Information Systems and e-Business Management
https://doi.org/10.1007/s10257-019-00422-x

ORIGINAL ARTICLE

A novel dynamic data replication strategy to improve access efficiency of cloud storage

Sujaudeen Nannai John[1] · **T. T. Mirnalinee**[1]

Received: 4 May 2019 / Revised: 30 May 2019 / Accepted: 23 July 2019

Abstract

Cloud computing provides on demand services to cloud users, and one among them is storage. Currently, large amount of data gets generated and demand an enormous storage. Users can avail the privilege to store their data remotely and can access them through Internet. Of course, the adoption of cloud lends the kind of storage that the user wants. Since data gets accumulated, the time it takes to store and retrieve the data is very long and difficult. Also, unfortunately the existing method of storage is to be optimized for better performance. The factors that affect the performance of cloud storage are response time, data availability and migration cost. Hence to improve these factors the data can be replicated to multiple locations. The decision on which data to be replicated, number of replicas to be created, where the replica has to be placed, management of the replicated data and the provision of optimal replica to the user are the major challenges involved in dynamic replication. We intend to propose, a novel dynamic data replication strategy with intelligent water drop (IWD) algorithm to address the challenges of replication and for the management of cloud storage. The popularity and size of the data are considered for replication. A swarm intelligence based optimization algorithm named IWD algorithm is used to optimize the process of replication and management of storage in cloud. We have compared our D2R-IWD algorithm with popular optimization techniques such as PSO, GA and found out that our methodology gives better result in terms of access efficiency for several test cases thereby improve the performance of cloud.

Keywords Cloud storage · Dynamic data replication · Intelligent water drop algorithm · Replica management · Performance

✉ Sujaudeen Nannai John
nsujaudeen@gmail.com

T. T. Mirnalinee
mirnalineett@ssn.edu.in

[1] DCSE, SSN College of Engineering, Chennai, Tamilnadu, India

Published online: 06 August 2019

1 Introduction

Cloud computing is based on leveraging the Internet to consume software or other IT services on demand. End users share processing power, space, bandwidth, memory and software. In cloud computing, the resources are efficiently utilized by sharing among different group of users. Users can pay as they consume and only use what they need at any given time. IT decision makers Survey states that 68% of the respondents believe by 2014, more than 50% of their company's IT services are migrated to Cloud platforms. In the contemporary society of Big Data, the data volume is growing faster than storage capacity (Gantz and Reinsel 2012). Each week, Facebook requires extra 60 TB of storage just for new photos (Beaver et al. 2010). YouTube users upload over 400 h of video every minute and it requires 1 Petabyte of new storage every day (Baesens 2014). According to the International Data Corporation (IDC)'s sixth annual study, until 2020 the digital data will double every two years (Gantz and Reinsel 2012). By 2020, approximately 40% of the data in the digital universe will be stored or processed by cloud computing providers (Gantz and Reinsel 2012). Amazon Simple Storage Service (S3) encountered a data corruption problem caused by a load balancer bug (Wang et al. 2015). Amazon Web Services (AWS) suffered major disruptions due to a DynamoDB failure (AWS 2016).

As a matter of fact, nowadays large volumes of data have been generated due to the advancement in technologies and increased number of users of Internet. Cloud computing has become a viable solution for data processing, distribution and storage. It provides storage as a service to the users to store their data remotely and access them whenever and wherever they need, through the internet. Cloud storage is a data storage model based on a virtualized infrastructure where data is stored in a logical pool and the physical storage is extended to various servers. The logical pool contains data that can be accessed by several users simultaneously. In order to improve the access efficiency, replication of data is considered to be the most viable solution. Replication is the process of creating an exact copy of the original data either in the same data center or in a remote location. It helps in improving the access efficiency, response time, availability of data and reduces the bandwidth consumption, cost of migration and user waiting time. There are two types of replication: (1) static replication and (2) dynamic replication. The static replication is achieved through data resource pool and number of replicas is based on history and is usually three. But it is not effective and there is no need to replicate all the data chunks. However, the dynamic replication is a method of replicating the data according to the varying needs of the cloud consumers and also with the changes in the environment (Milani and Navimipour 2016). But still there are major concerns arises in the process of replication like which data should be replicated? When the replication should be done? Where the replica should be placed? How many new replicas should be created? Which data should be provided to the user? Whether the load is balanced in storage? and finally, management of replica and storage as well.

Intelligent water drop (IWD) is a swarm-based nature-inspired optimization algorithm that contains few essential elements of natural water drops, the actions

and reactions that occur between river's bed and the water drops (Shah-Hosseini 2009). It has two main attributes (1) The velocity of the IWD and (2) the amount of soil on the path. When an IWD moves from one location to another, the increase in velocity is inversely proportional to the soil on the path and the soil of IWD increases as it moves between the locations. The IWD selects the path based on the amount of soil on each path. The IWD algorithm may be used for maximization optimization problems. The solutions are incrementally constructed by the IWD algorithm. Therefore, the IWD algorithm is a population-based constructive optimization algorithm. The IWD algorithm has been used for the travelling salesman's problem (TSP) and multiple (or multidimensional) knapsack problem (MKP) with promising results. Both the TSP and the MKP are NP-hard combinatorial optimization problems.

Data is not in demand at all the times. The data that is required may lose its validity because of so many factors. Accuracy and time constraints play a major role to consider the importance of data. Data that is received at proper time is considered to be the most appropriate. Data may be retrieved very late before it is being needed. Some data segments are important only for certain amount of time, then it loses its credibility and popularity. Some more data becomes obsolete after a long period. But still the data is stored and preserved as it may be required at any point. Hence very large amount of storage is used to store those data which is of low or no priority in near future. Also, the replication process in cloud stores duplicate copies of same data at different locations to make the data available at all the times which is of huge wastage of storage. Therefore, the process itself costs additional storage space to be used for the data that is of less importance. So there is a need to properly manage the replication mechanism and simultaneously optimize the storage itself (Mansouri and Javidi 2018 and Milani and Navimipour 2016). Hence, to match the changing access patterns of the consumers, the replication strategy itself should be dynamic to maintain high availability with improved performance. In order to solve these issues we design a novel dynamic data replication strategy to improve the access efficiency of cloud storage using combinatorial optimization algorithm.

In this paper, we have proposed a novel replication strategy to improve access efficiency of cloud storage preserving QoS attributes like high availability, performance and load balancing etc. The rest of this paper is organized as follows: the motivation of the problem is discussed in Sect. 2. Related works on the problems and techniques used for dynamic data replication are investigated in Sect. 3. Detailed description of our proposed architecture and methodology appears in Sect. 4. Experimental results and discussions on test cases appear in Sect. 5. Finally, we conclude the paper with future work in reference section.

2 Related work

Wu et al. (2014) proposed the index name server (INS) to manage file storage, de-duplication and load balancing. When the data being generated continues to increase, there will be redundant copies of the same data, this may affect the network bandwidth and the workload of the server. In order to reduce the workload

due to duplicate files, the index name server (INS) is used. The duplicate copy of the data is eliminated by deduplication technique which divides the files into chunks and calculates a unique hash code or fingerprint for each chunk. After checking the fingerprint of the requested file, the INS confirm whether the same fingerprint exists in the storage, if so the file chunk won't be uploaded and only a reference to the existing data is created. This improves the performance and bandwidth associated with the data transmission but the response time of the cloud storage and the availability of data to multiple users are not considered here.

Sun et al. (2012) used a dynamic data replication strategy (D2RS) which replicates only the popular data files and places them in a balanced way. While the replication of entire data will result in overload of storage node and increases the bandwidth usage due to data transfer. The popular data file is identified using the access histories and by setting different weight for different accessed data. More recently accessed data is given big weight, and if this popularity (replication factor of the data) is greater than a threshold i.e. the replication factor of the system, the replication process is triggered. The number of replicas needed to be created is determined based on the old file availability. The file with less replication factor can be removed and the new replica can be placed. The architecture of the system is in hierarchical manner which contains super data center, main data center and ordinary data center. This architecture minimizes the access time and network load by distributing replicas from super data centers to main or ordinary data centers.

Jeon et al. (2012) proposed a data replication strategy (DRS) is used to improve the data access in the cloud environment. The performance of the DRS is based on the user access patterns. These patterns are more flexible and unpredictable. This algorithm dynamically changes the replication strategy based on the changes in the user access pattern. It analyzes the past data access patterns and perceives the changes and identifies the optimal strategy. The system contains a job broker (JB) and multiple sites containing work node, replica manager (RM) and storage node (SN). The user submits the requests to the JB which in turn dispatches it to various sites. The work node performs the job with the data from the SN. The RM contains the pattern recognizer (PR) and replication optimizer (RO). The PR recognizes the data access pattern changes and the RO dynamically chooses the algorithm to optimize the performance of the system. This dynamic switching algorithm perceives the changes based on the change in the performance and switches the strategy.

Li et al. (2011) and Wei et al. (2010), the cloud storage should provide data with high efficiency and reliability in a cost effective manner. The data replication provides the reliability requirement of the storage. In static replication the number of replica is fixed and is usually 3, this provides reliability but requires more storage space for storing 3 copies of data. This in turn increases the storage cost. A novel cost effective dynamic data replication strategy called cost-effective incremental replication (CIR) was proposed. The CIR reduces the storage cost and fully utilizes the storage resources. It creates the replica in an incremental way and ensures the reliability. This approach uses a reliability model which considers the relationship between reliability and the number of replica. The minimum number of replica is initially one and as time goes on, more number of replica will be created

incrementally to meet the reliability requirement. This approach is suitable for temporarily used data that have low reliability requirement.

Grace and Manimegalai (2014) proposed the dynamic data replication techniques that enhance data availability and increases reliability, which is needed for the compute intensive applications. The two important challenges in creating duplicates of data are replica placement and replica selection. The replica placement can be logically divided into three stages, namely, replication decision, replica selection and file replacement. The replication decision stage decides when and where to create the replica. The second stage, decides which file needs to be replicated. The file replacement stage is activated if the best site to store the new replica does not have sufficient storage space. The replica selection techniques decide which replica location is the best place to access the data for users. If several replicas are available for a file, the optimization algorithm selects the optimal replica based on the parameters: access cost, access latency, bandwidth consumption, balanced workload, maintenance cost, job execution time, response time, fault tolerance and quality of assurance. The duplicate copies can be placed in appropriate nodes and can be selected for job execution. The replica placement and selection techniques also depend on the dynamic architecture of the grid, in which the users can join and leave the grid at any time. In this paper, various replica placement and selection strategies were discussed. These algorithms measure and analyze various parameters such as bandwidth consumption, access cost, scalability, execution time, makes span and storage consumption.

Niu et al. (2013) proposed an intelligent water drop algorithm (IWD) to provide a schedule considering multiple objectives in multiple objective job shop scheduling problem, which is the process of allocating resources to set of tasks based on some criteria like makes span, tardiness and mean flow time of the schedule. Intelligent Water Drops algorithm is one of the recent bio-inspired swarm-based optimization algorithms, which has been used for optimization problems. This algorithm follows the natural movement of water drops that flow in rivers and lakes. It considers to main attributes (i) The velocity of the IWD and (ii) the amount of soil on the path. When an IWD moves from one location to another, the increase in velocity is inversely proportional to the soil on the path and the soil of IWD increases as it moves between the locations. The IWD selects the path based on amount of soil on the path. The path followed by IWD gives the schedule and the locations traversed to reach the target from the source are the consecutive operations to be done to complete a job. For each path the velocity and the soil of IWD is updated which is used to select the optimal schedule.

Long et al. (2014), nowadays large data sets are becoming important part of shared resources. The volume of data is increasing to petabytes. Such large volumes are stored in cloud storage. These data can be managed in a distributed manner by data replication. The data replication involves two questions: (i) how many suitable replica of each data should be created? (ii) Where should these replicas be placed? To address this replica management problem a multi-objective off-line optimization approach with an improved artificial immune algorithm is used. With this strategy, five objectives namely: file unavailability, mean service time, load variance, energy consumption and mean access latency is addressed. This approach formulated

a mathematical model to describe the objectives along with their tradeoffs. The MORM strategy selects the individuals (data node) from the population based upon the storage space available, to place the replica. The individuals with highest fitness are cloned and the mutation process is done. The clone, mutation and selection process is executed repeatedly. Since it is a static approach the full knowledge of service time and access rate of the node should be known.

Wang et al. (2013), describes that data plays an important role in data intensive services; hence the cost and access time of the data greatly influence the quality of the service. There were many challenges involved in data intensive services; it includes the cost of transferring the data, the dynamic nature of the cloud, data replication and storage cost. In order to handle these challenges a data replica selection based on Ant Colony Algorithm is used. In this the ant selects the data replica such that the response time and the cost of transferring the data from server to endpoint are minimized. There are various pricing models like usage pricing model, package based, flat free subscription based and combination based pricing models. The service providers can switch among these pricing models to charge the users. The cost is calculated based on price of the replica, network bandwidth between the endpoint and the replica site and the size of the data. The data replica is selected by data replica (DR) agent. Once the DR agent is invoked the ants are placed at the service endpoint and moves from the endpoint to the each replica site. Each replica has an initial pheromone density based on which the ant chooses the replica. The ants make a local updating of pheromone after finding a set of data replica. The local updating is done using the pheromone intensity of the visited replicas. After all ants returns to their service end point a global updating of pheromone are done. This helps in selecting the appropriate replica which reduces the response time and transfer cost.

3 Proposed architecture and methodology

The primary objectives of the proposed D2R-IWD architecture are as follows:

(a) To identify the problems exist in the process of dynamic data replication in cloud storage.
(b) To design a novel dynamic data replication strategy to improve the access efficiency of cloud storage preserving the QoS attributes of cloud such as high availability, performance and load balancing etc.
(c) To propose an optimization algorithm to effectively manage the replicas and thereby improve the performance of cloud storage.

The major concerns of replication are the number of replicas to be created, location of the new replica, selection of data for replication, time to trigger the replication process, the management of replicated data and the provision of optimal replica to the user. Hence, our aim is to design a novel dynamic data replication strategy to improve the access efficiency of cloud storage preserving the QoS attributes of cloud such as high availability, performance and load balancing. The D2R replicates

the data dynamically based upon the number of access to the files and placement of replica considering the migration cost. The management of replicated data and the provision of optimal data to the user are done with the use of intelligent water drop (IWD) Algorithm. Also, it periodically checks for improving the capacity of the storage by removing unwanted replicas with balanced load on the storage.

The data service system of cloud contains a scheduling broker, cloud broker, data center broker and data centers. The scheduling broker is responsible for interaction with the end uses. Users submit the task to the scheduling broker through the user interface and are loaded in the task queue. All tasks that need to be served are processed in FCFS fashion. The cloud broker decides which data center should be selected to provide the services for the tasks. Each and every task stored in the queue is forwarded to the appropriate data center. The data center contains a replica catalog, in which all the files that are stored in the corresponding data center are indexed with its replicas. A replica log is placed which will keep track of all the activities with respect to accessing the replica, and the users that access the replica with its timestamp. Within each data center is a data center broker, which in turn contains information about the data that is stored at any point in time. The information about the data such as its location where the data is stored, number of access to the data and size of the data. The data center broker is involved in the process of replication and the management of replicated data. The proposed D2R-IWD in generic cloud architecture is represented in Fig. 1.

Firstly, we need to monitor the storage considering the set of files, their replicas, and the location in the datacenter. The storage is created with a specific name, bandwidth and capacity. The files are divided into fixed sized chunks and distributed to storage. The storage is monitored by calculating the bandwidth utilized, total number of access to the storage and the available storage space. Secondly, the dynamic data replication process is carried out calculating the replication factor based on the popularity degree of data. Also, a dynamic threshold is calculated based on the replication factor of the entire system and the adjustable system performance parameter. This helps in deciding when to initiate the replication process in the dynamic environment. When the replication factor is less than the dynamic threshold, it will trigger the replica manager to initiate the replication process automatically. Finally, the optimization and management of replicas to improve the access efficiency of cloud storage and the storage itself is done using a swarm based intelligent water drop algorithm.

3.1 Dynamic data replication

The data is replicated considering the dynamic changes in the cloud environment. The data that has more popularity is found using the number of access request to the particular data and that data will be chosen for replication. Figure 2 represents the dynamic data replication strategy is adopted taking list of files and the storages as input. The entire workflow of the replication process is described stepwise as follows.

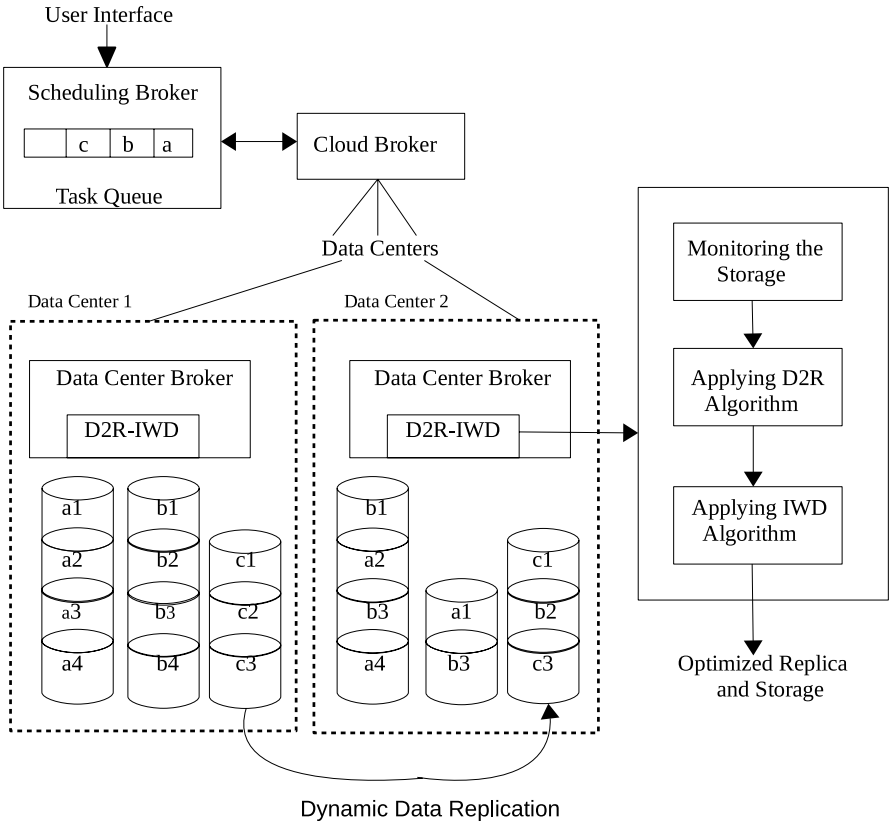

Fig. 1 D2R-IWD in generic cloud architecture

Step 1: The popularity of the data specified by the user as popularity degree (PD_i) is calculated based on the number of access to the corresponding data.
Step 2: The replication factor (RF_i) is calculated based on the popularity degree, number of replicas already available and the size of the data file.
Step 3: Dynamic threshold value (TH) is determined from the replication factor of each data file.
Step 4: The constraint ($RF_i > TH$) is checked for each file and if it is satisfied, then it will trigger the process of replication. Otherwise, the process is terminated.
Step 5: The number of replicas for each file satisfying the condition in step 4 is calculated.
Step 6: The final step of the replication strategy is to identify the appropriate storage to place the replica for user access.

The popularity degree (PD_i) of file f_i is calculated using the number of access (an_i) and the time based forgetting factor w_i as in Eq. 1.

Algorithm 1: Dynamic Replication

Input: List of files, storage and task submitted to broker.$< filelist >, < storageList >, < cList >$

Output: Replication of popular files.

begin

  for *each* $file_i$ *in file list* $< filelist >$ **do**

    Calculate the number of access an_i

    Popularity Degree as in Equation (1)

    Get the number of replica already present.

    Replication Factor by using Equation (2)

  end

  Replication factor of system $RF_{sys} \leftarrow \frac{\sum_{i=1}^{n}(PD_i)}{\sum_{i=1}^{n}(RN_i \times FS_i)}$

  $TH \leftarrow min\left((1+\alpha) \times RF_{sys}, max\left(\forall_{k\in[1,2,..l]} RF_k\right)\right)$

  for *each* $file_i$ *in file list* $< filelist >$ **do**

    if $RF_i > TH$ **then**

      add file to new list $<$ *replica list* $>$.

    end

  end

  for *each* $file_j$ *in* $<$ *replica list* $>$ **do**

    calculate:

    $old\ available_j \leftarrow number\ of\ replica(initially\ one)$

    $n \leftarrow old\ available_j + \beta \times (1 + old\ avalable_j)$

    Replicate$(file_j, n)$

  end

end

Fig. 2 Algorithm 1—dynamic data replication strategy

$$PD_i = an_i \times w_i \tag{1}$$

Based on the popularity degree the replication factor (RF_i) for each file is calculated as in Eq. 2.

$$RF_i = \frac{PD_i}{RN_i \times FS_i} \tag{2}$$

where RN_i is the number of replica and FS_i is the size of the data file.

Time based forgetting function represents the loss of popularity of the data with the users as it becomes old over time and is derived based on the number of access to the data over a period of time.

The dynamic threshold (TH) value is calculated based on the replication factor of the entire system and the adjustable system performance parameter (α) as in Eq. 3.

$$TH = \min\left((1+\alpha) \times RF_{sys}, \max\left(\underset{k\in[1,2,\ldots,l]}{\forall} RF_k\right)\right), \alpha \in [0,1] \tag{3}$$

The replication process is triggered for the files whose replication factor is greater than the dynamic threshold ($RF_i > TH$). The number of replicas to be replicated is determined based on the old file availability. The suitable location for the replica placement is determined by the access information of the storage node to balance the load on the system.

3.2 Optimization of replicated data using IWD

IWD is a bio-inspired multi-objective Algorithm which uses the movement of water drops over the river bed. Here the intelligent water drop is used for two operations, (i) for selecting the optimal replica for the user request and (ii) for the removal of replica to manage the storage space. Figure 3 represents the data flow diagram of the IWD algorithm for the optimization of the replicas and that of the storage space.

Here we generate a graph which contains the adjacency list of files for the storage. The IWD will visit each file in the storage and provide the suitable one in order to complete the task. The underlying chunking mechanism in which the files are

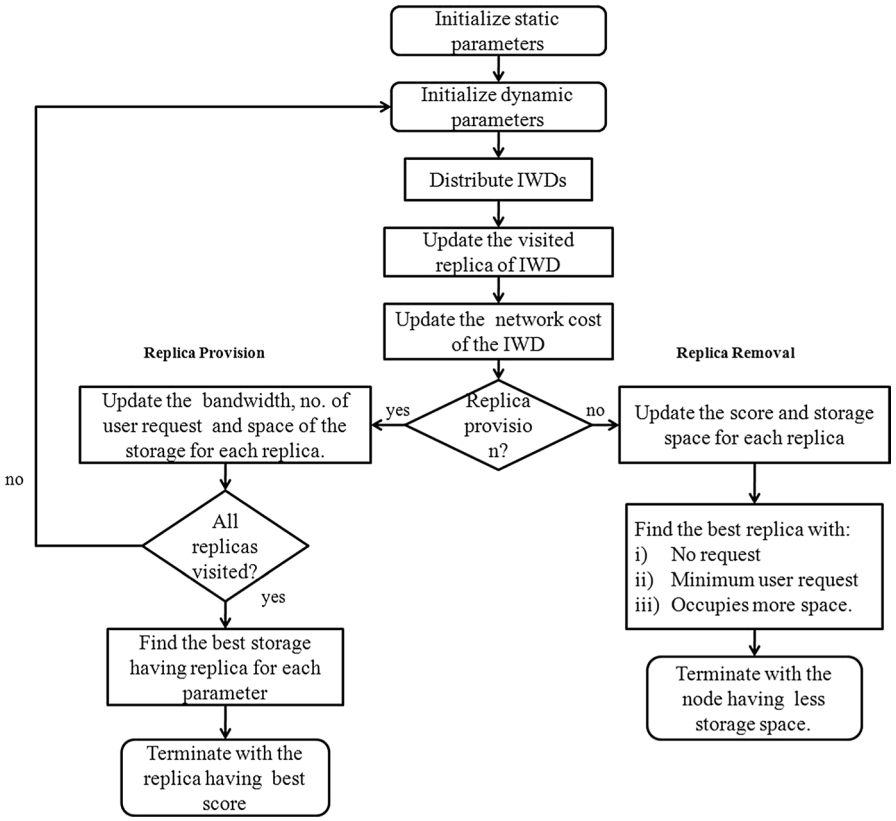

Fig. 3 Data flow diagram of IWD for replica provision and removal process

separated and stored can be of any type, either a fixed or variable size chunks. Each and every IWDs are given with the list of files, number of replicas and its location in the storage. Several factors are considered before making a decision of appropriate selection of replica. The steps in replica selection are described as follows.

3.2.1 Selection of replica

IWD is used for the selection of optimal replica according to the user request. The factors considered in the selection of optimal replica are the bandwidth, number of

Algorithm 2: IWD Select

Input: A single cloudlet *cl*, list of files $< File\ List >$, list of storage $< Storage\ list >$,adjacency list of file $< graphList >$

Output: Returns the optimal replica and the storage.

begin
- Get the file list $< cl\ file\ list >$ requested by *cl*
- **for** *each file i in* $< cl\ file\ list >$ **do**
  - **for** *each adjacency list from* $< graphList >$ *of storage* **do**
    - **for** *visit each file j in the list* **do**
      - **if** *file i* == *file j* **then**
        - add storage to new Map $Map < file, sub\ storage >$
      - **end**
    - **end**
  - **end**
- **end**
- **for** *each file k* $Map < file, sub\ storage >$ **do**
  - get the $< sub\ storage >$
  - **for** *each storage x from* $< sub\ storage >$ **do**
    - bw ← getUsedBandwidth()
    - access ← getNumberOfAccess()
    - space ← getAvailabelspace()
    - networkcost ← $\left(\frac{bw}{1}\right) + \left(\frac{access}{2}\right) + \left(\frac{space}{3}\right)$
    - add the storage and networkcost to $Map < storagex, networkcost >$
  - **end**
- **end**
- get the storage with $minimum\ (networkcost)\ \ from\ Map < storagex, networkcost >$
- return $< file, storage >$

end

Fig. 4 Selection of replica using IWD Algorithm

user request to a particular replica, total number of access to the storage and the storage space available. Figure 4 shows how IWD is used for selection of replica.

(a) *Initialize the static parameters* The static parameters such as the number of iterations, ($iter_{max}$), iteration count $iter_{count}$ initially 0, number of IWDs $N_{IWD} = N_c$ the number of storage nodes, network cost updating parameters (a_s, b_s, c_s), initial network cost for each storage node netcost(i) and initial bandwidth of storage have to initialized. The network cost includes the utilized bandwidth, the number of access to the storage and the available space.

(b) *Initialize the dynamic parameters* The dynamic parameters such as the visited list of replica ($V_c(IWD)$) which is initially empty.

(c) C*reation of IWD* Create the IWDs and distribute them to the nodes.

(d) *Selection of path* Update the visited list of IWD (V_c(IWD)). IWD chooses the storage that has the files requested by the user and returns those storages in each iteration. The IWD will use the graph list containing the adjacency list of files for each storage and returns the storage that contains the requested file.

(e) *Selection of total best* The IWD chooses the storage that has minimum network cost (i.e.) minimum of utilized bandwidth, number of access and maximum available space, from the set of storages that contains the files requested. The network cost is calculated as in Eq. 4.

$$netcost(i) = \frac{bw}{a_s} + \frac{access}{b_s} + \frac{space}{c_s} \tag{4}$$

where $netcost(i)$ is the network cost of the storage i, bw, $access$ and $space$ are the utilized bandwidth, number of access and available storage space of the storage i respectively and a_s, b_s, c_s are the network cost updating parameters. The network cost is updated for each request from the user. Also, the location of the user is taken into consideration before being applied with the algorithm. Here, the bandwidth is given higher precedence over the number of access and the available space. So, the network cost is directly proportional to the consumed bandwidth. However, in some cases if the priority is changed then those parameters are updated with higher values and so on.

(f) *The iteration best and total best* The storage containing the replica are found for each iteration and returned as iteration best. The optimal replica based on the storage is selected by the IWD as the total best as in Eqs. 5 and 6.

$$T^{IB} = storage \;\; containing \;\; requested \;\; file \tag{5}$$

$$T^{TB} = \begin{cases} \min(netcost(i)) \;\; if \;\; multiple \;\; storage \\ T^{IB} \;\; else \end{cases} \tag{6}$$

where T_{TB} is the total best solution and T_{IB} is the iteration best.

3.2.2 Removal of replica

If more number of replicas is created the storage space gets minimized. In order to efficiently utilize the available space the IWD is used. It helps in identifying the replica with less popularity or less number of access and it can be deleted. The user request can be redirected to other replica and the process is carried out periodically. The working mechanism of IWD for the removal of replica is shown in Fig. 5.

(g) The static, dynamic parameters are initialized and the IWDs are created and distributed. The visited list of IWD is also updated as in the replica selection process.
(h) IWD chooses the storage that has maximum network cost, since it is the storage with more bandwidth utilization, more number of access and less available space. The network cost is calculated giving priority to the storage space followed by the available bandwidth and then the number of access. This can also be altered depending upon the requirement and the objective of the problem.
(i) Based upon the result of the IWD, the storage is selected and each files in the storage is visited by the IWD with the help of graph list.
(j) The IWD returns the replica to be deleted considering the following conditions:

  (i) The replica with no access or less number of access.
  (ii) The replica occupying more space.

Algorithm 5: IWD Deletion

Input: List of files < *File List* >, list of storage < *Storage list* >,adjacency list of file < *graphList* >
Output: Deletion of the replica.
begin
  for *each storage i in* < *Storage list* > **do**
    bw ← getUsedBandwidth()
    access ← getNumberOfAccess()
    space ← getAvailabelspace()
    networkcost ← $\left(\frac{bw}{2}\right) + \left(\frac{access}{3}\right) + \left(\frac{space}{1}\right)$
    add the storage and networkcost to *Map* < *storage x*, *networkcost* >
  end
  get the storage with *max* (*networkcost*) from *Map* < *storage x*, *networkcost* >
  for *each file j in* < *graphList* > *of storage* **do**
    replica ← getNumberReplica()
    if *replica* > *2* **then**
      number of access ← getAccess()
      storage.deleteFile()
    end
  end
end

Fig. 5 Removal of replica using IWD Algorithm

(k) Here the iteration best is the storage with maximum network cost and the total best is the replica that is returned for the deletion. A minimum of two replica is maintained and the user request of the replica to be deleted is redirected to other replicas to balance the load of the cloud storage. The process of deletion is instantiated whenever there is less amount of storage, followed by more bandwidth utilization and then the priority moves to the total number of access on the storage. The proposed strategy automatically deletes the unwanted replicas in the whole storage and free the storage space thereby efficiently improves the capacity of the storage to accommodate more amount of data.

4 Experimental results and discussions

CloudSim has been used for simulating and evaluating the proposed system. CloudSim is a Java based simulator which provides many classes for modeling and simulating cloud systems. New classes can be easily developed or can be inherited from the already developed classes. It runs on machine with Intel XEON E 52630L-3.1 GHz processor with 16 GB RAM and 1 terabyte hard disk as a master node as in Hadoop distributed storage. We have also used several Intel Core i5A 380-3.1 GHz processor machines with 4 GB RAM each and 500 gigabytes hard disk for the slave nodes. Searching and retrieval of data in Hadoop is compared with our model with respect to access time (Lee et al. 2015). We have compared our environment with Hadoop, distributed cloud storage and tabulated the results. Also, we compared our method with several optimization techniques such as particle swarm optimization (PSO) and genetic algorithm (GA) (Lizhen et al. 2018). In doing so, we have allocated 6 storage spaces each of varying capacity [50, 100 GB]. Different types of files such as text, audio; image and video are divided into fixed size chunk and distributed among the storage. Initially there is one replica for each set of files up against the default replica size of 3 because it will increases redundancy and hence occupy three times the storage space. The environment was tested for 1000 different inputs requesting various files to complete the task.

For each file the number of access, number of replica already present, popularity degree and replication factor are calculated using the Eqs. 1 and 2. Figure 6

--------------------------CALCULATION OF THRESHOLD PARAMETER--------------------------

FileName	Number_of_Access	Number_of_chunks	No_of Replica	Popularity Degree	Replication factor
e	235	9	1	143.61325	0.015974777
a	255	2	1	155.83566	0.15218326
d	262	7	1	160.1135	0.023720518
j	228	1	1	139.33542	0.25379857
f	283	1	1	172.94702	0.22726284
i	260	1	1	158.89127	0.32493103
c	255	5	1	155.83566	0.034025256
g	248	3	1	151.55782	0.059551205
h	260	7	1	158.89127	0.023803934
b	268	3	1	163.78021	0.06720567

Fig. 6 Replication factor of each file

shows the number of access, number of replica present initially, the popularity of the file based on the number of access and the replication factor.

Based on the threshold calculated as in Eq. 3 the files to be replicated are chosen. Figure 7 shows the files to be created and the number of new replicas. After the replication of the files is done, the user request is satisfied by selecting the optimal replica with the help of IWD.

After the replication of the files is done, the user request is satisfied by selecting the optimal replica with the help of IWD. To do so an adjacency list is created for each and every file as shown in Fig. 8.

The vertices represent the fixed size chunk of each file. The path to reach a particular chunk is found out in the random graph generated for each storage. This helps to identify the replica and execute the IWD algorithm to properly

min of (RF of system (0.049236532), max of RF of all files (0.32493103))
Threshold: 0.049236532

CALCULATION OF NUMBER OF NEW REPLICA

file_name	file_size	old_file_availability	sum_of_rf	beta	number_of_new replicas
a	1024	1	1.0849326	0.014724212	1
j	549	1	1.0849326	0.14026979	1
f	761	1	1.0849326	0.021863587	1
i	489	1	1.0849326	0.23393027	1
g	2545	1	1.0849326	0.20947185	1
b	2437	1	1.0849326	0.2994942	2

The file ‘a’ with 2 chunks are replicated 1
The file ‘f’ with 1 chunk is replicated 1
The file ‘j’ with 1 chunk is replicated 1
The file ‘i’ with 1 chunk is replicated 1
The file ‘g’ with 3 chunks are replicated 1
The file ‘b’ with 3 chunks are replicated 2

Total no. of chunks to be added to storage is 14

Fig. 7 Determine the number of new replica

Random graph has 8 vertices
Adjacency List Representation of the random graph for Storage 1 is:

f1 → g2
b1 → g2 → d2 → d2 → g2 → a2
d4 → b2 → a2 → g2 → c4 → g2
a2 → c4 → b2 → c4 → d4 → c4 → b1
c4 → g2 → a2 → b2 → g2 → a2 → b2 → g2 → d4 → a2
b2 → c4 → d4 → g2 → a2 → d2 → c4 → d2 → g2
d2 → b1 → b2 → b1 → b2
g2 → c4 → c4 → b2 → b1 → d4 → c4 → b2 → f1 → d4 → b1

Fig. 8 Adjacency list representation of random graph for sample storage

Table 1 Sample storage list and network cost of each file

File name	No. of replica	Storage name	Utilized BW	No. of access	Available space	Network cost
b	3	Storage_0	7	1399	65,988	22,702
		Storage_2	7	1337	64,416	22,147
		Storage_5	11	863	67,191	22,839
g	2	Storage_0	7	1399	65,988	22,702
		Storage_2	7	1337	64,416	22,147

Table 2 Illustration to select optimal storage for deletion

File name	No. of access	Storage	Network cost	Optimal storage
b	65	Storage_0	66,457	Storage_2
		Storage_2	64,864	
		Storage_5	67,483	

manage the replicas in the storage. Table 1 shows the storage list containing the requested files. The optimal replica is chosen based on the network cost of the storage and number of access. Consider the user requests for the files [g, b]. The file g has 2 replicas and they are present in storage_0 and storage_2, among the two storages the user request is satisfied with the one from storage_2 since the network cost of storage_2 is less compare to that of storage_0. The network cost is calculated as in algorithm 2.

After the selection of optimal replica the management of the storage is done. The replica with minimum number of access is selected and is deleted from the storage with less available space using Algorithm 5. Table 2 shows the file to be deleted and the corresponding storage from which it is to be deleted.

Upon calculating the network cost of the files, our D2R-IWD algorithm selects the appropriate replica and initiates the deletion process and redirecting the existing users to access other replicas to balance the load on the storage. Therefore, in the above case file b is opted for deletion from storage_2 and updated in the replica catalog. The proposed IWD algorithm runs in the background of all the nodes and optimally removes the unnecessary replicas from the storage.

4.1 Test case 1: comparison of replication strategy: D2R-IWD versus Hadoop distributed storage, PSO and GA

The performance of D2R-IWD is compared with the traditional replication strategy adopted in Hadoop, a distributed environment for provisioning the requested file to the user. We have also compared our results with popular optimization techniques such as PSO and GA. Here, we refer to storage and retrieval of files across Hadoop as the task. The number of tasks is varied from 50 to 1000 and the

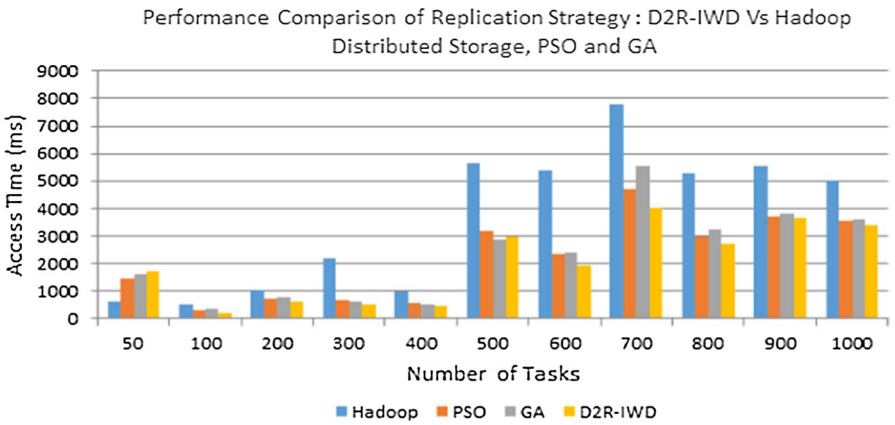

Fig. 9 Comparison of Replication strategy w.r.t. storage and optimization techniques

comparison is made in terms of access time (Bai et al. 2016). Figure 9 shows the performance comparison of replication strategies of D2R-IWD with other techniques. As the number of tasks to access the same file increases the popularity of the file increases. D2R-IWD works well with peak loads, but we have to note that there is less improvement during minimum load conditions on different file access. The access time of the storage decreases as the number of tasks increases as per the proposed algorithm. In initial time periods, the proposed D2R-IWD algorithm has little bit higher access time than the traditional replication strategy. For 50 tasks the access time is around 1750 ms.

When the number of task is increased to 100, the access time steeply decreases and reached to less than 500 ms for more number of tasks. The access time is drastically reduced irrespective of the number of tasks and provides a better result with respect to access time. Hence, the access time is inversely proportional to the number of user requests initially but there is a tremendous increase in performance of the replication process for large number of file access with 16% increase than PSO, 20% increase than GA and almost double than that of Hadoop. From Fig. 9, it is inferred that our D2R-IWD provides minimum access time for increased number of tasks and has a very low convergence rate than PSO and GA.

4.2 Test case 2: multiple users requesting single file

Here the response time in accessing the file from storage was tested. The number of users is increased, in which multiple users requesting the same file. There is a tremendous increase in the load in accessing the file, which in turn place the burden on the servers in which the files are replicated. Also, there is a possibility of higher response time in accessing the file there by affects the performance of the cloud. Figure 10 shows the comparison of the proposed D2R-IWD algorithm with Hadoop with respect to storage, and PSO, GA with respect to algorithm efficiency. As the number of users increases the response time in accessing the file

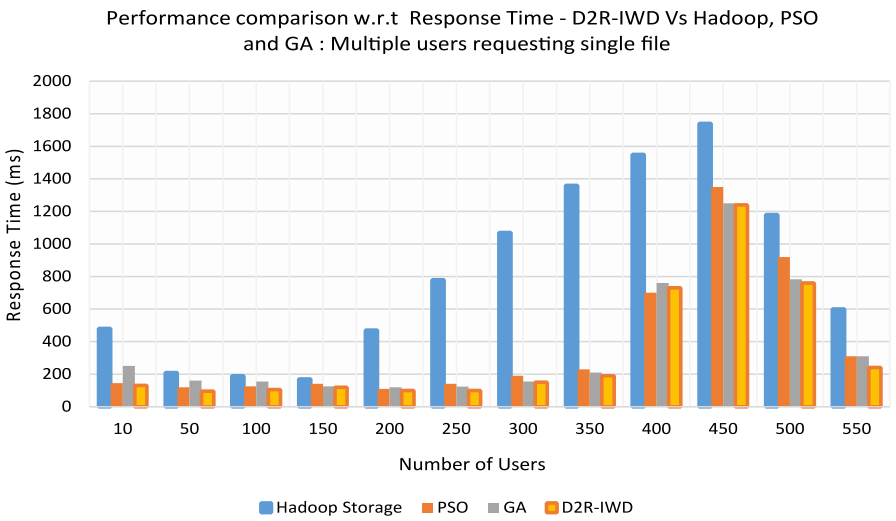

Fig. 10 Response time w.r.t. multiple users requesting single file

from the storage keeps relatively decreased when compared with the traditional cloud storage. There is a huge improvement in the performance of the system using our algorithm for replication. To illustrate the above stated improved performance, say for instance, for every increase in 50 users for accessing the same file the algorithm behaves close to the existing distributed storage model till it reaches 150 users. As it reaches 200 users and thereon, for every increase in the number of users there is a proportional decrease in response time till it reaches the maximum number of users of 550. However, at around 400 users in accessing the single file from storage, there is an increase in response time for both the methods.

Even though at some point in time both behaves more or less similar, our methodology proves to be better than PSO and GA in minimizing the response time in accessing the file from storage irrespective of the increased number of users. It is observed that there is 22% increase in response time as a whole than PSO, 27% increases than GA and almost double than that of Hadoop.

4.3 Test case 3: multiple users requesting different files with increased number of users

A total of six storage space was created with each of capacity 70 GB in the master and slave nodes. Each and every storage contains different chunks of various files with different size. Initially the chunks are stored randomly across the storage and we compared our algorithm on replication with Hadoop distributed storage, PSO and GA. In D2R-IWD, after the decision of number of replicas was made, the files are stored based on the network cost. Whereas in traditional methodology the number of replica is fixed and the storage is selected randomly. The graph in Fig. 11 shows the comparison with respect to the access time plotted against the

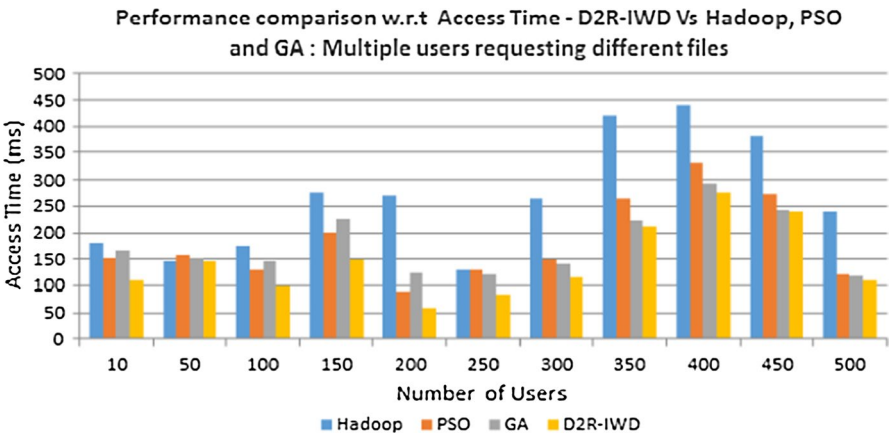

Fig. 11 Access time w.r.t. increased number of users requesting different files

increased number of users requesting for different file access. Here the number of users is increased in terms of 50 users and the access time varies according to the user request for different files of varying size from the storage.

As we can see from the graph, there is steady state that is maintained when the number of users is increased to 50 for both the cases and from there on, the access time tremendously decreases for increase in users accessing different files from storage. The average access time in adopting a cloud storage without optimization algorithm is 342 ms for increasing the user access whereas; the average access time on applying our D2R-IWD Algorithm is 147 ms. Hence, irrespective of whatever the chunking mechanism is used, our algorithm provides better result in reducing the access time of the data with 18% increase than PSO, 21% increase than GA and almost double than that of Hadoop. However, both these algorithms work better in align with our proposed approach and takes longer time to converge closer to our method.

4.4 Test case 4: performance comparison: optimizing the cloud storage space

We also turned our perspective over how to reduce the storage space with the same set of files and optimize the storage space with same or better performance to the end users. The proposed algorithm keep on executing in the background of the storage and check for the usage of files, their properties and decide on maintaining the replicas and optimize the storage.

In general, once the replicas are defined, it is not going to change unless the administrator intervenes. Our D2R-IWD algorithm runs continuously in the background and the updated list of files that are frequently accessed or not, are traced and maintained in a list and are repeatedly updated for a specific time frame. Three sets of lists on accessing the files were maintained: frequently accessed, less frequently accessed and not accessed for a long time. It helps in reducing the number of replicas and optimally frees the storage space.

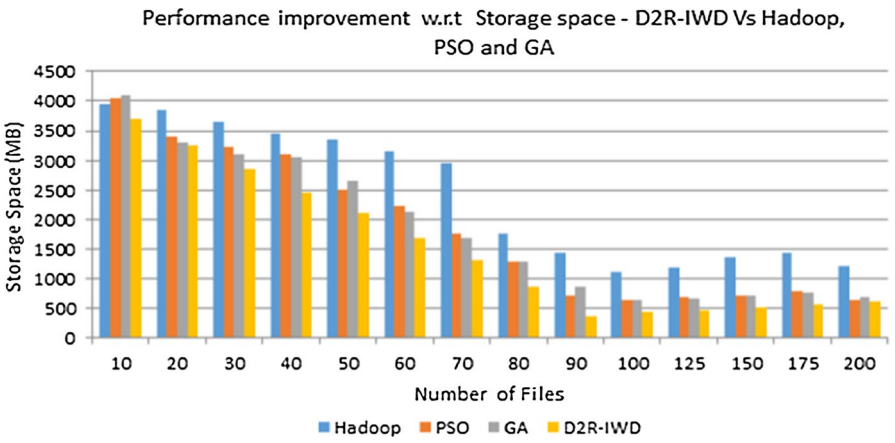

Fig. 12 Consumed storage space w.r.t. number of files

Figure 12 shows the comparison of D2R-IWD and the distributed storage in terms of available free space. Also, we compared our algorithm with standard optimization techniques such as PSO and GA. Here, the number of files to be stored is increased in relative to size over time and the storage space is compared. The total space consumed in cloud storage without our proposed strategy is averaged to 3000 MB for increased number of files in the storage whereas, the average storage space on applying our D2R-IWD Algorithm is 1650 MB. Almost there is 40% improvement in the free space of storage. However, PSO and GA works moderately in managing the storage space and takes longer time to reach an optimum position unlike the proposed approach. Therefore, the graph clearly shows that D2R-IWD manages the storage space in an efficient manner neither than Hadoop in terms of storage where the later almost doubled. Furthermore, in terms of optimization algorithm it is perceived that there is 24% increase in free storage space for PSO and 28% increase for GA.

5 Conclusion and future work

In cloud computing environment failures are normal rather than exceptional. Hence data availability, fault tolerance and access efficiency plays a major role in the performance of cloud. However data redundancy affects the storage space predominantly. The existing replication strategy arbitrarily fixes the number of replicas for each and every file. Hence, in this paper, a novel dynamic data replication strategy with intelligent water drop algorithm is presented, which considers parameters like bandwidth, number of user access, available storage space and traffic. The two main contributions of this paper includes the dynamic data replication strategy, replicates the data based upon the number of access or the popularity of the data. Secondly, the intelligent water drop algorithm helps in the selection of optimal replica based upon the bandwidth, traffic and number of user request. It is also used for the management

of cloud storage by selecting the replica for removal considering the number of access and the storage space occupied by the replica. We have tested our algorithm with storage similar to Hadoop and analyzed the access time for storage and retrieval of data. It is observed that there is almost 40% increase in the free space of storage. We have also compared our approach with standard optimization algorithms like PSO and GA on replication strategy and found out that there is 16% increase than PSO, 20% increase than GA. Furthermore, a detailed comparison was made in accessing the same data with variable loads and observed that there is 22% increase in response time as a whole than PSO and 27% increases than GA. For accessing different data, there is a tremendous improvement in access time with 18% increase than PSO, 21% increase than GA. We have tested it with four different test cases to improve access efficiency and obtained promising results preserving the high availability, performance and load balancing of the cloud storage. For future works, the energy factors of the storage node can be considered. Several other QoS attributes can be optimized with respect to cloud storage. Grouping of files with respect to storage can be explored to minimize the access time. Also, optimizing the query to retrieve the data from storage remains a challenging issue. Several other prediction mechanisms can be applied and verified to increase the accuracy of the selection process and the proposed model can be tested and incorporated in the real cloud environment.

References

Baesens B (2014) Analytics in a big data world: the essential guide to data science and its applications. Wiley

Bai X, Jin H, Liao X, Shi X, Shao Z (2016) RTRM: response time based replica management strategy for cloud storage system. In: Proceedings of the grid and pervasive computing. Springer, pp 124–33

Beaver D, Kumar S, Li HC, Sobel J, Vajgel P et al (2010) Finding a needle in haystack: Facebook's photo storage. In: OSDI, vol 10. pp 1–8

Gantz J, Reinsel D (2012) The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC analyze the future, vol 2007. pp 1–16

Gill NK, Singh S (2016) A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers. Future Gener Comput Syst 65:10–32

Grace RK, Manimegalai R (2014) Dynamic replica placement and selection strategies in data grids a comprehensive survey. J Parallel Distrib Comput 74(2):2099–2108

Jeon M, Lim KH, Ahn H, Lee BD (2012) Dynamic data replication scheme in the cloud computing environment. In: 2012 Second symposium on network cloud computing and applications (NCCA). IEEE, pp 40–47

Lee J, Chung J, Lee D (2015) Efficient data replication scheme based on hadoop distributed file system. Int J Softw Eng Appl 9(12):177–186

Li W, Yang Y, Yuan D (2011) A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. In: 2011 IEEE ninth international conference on dependable, autonomic and secure computing (DASC). IEEE, pp 496–502

Lin J-W, Chen C-H, Chang JM (2013) QoS-aware data replication for data intensive applications in cloud computing systems. IEEE Trans Cloud Comput 1(1):101–115

Lizhen C, Junhua Z, Lingxi Y, Yuliang S, Hui L, Dong Y (2018) A genetic algorithm based data replica placement strategy for scientific applications in clouds. IEEE Trans Serv Comput 11(4):727–739

Long SQ, Zhao YL, Chen W (2014) Morm: a multi-objective optimized replication management strategy for cloud storage cluster. J Syst Archit 60(2):234–244

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

Milani BA, Navimipour NJ (2016) A comprehensive review of the data replication techniques in the cloud environments: major trends and future directions. J Netw Comput Appl 64:229–238

Nachiappan R, Javadi B, Calheiros RN, Matawie KM (2017) Cloud storage reliability for Big Data applications: a state of the art survey. J Netw Comput Appl 97:35–47

Niu S, Ong S, Nee AY (2013) An improved intelligent water drops algorithm for solving multi-objective job shop scheduling. Eng Appl Artif Intell 26(10):2431–2442

Qu Y, Xiong N (2012) RFH: a resilient fault-tolerant and high-efficient replication algorithm for distributed cloud storage. Presented at the 2012 41st international conference on parallel processing (ICPP)

Shah-Hosseini H (2009) Optimization with the nature-inspired intelligent water drops algorithm. In: Santos WPD (ed) Evolutionary computation, Vienna, Austria, pp 297–320

Sun DW, Chang GR, Gao S, Jin LZ, Wang XW (2012) Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. J Comput Sci Technol 27(2):256–272

Wang L, Luo J, Shen J, Dong F (2013) Cost and time aware ant colony algorithm for data replica in alpha magnetic spectrometer experiment. In: 2013 IEEE international congress on big data (BigData congress). IEEE, pp 247–254

Wang P, Dean DJ, Gu X (2015) Understanding real world data corruptions in cloud systems. In: IEEE international conference on cloud engineering (IC2E), pp 116–125

Wei Q, Veeravalli B, Gong B, Zeng L, Feng D (2010) CDRM: a cost-effective dynamic replication management scheme for cloud storage cluster. In: Proceedings of the 2010 IEEE international conference on cluster computing, Heraklion, Crete, Greece, Sept. 20–24, 2010, pp 188–196

Wu TY, Pan JS, Lin CF (2014) Improving accessing efficiency of cloud storage using de-duplication and feedback schemes. IEEE Syst J 8(1):208–218