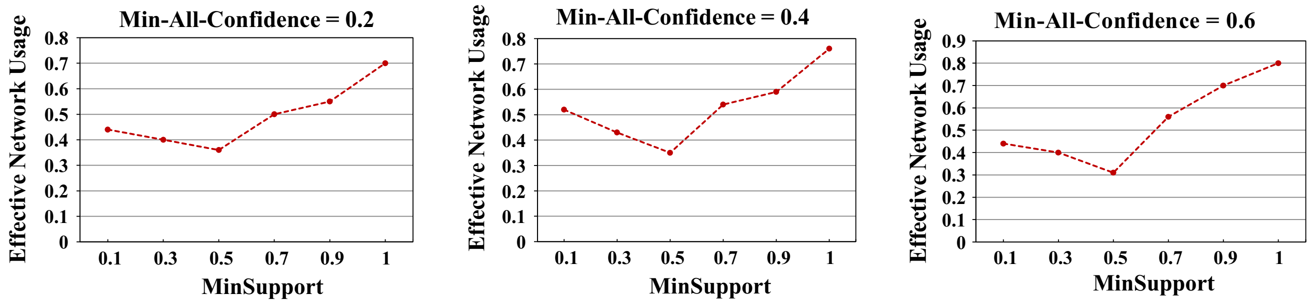


Fig. 23 ENU in different *MinSupport* value

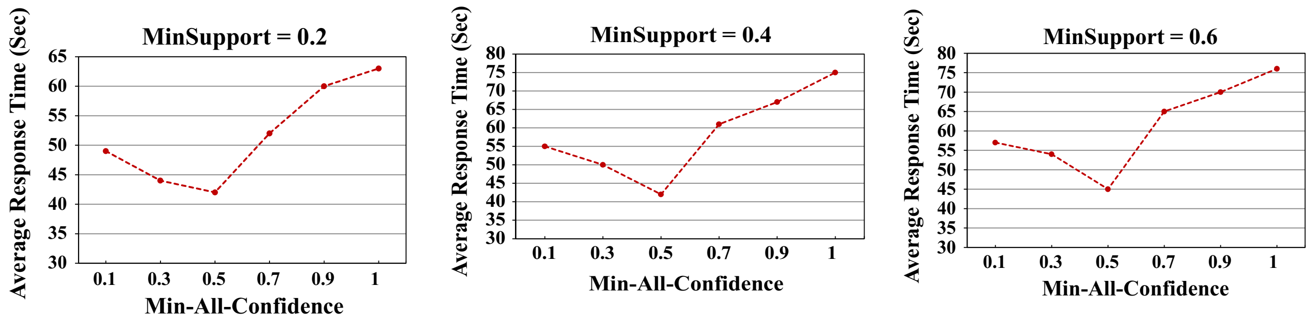


Fig. 24 Average response time in different *Min-All-Confidence* value

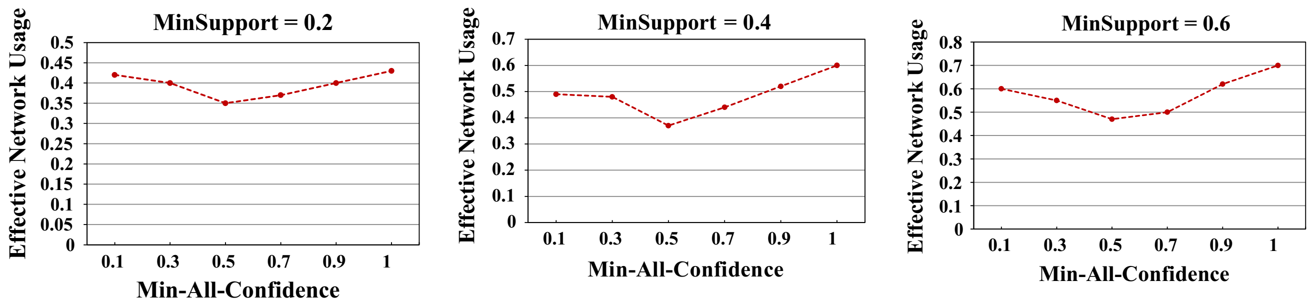


Fig. 25 ENU in different *Min-All-Confidence* value

5.1.3 Network topology architecture

In our simulation, the input network topologies are generated under Waxman model (Jung et al. 2012). In Waxman model, N data centers are randomly placed into a square. A link is inserted between two data centers u and v, with a probability $p(u,v) = \beta e^{-d(u,v)/\alpha L}$, where $d(u, v)$ represents the Euclidean distance between u and v, L is the largest possible distance between two data centers in the square. β and α are Waxman parameters, and $\alpha, \beta \in (0, 1]$. Figure 26 shows impact different value of α and β on average response time, when these two parameters have small value, it means the probability that exist links between data centers is low and the system has low number of links. A low number of links means that a heavier load on the system and response time getting high value.

Table 23 shows the simulation values for bandwidth test. In this test, we evaluate average response time in terms of different bandwidth value. As shown in Fig. 27, with the bandwidth reducing, the response time increases dramatically, especially when the bandwidth is lower than 700. The difference in response time of the proposed algorithm with the other replication algorithms for the bandwidth 100 is tangible. DMDR algorithm reduces bandwidth consumption by storing popular files based on access history in the best location. Therefore, most of time necessary files for task execution are locally available and do not need to transfer large files remotely.

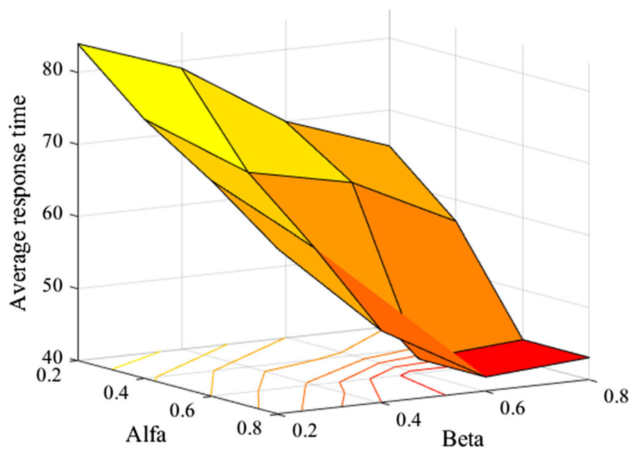


Fig. 26 Average response time in terms of network topology parameters

6 Real industry applications

In many real commercial applications, data files tend to have correlations (i.e., some files are requested by the same tasks). Examples include computational genomics, astrophysics, climate change modeling, biomedical information research network (BIRN), space shuttle flight simulation, high-energy physics, and earth observation (Russel et al. 2001; Keator et al. 2008; ESA 2010). Such data-intensive applications that mainly try to solve some of the most important problems facing human beings are becoming increasingly prevalent in various scientific and industrial disciplines.

6.1 High-energy physics

One of the most important activities in the department of Energy funded SciDAC-Data project is the analysis of the more than 410,000 high-energy physics files that have been generated by the Fermilab storage facilities. More than 5.6 million recorded projects process and analyze these files. The SciDAC-Data project began to analyze these huge files in order to present data-driven descriptions of high-energy physics (HEP) workflows and data management (Ding et al. 2016). The results include the investigation of file popularity, files access dependency, access pattern, and the correlations file overlap. Moreover, experiments present that how tasks and scheduling methods can be combined with various replication algorithms to execute the necessary tasks and find the requirements of physics analysis for the next generation in HEP computing. Therefore, the meta-information of these data along with HEP analysis chains are useful to understand how modern computing and data delivery is being performed.

Fig. 27 Average response times for different bandwidth values

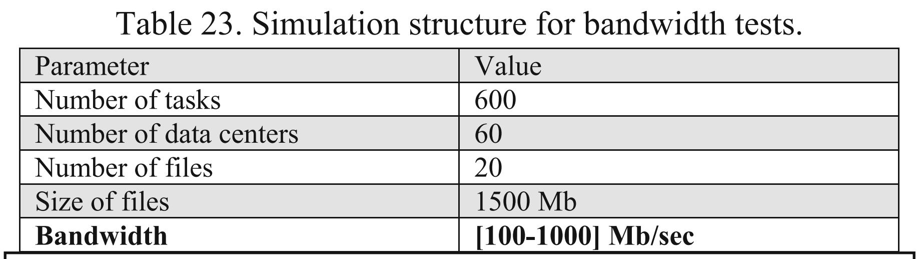
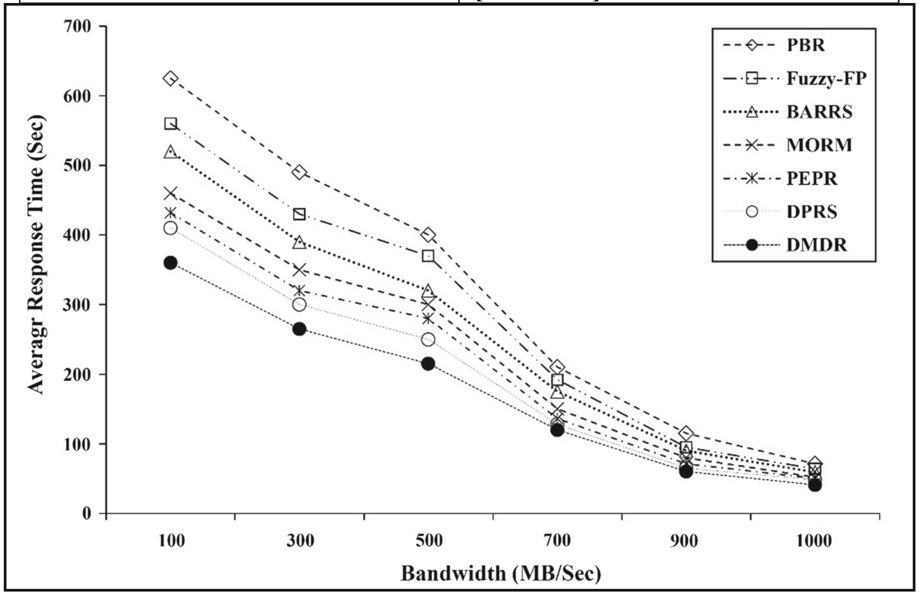


Table 23. Simulation structure for bandwidth tests.

Parameter	Value
Number of tasks	600
Number of data centers	60
Number of files	20
Size of files	1500 Mb
Bandwidth	**[100-1000] Mb/sec**



6.2 Bioinformatic

Life science application is another motivating example that contains a large number of independent tasks and processes several files on computational servers. PattInProt is one of the applications which data replication can improve performance. This application focuses on the signatures and sequences of proteins. Genomic programs like full genomes sequencing projects generate huge files and present them to the community (Bernal et al. 2001). In addition, there are different bioinformatic tools for analyzing these large files. But most of them can be represented based on Fig. 28.

We can see from Fig. 28 that such bioinformatics applications access to several international protein sequence datasets such as Swiss-Prot/TrEMBL (Desprez and Vernois 2006). Therefore, data replication technique is necessary for reducing response time.

6.3 Simulation results

We have to use the simulator because target platforms are distributed and have multiple administrative domains. We set three tests (see Tables 24–26) to realistically represent a data-intensive task execution and evaluate performance of proposed method compared with other methods. Based on bioinformatics example, we consider requests as tasks, data centers as computational servers and files as data bases.

Figure 29a shows the performance of proposed replication algorithm for response time in terms of bandwidth and number of tasks compared with DPRS. We consider [700–1000] for network bandwidth. Figure 29a shows that the proposed algorithm for a high number of tasks leads to a greater reduction in time (10% performance better compared with DPRS).

Figure 29b shows response time for DMDR and DPRS algorithms in different workloads (number of files and file size). From Fig. 29b, we can see that with the increasing number of workloads, our strategy has shorter response time (about 5%) than the other one strategy, indicating our strategy have more effective performance as workload increasing. Since, our strategy reduces the number of remote file accesses.

Figure 30 shows response time based on different number of tasks for various replication methods. In low loads, response times for DPRS, DMDR, BARRS, and MORM are not impacted significantly. For example, response time between DPRS and DMDR is about 150

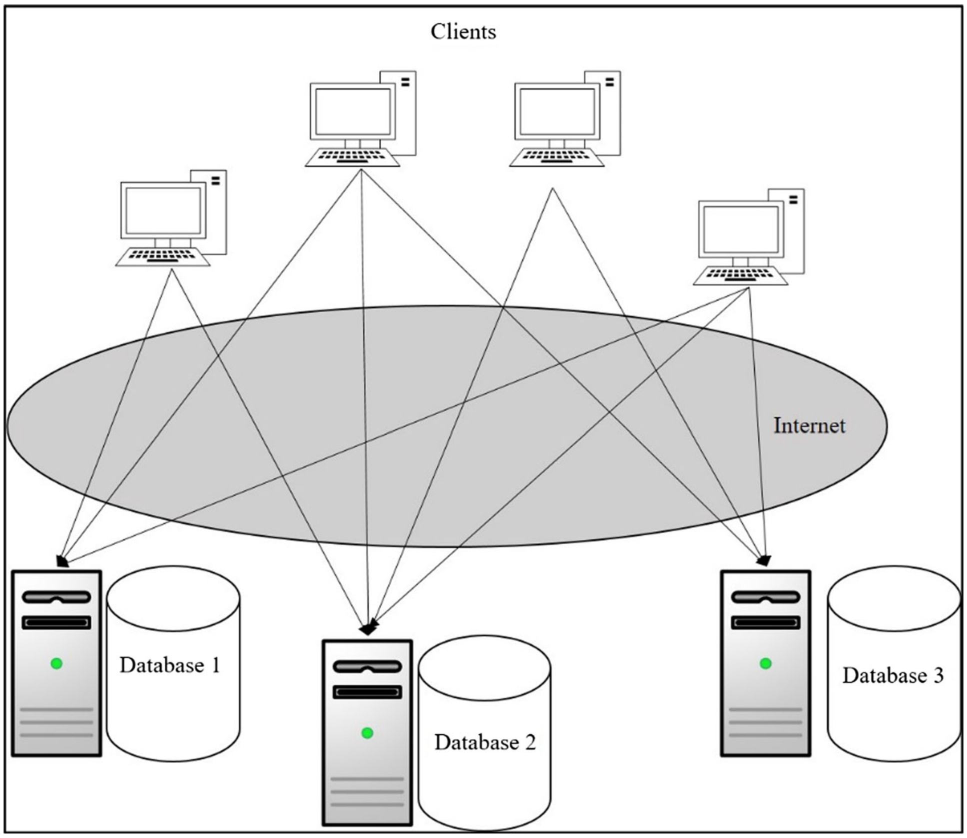


Fig. 28 Current view of bioinformatics application (Desprez and Vernois 2006)

Test number 1

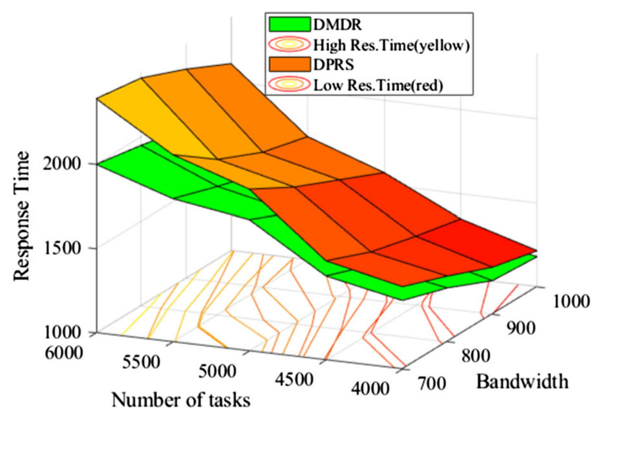
Table 24. Example parameters: number of tasks and bandwidth.

Parameter	Value
Number of tasks (requests)	**[4000-6000]**
Number of datacenters (computational servers)	60
Number of files (data bases)	50
Size of files (data bases)	1500
Bandwidth	**[700-1000]**

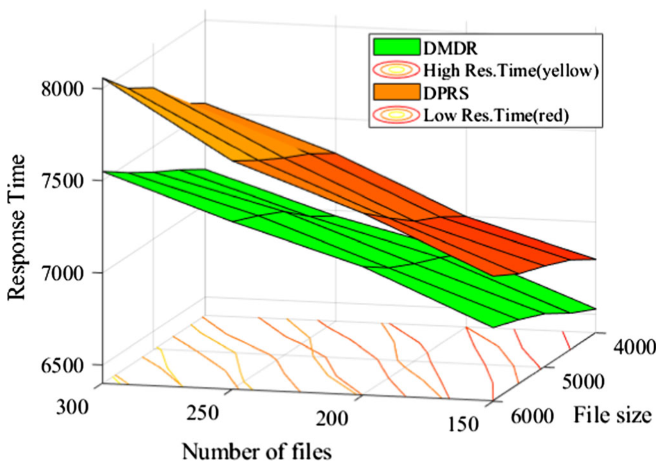
Test number 2

Table 25. Example parameters: number of files and file size.

Parameter	Value
Number of tasks (requests)	6000
Number of datacenters (computational servers)	60
Number of files (data bases)	**[150-300]**
Size of files (data bases)	**[4000-6000] Mb**
Bandwidth	800



(a)



(b)

Fig. 29 Response time based on different number of **a** tasks and bandwidth, **b** files and file size

and between PBR and Fuzzy-FP is about 100. However, significant response time degradations occur at higher number of tasks where more and more tasks with different files are added to the system. We can see from Fig. 30 that the proposed method achieves better performance in comparison with the other algorithms because of using appropriate replacing and placing strategies during replication.

7 Conclusion

Nowadays, in various scientific disciplines, huge cloud-based applications have put forward higher request for storage and computing resources. It is essential to enhance data availability and the performance of the cloud system. In order to meet these goals, the replication technique is used. First, we designed a heterogeneous cloud environment to understand the relation between data file access, and replication cost. Then, we introduce a Data Mining-based Data Replication (DMDR) strategy. This replication method has three main steps and is suitable for replicating data files in cloud. In the first phase, maximal frequent correlated files are extracted based on file access history. In the second phase, DMDR stores replicas on a suitable location, with reduced access latency according to the centrality factor. In the third phase, the replacement decision is made in order to provide better response time. It replaces the replicas based on the importance value. The new replication schema was simulated using the CloudSim toolkit package. Our proposed mechanism replicates the data over the cloud nodes reasonably well and is easily implementable in a real environment. It stands good without increasing additional overheads. From the experiment results, it can be concluded that DMDR can achieve a good improvement of performance in term of average response time, effective network usage, replication frequency, 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 business driven limitations. Also, we want to

Test number 3

Table 26. Example parameters: different number of tasks.

Parameter	Value
Number of tasks (requests)	**[4000-6500]**
Number of data centers (computational servers)	60
Number of files (data bases)	50
Size of files (data bases)	[4000]Mb
Bandwidth	800

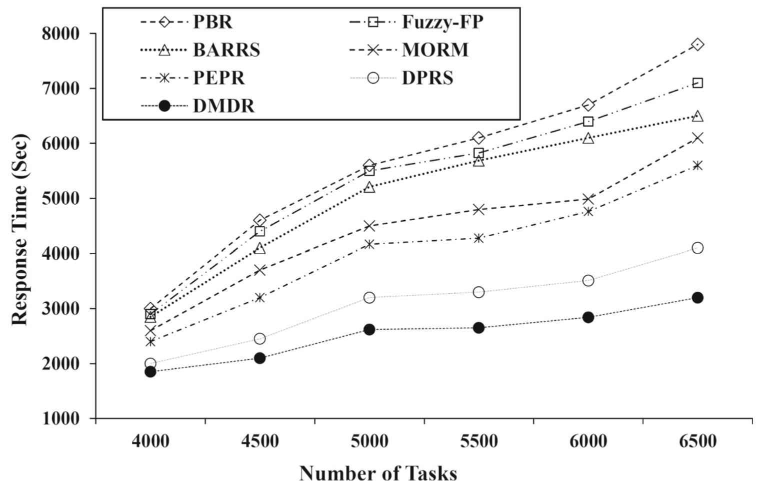


Fig. 30 Response time for large number of tasks

perform more realistic evaluation with data accesses in actual applications.

Compliance with ethical standards

Conflict of interest N. Mansouri declares that he has no conflict of interest. M.M. Javidi declares that he has no conflict of interest. B. Mohammad Hasani Zade declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abouzeid A, Bajda-Pawlikowski K, Abadi D, Silberschatz A, Rasin E (2009) HadoopDB A: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proc VLDB Endow 2(1):922–933

Ahmed I, Socci C, Severini F, Yasser QR, Pretaroli R (2018) Forecasting investment and consumption behavior of economic agents through dynamic computable general equilibrium model. Financ Innov 4:7

Al-Asaly MS, Hassan MM, Alsanad A (2019) A cognitive/intelligent resource provisioning for cloud computing services: opportunities and challenges. Soft Comput 32(19):9069–9081

Alghamdi M, Tang B, Chen Y (2017) Profit-Based file replication in data intensive cloud data centers. In: IEEE international conference on communications

Barroso LA, Clidaras J, Holzle U (2013) The datacenter as a computer: an introduction to the design of warehouse-scale machines, 2nd edn. Morgan and Claypool Publishers, San Rafael

Bernal A, Ear U, Kyrpides N (2001) Genomes online database (GOLD): a monitor of genome projects world-wide. Nucl Acids Res 29:126–127

Bojanova I, Samba A (2011) Analysis of cloud computing delivery architecture models. In: IEEE workshops of international conference on advanced information networking and applications, pp 453–458

Bouyer A, Karimi M, Jalali M (2009) An online and predictive method for grid scheduling based on data mining and rough set. In: Computational science and its applications, lecture notes in computer science vol 5592, pp 775–787

Brin S, Motwani R, Silverstein C (1997) Beyond market baskets: generalizing association rules to correlations. In: Proceedings of the ACMSIGMOD international conference on management of data, pp 265–276

Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50

Cameron DG, Carvajal-schiaffino R, Paul Millar A, Nicholson C, Stockinger K, Zini F (2003) UK grid simulation with OptorSim. UK e-Science all hands meeting

Casas I, Taheri J, Ranjan R, Wang L, Zomaya AY (2017) A balanced scheduler with data reuse and replication for scientific workflows in cloud computing. Future Gener Comput Syst 74:168–178

Cassandra (2011) http://incubator.apache.org/cassandra/. Accessed 2019

Cooper B, Baldeschwieler E, Fonseca R, Kistler J, Narayan P, Neerdaels C, Negrin T, Ramakrishnan R, Silberstein A, Srivastava U, Stata R (2009) Building a cloud for Yahoo! IEEE Data Eng Bull 32(1):36–43

Croda RMC, Romero DEG, Morales SOC (2019) Sales prediction through neural networks for a small dataset. Int J Interact Multimed Artif Intell 5(4):35–41

Desprez F, Vernois A (2006) Simultaneous scheduling of replication and computation for data-intensive applications on the grid. Journal of Grid Computing 4(1):19–31

Ding P, Aliaga L, Mubarak M, Tsaris A, Norman A, Lyon A, Ross R (2016) Analyzing how we do Analysis and Consume Data, Results from the SciDAC-Data Project. Argonne National Lab. (ANL), Argonne, IL (United States)

Doraimani S (2007) Filecules: a new granularity for resource management in grids (Master thesis). University of South Florida, USA

Duan R, Prodan R, Fahringer T (2006) Data mining-based fault prediction and detection on the grid. In: Proceedings of the 15th IEEE international symposium on high performance distributed computing, pp 305–308

Elango P, Kuppusamy D (2016) Fuzzy FP-tree based data replication management system in cloud. Int J Eng Trends Technol 36:481–489

ESA (2010) Observing the earth. http://www.esa.int/Our_Activities/Observing_the_Earth. Accessed 2019

Grace RK, Manimegalai R (2014) Data access prediction and optimization in data grid using SVM and AHL classifications. Int Rev Comput Softw 9(7):1188–1194

Gupta BB, Agrawal DP, Yamaguchi S, Sheng M (2018) Advances in applying soft computing techniques for big data and cloud computing. Soft Comput 22(23):7679–7683

Hamrouni T, Faouzi SS, Charrada B (2015) A data mining correlated patterns-based periodic decentralized replication strategy for data grids. J Syst Softw 110:10–27

Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques. Morgan Kaufmann Publishers, Burlington

HBase (2016) http://hadoop.apache.org/. Accessed 2019

Hong TP, Kuo CS, Chi SC (1999) Mining association rules from quantitative data. Intell Data Anal 3(5):363–376

Jalil AM, Hafidi I, Alami L, Khouribga E (2016) Comparative study of clustering algorithms in text mining context. Int J Interact Multimed Artif Intell 3(7):42–45

Jung JK, Jung SM, Kim TK, Chung TM (2012) A study on the cloud simulation with a network topology generator. Int J Comput Inf Eng 6(11):1312–1315

Keator DB, Grethe JS, Marcus D, Ozyurt B, Gadde S, Murphy S, Pieper S, Greve D, Notestine R, Bockholt HJ, Papadopoulos P (2008) A national human neuroimaging collaboratory enabled by the biomedical informatics research network (BIRN). IEEE Trans Inf Technol Biomed 12(2):162–172

Khalili AS (2019) A Bee Colony (Beehive) based approach for data replication in cloud environments. Lecture notes in electrical engineering. Nature Singapore Pte Ltd, Singapore, pp 1039–1052

Khanli LM, Isazadeh A, Shishavanc TN (2011) PHFS: a dynamic replication method, to decrease access latency in the multi-tier data grid. Future Gener Comput Syst 27(3):233–244

Ko SY, Morales R, Gupta I (2007) New worker-centric scheduling strategies for data-intensive grid applications. In: Proceedings of the 8th ACM/IFIP/USENIX international conference on middleware, pp 121–142

Kou G, Lu Y, Peng Y, Sh Y (2012) Evaluation of classification algorithms using MCDM and rank correlation. Int J Inf Technol Decis Mak 11(1):197–225

Kou G, Peng Y, Wang G (2014) Evaluation of clustering algorithms for financial risk analysis using MCDM methods. Inf Sci 275:1–12

Kou G, Chao X, Peng Y, Alsaadi FE, Viedma EH (2019) Machine learning methods for systemic risk analysis in financial sectors. Technol Econ Dev Econ 25(5):716–742

Lee YK, Kim WY, Cai YD, Han J (2003) COMINE: efficient mining of correlated patterns. In: Proceedings of the 3rd IEEE international conference on data mining, pp 581–584

Long SQ, Zhao YL, Chen W (2014) MORM: a multi-objective optimized replication management strategy for cloud storage cluster. J Syst Architect 60:234–244

Lou C, Zheng M, Liu X, Li X (2014) Replica selection strategy based on individual QoS sensitivity constraints in cloud environment. Pervasive Comput Netw World 8351:393–399

Manjula S, Indra Devi M, Swathiya R (2016) Division of data in cloud environment for secure data storage. In: International conference on computing technologies and intelligent data engineering (ICCTIDE)

Mansouri N, Javidi MM (2018a) A hybrid data replication strategy with fuzzy-based deletion for heterogeneous cloud data centers. J Supercomput 74(10):5349–5372

Mansouri N, Javidi MM (2018b) A new Prefetching-aware data replication to decrease access latency in cloud environment. J Syst Softw 144:197–215

Mansouri N, Kuchaki Rafsanjani M, Javidi MM (2017) DPRS: a dynamic popularity aware replication strategy with parallel download scheme in cloud environments. Simul Model Theory 77:177–196

Mansouri N, Mohammad Hasani Zade B, Javidi MM (2019) Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. Comput Ind Eng 130:597–633

Mell P, Grance T (2009) Definition of cloud computing. National Institute of Standard and Technology

Moradi S, Mokhatab Rafiei F (2019) A dynamic credit risk assessment model with data mining techniques: evidence from Iranian banks. Financ Innov 5:15

Mukundan R, Madria S, Linderman M (2014) Efficient integrity verification of replicated data in cloud using homomorphic encryption. Distrib Parallel Databases 32(4):507–534

Newman M (2009) Networks: an introduction. Oxford University Press, Oxford

Nivetha NK, Vijayakumar D (2016) Modeling fuzzy based replication strategy to improve data availability in cloud datacenter. In: International conference on computing technologies and intelligent data engineering

Omiecinski E (2003) Alternative interest measures for mining associations in databases. IEEE Trans Knowl Data Eng 15(1):57–69

Park J, Kim U, Yun D, Yeom K (2017) C-RCE: an approach for constructing and managing a cloud service broker. J Grid Comput 17(1):137–168

Peer Mohamed MS, Swarnammal SR (2017) An efficient framework to handle integrated VM workloads in heterogeneous cloud infrastructure. Soft Comput 21:3367–3376

Peng Y, Gang K, Shi Y, Chen Z (2008) A descriptive framework for the field of data mining and knowledge discovery. Int J Inf Technol Decis Mak 7(4):639–682

Peng Y, Kou G, Wang G, Shi Y (2011) FAMCDM: a fusion approach of MCDM methods to rank multiclass classification algorithms. Omega 39(6):677–689

Qi G, Tsai WT, Li W, Zhu Z, Luo Y (2017) A cloud-based triage log analysis and recovery framework. Simul Model Pract Theory 77:292–316

Rehman Malik SU, Khan SU, Ewen SJ, Tziritas N, Kolodziej J, Zomaya AY, Madani SA, Min-Allah N, Wang L, Xu CZ, Malluhi QM, Pecero JE, Balaji P, Vishnu A, Ranjan R, Zeadally S, Li H (2016) Performance analysis of data intensive cloud systems based on data management and replication: a survey. Distrib Parallel Databases 34:179–215

Russel M, Allen G, Daues G, Foster I, Seidel E, Novotny J, Shalf J, Laszewski G (2001) The astrophysics simulation collaboratory: a science portal enabling community software development. In: Proceedings 10th IEEE international symposium on high performance distributed computing

Saleh A, Javidan R, Fatehikhaje MT (2015) A four-phase data replication algorithm for data grid. J Adv Comput Sci Technol 4:163–174

Sánchez A, Montes J, Dubitzky W, Valdés JJ, Pérez MS, Miguel PD (2008) Data mining meets grid computing: time to dance? In: Dubitzky W (ed) Data mining techniques in grid computing environments. Wiley, New York, pp 1–16

Settouti N, Bechar MEA, Chikh MA (2016) Statistical comparisons of the top 10 algorithms in data mining for classification task. International J Interact Multimed Artif Intell 4:46–51

Thusoo A, Sarma J, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R (2009) Hive—a warehousing solution over a MapReduce framework. In: Proceedings of the VLDB endowment, pp 1626–1629

Torres-Franco E, García JD, Sanjuan-Martinez O, Aguilar LJ, Crespo RG (2015) A quantitative justification to dynamic partial replication of web contents through an agent architecture. Int J Interact Multimed Artif Intell 3(3):82–88

Tos U, Mokadem R, Hameurlain A, Ayav T, Bora S (2018) Ensuring performance and provider profit through data replication in cloud systems. Clust Comput 21(3):1479–1492

Wu T, Chen Y, Han J (2010) Re-examination of interestingness measures in pattern mining: a unified framework. Data Min Knowl Discov 21(3):371–397

Zaki MJ, Meira WJ (2014) Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press, Cambridge

Zhong H, Zhang Z, Zhang X (2010) A dynamic replica management strategy based on data grid. In: Ninth international conference on grid and cloud computing, pp 18–23