Received: November 18, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Popularity (Hit Rate) Based Replica Creation for Enhancing the Availability in Cloud Storage

S. Annal Ezhil Selvi[1*] **R. Anbuselvi[1]**

[1]*Department of Computer Science, Bishop Heber College, Trichy, Tamilnadu, 620017, India*
* Corresponding author's Email: ezhilabel.bhc@gamil.com

Abstract: In cloud computing, the replication management system has been well adopted in cloud storage applications. To provide the availability and reliability, the replication system replicates the files and can be stored in different server. The system led some complicated issues such as high memory consumption, incurred high storage cost and to access the file is more complicated issues in recent cloud storage applications. In the existing technique, File Accessing Frequency based Ranking (FAFR) Algorithm and Dynamically Reduced Replica for Rarely Accessed files (DRRRA) algorithm work jointly and identify the rarely accessed files and retain the replica in two server other replicated files are deleted. To provide access to more request with 2 or 3-replica is a complicated issue. Thus, this paper proposes a Dynamic replica Creation for Availability enhanced Storage (DRCAES) algorithm which jointly work with FAFR algorithm to predict most frequently accessed files and automatically replicated to other server based on server memory. The aim of this proposed approach is to enhance the availability, thereby reducing the request-response delay time. Thus the proposed approach optimizes the number of replicas, occupied space, and cost.

Keywords: Cloud storage, Replication, Reduce replica, Dynamic replica, File popularity, File accessing frequency.

1. Introduction

Microsoft Azure, Amazon, Google Cloud Storage (GCS) and as leading Cloud service Providers offer different types of storage (i.e., sequences of files, etc.) with different prices for data storage services. The data storage services and accessing files are very difficult issues on Redundancy Storage. Each cloud service provider also provides and monitors the commands to retrieve, store and delete data through network services, which impose in- and out-network delay and cost on an application [1]. In leading Cloud service provider in network cost is free, while out-network cost (network cost for accessing) is charged and may be different for usage of cloud providers. In cloud server Data transferring or data replication among from one server to other server, this shows significant price differences among them. The existing problem on this diversification plays an essential role in the optimization of data management request response and delay in cloud environments [2]. This proposed techniques at optimizing this request response and delay that consists of residential cost (i.e., storage) and potential migration cost (i.e., network server cost).

Cloud service provider many applications are moving towards a distributed interconnected network environments. In this distributed environment, the data storage and all computational cloud resources are distributed during different and widespread locations based on ranking.

A cloud server store the data, the data can have a huge number of users that require having access to huge data volumes. For example, consider a set of documents or images or videos that needs to be read and accessed by a number of user spread worldwide, in a distributed way. The access to vast data volumes by huge number of users can be access very time consuming. As the size of the system is increased, the tasks of providing such data service

becomes more difficult since its users suffer from long delays in data access.

Replica cloud storage is a new research field; the dynamic replication policy is still rarely seen. In this paper, the focal point is on early distributed storage file system replica creation approach, combine with the quality of cloud storage, this paper design a dynamic replication strategy based on high prediction. It create a replica according to the file access history quality, so with the purpose of users can access the nearby which bring the cloud system to one of the most excellent status with ranking, specifically the replica number of minimum, access to the highest efficiency, network lifetime is increased.

A fixed number of replicas for every file is insufficient to give quick file read for hot files while waste resources for storing replicas of cold files. Random selection of replica destination require observance all Data Centers active to ensure data availability, which though waste power consumption. As the random selection of replica destination does not think purpose of bandwidth and request handling capacity, network congestions could occur due to capacity restriction of some links and server may turn into overloaded by data requests.

In this environment, data replication is essential so that the users can retrieve the most request data from storage residing in nearby server. The replication is based on server memory [3]. The performance of the distributed networks is crucially affected by the replication strategy used. The huge majority of the known replication strategy determines the replicas by computing an easy ranking based on the number of requests for each file on individual cloud server. The "most accessing" files, the ones with the highest ranking value, are selected for replication due to the memory based other server replicated.

This ranking technique that it is quite possible that, the accessing files with high recent demand will be requested on cloud server, with even higher hit rates. Since the computation is quite simple, the strategy mainly focuses on the problem of select the most suitable files for storing the replicas based on memory [4].

The two main drawbacks of the strategies proposed are related to the following techniques implemented in this work. First one is optimization process and second one is high prediction ranking algorithm. The scheme presented in the literature does not take into account the change that might suggest itself in the interest of users for certain files. Instead, they are mostly involved in one or more factors that decide the importance of the file themselves, like the file size, the number of requests for an entity file or the contents of a file. The user request response time and files are analysed optimization process [5].

The Second process user request analysed and which files are most hitting on individual server. The existing replication decision algorithm satisfies the better request response time but the replication of hitting is complicated [5]. The hitting ratio is calculated or monitor on high prediction ranking algorithm. This algorithm identifies the most hitting files and replicated to other server based on server memory. The replication process implemented on this server automatically reduced the delay and request response time on server. At the end of this approach the cloud storage system will act as a Role-Based intelligent System (RBIS). So that, the user can an efficient data storage on cloud computing environment.

Some of the roles of the DRCAES approach is listed below,

- Dynamically predict rarely accessed files and most frequently accesses the file using FAFR model.
- Through DRRRA dynamically reduce the number of replicas of that rarely accessed files, so that, the cost and occupied space will be minimized. For reduction of the replica, it finds minimum available Space of DC among DC' where that file exists (Removal).
- In DRCAES, dynamically create and place the new replica for the most accessed file if the frequency of each replica is equally accessed otherwise it won't replicate. The new replica placed in the data center that has more available space and that file does not exist.
- Balanced storage retained during removal and new replica placement. Because, it analyses all aspects of Storage system like available Space, SLA, Accessing frequency of all existing replica.

The existing work are discussed in Section 2, overall back ground process are described in section 3, In section 4 the proposed work which is improve the overall request time and reduced delay using high prediction ranking algorithm are described. The section 5 discusses the results details. And section 6 is conclusion.

2. Related works

Data replication issue effectively requires taking a closer look at the arrangement of most common services and applications deployed on storage clouds to provide services to other parties. Such applications are usually implement as multi-tier

applications running on distributed software system. The multi-tier application user giving the request the overall response time is very high. The storage strategies so far consider the number of requests as the main hitting files for computing the popularity of each file [6].

The hitting files identify the limitation of the current research of data replication in cloud server: they are whichever hypothetical investigation without realistic consideration, or heuristics-based execution without a provable performance guarantee. The most directly related work to this replication work is complicated process on clouds server. The data replication and request response on cloud server as a static optimization problem on user access [7]. They show that this problem is NP-hard and request delay, which means that present, is no polynomial algorithm that provides an accurate solution. They only reflect on static data replication for the intention of proper analysis. The limitation of the static approach is that the replication cannot regulate to the dynamically shifting user access prototype. Additionally, their centralized process of integer programming technique cannot be simply implementing in a distributed cloud server.

The request response and resource sharing use an auction protocol to make the replication choice and to trigger long-term optimization by with file access patterns. In this propose utility-based replication strategies on clouds server. In this process address the data replication for availability in the face of unreliable works, this is different from this optimization work [8, 9].

The random collection of replica destination neglects server heterogeneity (i.e., different Data Centers vary in data request handling capacities and network capacities). The write due to creating replicas in production clusters at searching engine application for almost half of all cross-rack traffic. While the network within clusters is frequently underutilized, there exist some traffic jam links resulting from the network usage imbalance [10].

To assume the multi-facility cloud resource allocation problem, they are mainly involved in solutions that are agreeable to parallel implementations. There is quite a lot of reason. First, for a cloud resource allocation, problem (1) is inherently an important convex resource optimization problem, with millions of variables or still more. A centralized process of cloud server resource allocation solver is extremely inefficient in solving such large-scale cloud storage problems [11].

3. Background process

Cloud services, such as search engines, education portal, parallel application, social networking, etc., are often deploy on a geographically spread the infrastructure, i.e. data centers placed in different regions and better and reliability. A usual query is then how to direct the workload from users along with the set of geo distributed data centers in organize to achieve a desired transaction between performance and delay, since the power price exhibit an important degree of geographical diversity [12]. This query has involved much attention recently and is usually referred to as geographical cloud server load balancing.

The resource scheduling based data replication problem and focus on scheduling pathetically parallel resource usage which are collected of a set of independent responsibilities with very minimal or no data synchronization. A huge number of applications fit in to this type of resource sharing on cloud storage. Examples consist of distributed relational database query, search engine query, BLAST searches, data processing, and image processing applications such as shaft tracing. To effect apathetically parallel resource allocation, each of its tasks is placed on a physical server and executed in an external server added for that task. The completion time of this resource request is the completion time of the last finished request and overall request response, i.e., the make criticize of that set of request are completed [13].

The conventional data caching/replication problem have been considered extensively in the framework of the Web distributed cloud databases and multimedia systems. What be different from Web caching is that disk memory and I/O bandwidth are the main concerns in multimedia storage systems. A number of algorithms are proposed to attain high acceptance rate and resource utilization by balancing the use of different request response resources. Unlike Web search engine and multimedia data, database contents are access by both read and write operations based in optimization process and ranking [14].

It is assumed that the frequent accessed files in the past will be accessed more than the others in the future. This is called as high prediction ranking on temporal locality. With the property of sequential locality, a most accepted data file is resolute by analysing the number of access to the data files from users. After discovery the best popular file, we trace to the client that produce the most requests for the popular data file and a new replica is placed in it. Therefore, in this application have to collect history

of records regarding the end-to-end data transfers to decide which file should be replicated [15].

4. Proposed framework and algorithm

Dynamic replication of data is another significant issue since access frequencies to individual data items are likely to modify in most cloud server environments. The aim is to make the replication strategy rapidly and accurately adapt to changes and achieve optimal ranking process on long-term performance.

In [12], they establish that, to take advantage of cached data, it is sometimes essential to procedure individual queries using "suboptimal" plans in arrange to reach high system performance. In data replication is triggered as a result of changes of request rates in cloud server.

4.1 Dynamic replica creation availability enhanced storage framework

The proposed contribution is used an optimization based high prediction ranking algorithm. The Ranking algorithm (FAFR) is giving better performance on request response time and delay. The 'N' number of user agreed request and access the files on cloud server. To find the files and request user identification process and queuing process all are calculated the cloud server. Fig. 1 is architecture of optimization based ranking with conclude of input request response. The user requests the input and cloud analyse the user which files are request. The server files viewed by ranking (based on most popularity). Example the java.doc files mostly requested and download the files from user means, the java.doc is replicated to other server based on server memory.

Design data sharing-aware optimization algorithms for solving the resource request problem. Before relating the algorithms establish few static definition and assumption concerning the cloud servers. The Data Centers manage by the cloud service provider are in one of the following two states: active (available) and replicate.

An active cloud server is a server that is powered on and is currently considered for all resource allocation by the algorithms. A replication is a server that is most frequent files are request in cloud server considered for replicated data on other server based on memory allocation by the algorithms. So it can denote by Fi, Si the set of most frequent access file and number of Data Centers. When the entire cloud Data Centers hosted by files, and accessing most frequent files are replicated files on ranking based techniques in clouds server.

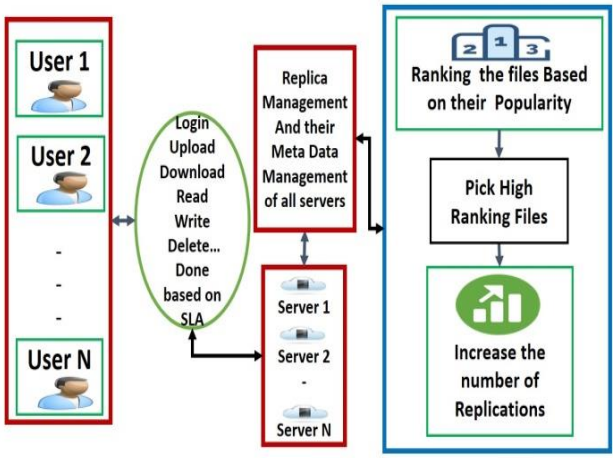

Figure.1 Proposed architecture

Data Centers are configured and located in different geo-locations. Files are stored in that configured Data Centers. The previous work of this research proposed an algorithm (Dynamically Reduced Replica for Rarely Accessed file (DRRRA) Algorithm [15]) reduce the replica based on file popularity (Least accessed files) which is the result of Ranking Algorithm (File Accessing Frequency based Ranking) [14]. The ranking algorithm predicts the files based on their popularity. But, this paper focuses the most frequently accessed file which is the most popular file. If the user need is increased for a file that files replicated dynamically.

4.2 Mathematical model for replica creation and placement

The following notation and equations are used in the prediction process of frequently accessed files and replica increasing process.

- m: Number of uploaded files.
- n: Number of Data Center (DC).
- DC_i : i^{th} DC, i ← 1,…,n.
- F_j : j^{th} file, j ← 1,…,m.
- $\theta_{i,j}$: Replica Accessing Frequency (RAF) (Hit Rate) of i^{th} file in j^{th} DC. It is shown in the following matrix (m X n) representation.
- $DC_1 \quad DC_2 \quad \dots \quad DC_n$
- $$\begin{matrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{matrix} \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{i,n} \\ \theta_{2,1} & \theta_{2,2} & \cdots & \theta_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ \theta_{m,1} & \theta_{m,2} & \dots & \theta_{m,n} \end{bmatrix}$$
- FAF: File Accessing Frequency computed using eq. (1) as,
- $\mathbf{FAF}_j = \sum_{i=1}^{m}(\theta_{i,j})$, i ← 1,…, n Λ j ← 1,…,m. (1)

- $TFL_{i,j}$: Total File list(TFL) in i^{th} file in j^{th} DC, where i ← 1,…,n Λ j ← 1,…,m.
- $AFL_{i,j}$: L where is Allocated File List (AFL) of i^{th} file in j^{th} DC, if $\theta_{i,j} > 0$, $\in TFL_{i,j}$
- $NAFL_{i,j}$: Non Allocated File List of i^{th} file in j^{th} DC, if $\theta_{i,j} = 0$, $\in TFL_{i,j}$
- $TFLDC_{i,j}$:K where is Total File List of each Data Center' of i^{th} DC in j^{th} File, where i ← 1,…,n Λ j ← 1,…,m.
- $AFLDC_{i,j}$: D where is Allotted File List of each Data Center, of i^{th} DC in j^{th} File, if $\theta_{i,j} \neq 0 \in K$.
- DCC_j : Data Center' Capacity (DCC) of j^{th} DC, where j ← 1,…, n .
- FS_j: File's Sizes of j^{th} DC, where j ← 1,…, m .
- OS_j: Occupied Space of j^{th} DC is calculated using the following equation
  $OS_j = \sum_{j=1}^{n} FS_j$ $\in D$, j ← 1,…,n. (2)
- AS_j : Available Space of j^{th} DC calculated using eq. (3) as,
  $AS_j = DCC_j - OS_j$ j ← 1,…,n. (3)
- FA_Th: Frequently Accessed files Threshold Value.
- Th_LL: Low Limit of threshold value.
- Th_UL: Upper Limit of threshold value

In prediction process, there are two levels of prediction done to find the files which are really needed to increase the number of replicas for meeting the availability enhancement requirements. In first level prediction, the most frequently accessed files are predicted using Eq. (4) based on the FAF of the FAFR model.

The second level prediction able to done based on the result of the first level prediction. By first level prediction, some of the most frequently accessed file is in resulting set. From that files, the second level prediction finds the files which are really required to provide seamless availability that is done using Eq. (5) based on RAF of the FAFR model.

$$F_i = FAF \geq \text{FA\_Th} \quad (4)$$

$$\theta_{i,j} = \begin{cases} Max\left((AS_j) \in NAFL_{i,j}\right) = 1 & \text{if th\_LL} \leq \theta_{i,j} \leq \text{th\_UL}, \\ & \forall\ \theta_{i,j} \in F_i \\ 0 & Otherwise \end{cases} \quad (5)$$

To be exact, Check the frequency of each replica (if it is equally accessed it will be replicated otherwise it won't be replicate). Then the Replica Placement will be done based on available space of the data center. That is, the dynamically created replica will be placed in Datacenter that has more available space and that file does not exist.

4.3 File accessing frequency based ranking (FAFR) algorithm

FAFR is a ranking algorithm which proposed in [14] itself. But in this paper, the new name is given with some refined work. θ is the Replica Accessing Frequency (RAF) (hit rate). Initially, θ value is 0 when the file is uploaded then the θ value became 1. And whenever the file is accessed the θ value will be incremented. Finally, the summative θ value is calculated and considered as a File Accessing Frequency.

Input: DC_i, F_i, $M(F_i)$, k=0, θ =0,q
Output: Rank { q's result set}

1. Done ∀ file in ∀ Data center's when user interface triggered
2. If file upload
3. θ =1
4. If file access
5. θ = θ +1;
6. for each F_i do
7. for each DC_i do
8. K=k+ θ
9. End
10. Rank. insert (all Values)
11. If Rank =q
12. Return Rank

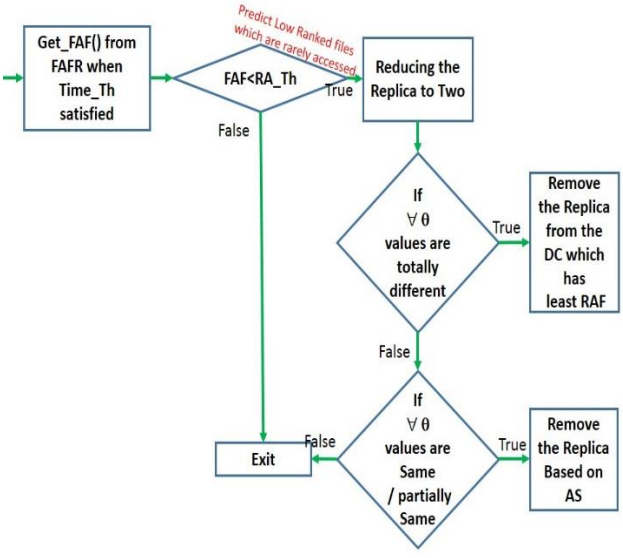

Figure.2 Working principle of DRRRA

In Fig. 2 is shown the working principle of Dynamically Reduced Replica for Rarely Accessed files (DRRRA) is described in [15]. FAFR and DRRRA are jointly worked and predict Rarely Accessed files then reduce their replica to 2-replica strategy. Here the minimum replica is 2 for assuring Reliability and maximum is 3-replica strategy.

4.4 Dynamic replica creation for availability enhanced storage (DRCAES) algorithm

The following DRCAES algorithm dynamically finds the most frequently accessed file using FAFR [14]. Then, based on FAFR result DRCAES dynamically replicate the replica and place it on DC that have maximum Available Space (AS) and that DC does not have that file's replica on it.

Input : F_i, DC_i , NAL_k, $\theta_{i,j}$, FA_Th, Th_LL , Th_UL., DC, Replica
Output: Dynamically Increased Replica.

1. Set values to FA_Th, Th_LL and Th_UL.
2. If user access
3. FAF← getFAF()
4. If FAF>= FA_Th then
5. F_i ← getHighRankFiles()
6. DC_i ← getHighRankFilesDC()
7. NAL_k ← get_allNonAllocatedList()
8. For each F_i do
9. For each DC_j do
10. $\theta_{i,j}$ ← getRAF(F_i , D_j)
11. If(θ i, j >= Th_LL and $\theta_{i,j}$ < Th_UL) then
12. Replica=copyof(F_i)
13. DC=getDCid(max(AS_i ←getAvailableSpace(NAL_k)))
14. REPLICATE(DC, Replica)
15. End
16. End

In DRCAES algorithm Step (3) is used to get most frequently accessed files. For every user access monitored and when FAF reach the FA_Th valued that time it performs the second level prediction which is done in RAF. That is, the every RAF should satisfy the range which mentions in step 10. End of this checking, if the result set has files that files are required to increase their number of replications, that will be done based on SLA specification and Available Space of DC which is defined in Step (12).

Step (11) is used to verify if the file need to replicate or not. That is, Th_LL is lower limit of threshold value range and Th_UL is a upper limit of threshold value range which determine the ranges of threshold values are decide to replicate or not.Then, Step(12-14) are work when step (11)'s decision.

Example:

The worked out examples of DRCAES algorithm is presented in tables 1 to 6, which will be discussed below one by one. After this discussion, the reader can easily understand the concept of our approach clearly.

The Prediction of Frequently Accessed Files based on FAF process is represented in table 1. The FA_Th value for validation is 15. When the FAF reach 15 that file comes under the consideration. For an example, using Eq. (4) the files 1, 3, and 5 are in consideration.

In Table 1 shows seven Data Centers are configured with 5GB Memory and they are located in different geo-locations. And five different files are stored on among the 7 DC.

Next, these files are verified by second level prediction process that is Prediction of highly needed files based on RAF which is explained in table 2. In second level prediction done using Eq. (5) which checks the individual replica frequency, if it all equally accessed that file will be replicated in one more data center.

In Eq. (5), there are two threshold values involved. First one is Th_LL, its value for validation is 10. The second one is Th_UL, its value for validation is 20. The file 3's replicas residing in DC2, DC4, and DC5, as well as their RAF, is 3 is 8,10,6 respectively. So this files not in the range of Th_UL and Th_UL. Thus, this file need not be replicated, but other two files 1 and 5 need to be replicated because it satisfies the range values.

In table 3 depicted the process of replica placement which is described in Eq. (7). The file 1's replicas are residing in DC2, DC4 and DC7 along with their RAF are 14, 20 and 15 respectively.

The replica placement has done based on Available Space (AS) of the data center which doesn't have the replica of the file. Here, the DC1, DC3, DC5, and DC6 doesn't have the replica of file1 as well as their AS is 4.5, 4.9, 4.7 and 4.8 respectively.

The DC3 has the maximum of AS among the DCs which as mentioned in the previous point. So, the replica will be placed in DC3, the reflected changes in metadata are shown in table 4.

Table 1. Prediction of frequently accessed files based on FAF

S. No	File Name	File Type	NR	File Size in MB	RAF (Replica Accessing Frequency) (θ)							FAF
					DC 1 5GB	DC 2 5GB	DC 3 5GB	DC 4 5GB	DC 5 5GB	DC 6 5GB	DC 7 5GB	
1	Array_Java	docx	3	0.07	0	14	0	20	0	0	15	49
2	Tree_ds	pdf	2	0.11	0	0	1	0	0	1	0	2
3	Img_001	jpeg	3	0.31	0	8	0	10	6	0	0	24
4	CS_C	mp3	2	0.55	3	0	0	2	0	0	0	5
5	HelloEnglish	mp4	2	1	0	13	0	0	0	0	19	32
OS (Occupied Space In GB)					0.5	1.4	0.1	0.9	0.3	0.1	1.0	
AS (Available Space in GB)					4.5	3.6	4.9	4.1	4.7	4.9	4.0	

Table 2. Prediction of highly needed files based on RAF

S.No	File Name	File Type	NR	File Size in MB	RAF (Replica Accessing Frequency) (θ)							FAF
					DC 1 5GB	DC 2 5GB	DC 3 5GB	DC 4 5GB	DC 5 5GB	DC 6 5GB	DC 7 5GB	
1	Array_Java	docx	3	0.07	0	**14**	0	**20**	0	0	**15**	**49**
2	Tree_ds	pdf	2	0.11	0	0	1	0	0	1	0	2
3	Img_001	jpeg	3	0.31	0	**8**	0	**10**	**6**	0	0	**24**
4	CS_C	mp3	2	0.55	3	0	0	2	0	0	0	5
5	HelloEnglish	mp4	**2**	1	0	**13**	0	0	0	0	**19**	32
OS (Occupied Space In GB)					0.5	1.4	0.1	0.9	0.3	0.1	1.0	
AS (Available Space in GB)					4.5	3.6	4.9	4.1	4.7	4.9	4.0	

Table 3. Ex. 1: RAF based replica creation (Before placement)

S.No	File Name	File Type	NR	File Size in MB	RAF (Replica Accessing Frequency) (θ)							FAF
					DC 1 5GB	DC 2 5GB	DC 3 5GB	DC 4 5GB	DC 5 5GB	DC 6 5GB	DC 7 5GB	
1	Array_Java	docx	3	0.07	0	14	0	20	0	0	15	49
2	Tree_ds	pdf	2	0.11	0	0	1	0	0	1	0	2
3	Img_001	jpeg	3	0.31	0	8	0	10	6	0	0	24
4	CS_C	mp3	2	0.55	3	0	0	2	0	0	0	5
5	HelloEnglish	mp4	2	1	0	13	0	0	0	0	19	32
OS (Occupied Space In GB)					0.5	1.4	0.1	0.9	0.3	0.1	1.0	
AS (Available Space in GB)					**4.5**	3.6	**4.9**	4.1	**4.7**	**4.8**	4.0	

Table 4. Ex. 1: RAF based replica creation (After placement)

S.No	File Name	File Type	NR	File Size in MB	RAF (Replica Accessing Frequency) (θ)							FAF
					DC 1 5GB	DC 2 5GB	DC 3 5GB	DC 4 5GB	DC 5 5GB	DC 6 5GB	DC 7 5GB	
1	Array_Java	docx	4	0.07	0	14	1	21	0	0	15	49 +1
2	Tree_ds	pdf	2	0.11	0	0	1	0	0	1	0	2
3	Img_001	jpeg	3	0.31	0	8	0	10	6	0	0	24
4	CS_C	mp3	2	0.55	3	0	0	2	0	0	0	5
5	HelloEnglish	mp4	2	1	0	13	0	0	0	0	19	32
OS (Occupied Space In GB)					0.5	1.4	0.4	0.9	0.3	0.1	1.0	
AS (Available Space in GB)					**4.5**	3.6	**4.6**	4.1	4.7	**4.9**	4.0	

Table 5. Ex. 2: RAF based replica creation (Before placement)

S. No	File Name	File Type	NR	File Size in MB	RAF (Replica Accessing Frequency) (θ)							FAF
					DC 1 5GB	DC 2 5GB	DC 3 5GB	DC 4 5GB	DC 5 5GB	DC 6 5GB	DC 7 5GB	
1	**Array_Java**	**docx**	**4**	0.07	**0**	**14**	**1**	**20**	**0**	**0**	**15**	**49 +1**
2	Tree_ds	pdf	2	0.11	0	0	1	0	0	1	0	2
3	Img_001	jpeg	3	0.31	0	8	0	10	6	0	0	24
4	CS_C	mp3	2	0.55	3	0	0	2	0	0	0	5
5	**HelloEnglish**	**mp4**	**2**	1	**0**	**13**	**0**	**0**	**0**	**0**	**19**	**32**
OS (Occupied Space In GB)					0.5	1.4	0.4	0.9	0.3	0.1	1.0	
AS (Available Space in GB)					**4.5**	3.6	**4.6**	**4.1**	**4.7**	**4.9**	4.0	

Table 6. Ex. 2: RAF based replica creation (After placement)

S. No	File Name	File Type	NR	File Size in MB	RAF (Replica Accessing Frequency) (θ)							FAF
					DC 1 5GB	DC 2 5GB	DC 3 5GB	DC 4 5GB	DC 5 5GB	DC 6 5GB	DC 7 5GB	
1	**Array_Java**	**docx**	**4**	0.07	**0**	**14**	**1**	**20**	**0**	**0**	**15**	**49 +1**
2	Tree_ds	pdf	2	0.11	0	0	1	0	0	1	0	2
3	Img_001	jpeg	3	0.31	0	8	0	10	6	0	0	24
4	CS_C	mp3	2	0.55	3	0	0	2	0	0	0	5
5	**HelloEnglish**	**mp4**	**2**	1	**0**	**13**	**0**	**0**	**0**	**0**	**19**	**32**
OS (Occupied Space In GB)					0.5	1.4	0.4	0.9	0.3	0.1	1.0	
AS (Available Space in GB)					**4.5**	3.6	**4.6**	**4.1**	**4.7**	**4.9**	4.0	

Table 7: Comparison of Existing PRC [3], Proposed DRRRA [15] and DRCAES

S.no	FAF	Existing (PRC) conventional 3 Replica Strategy (Minimum Replica 1 Maximum Replica 3)				Proposed DRRRA Algorithm (Dynamically Reduced Replica of Rarely Accessed files (2-Replica)				Proposed DRCAES Algorithm (Dynamically create the Replica based on User need)			
		NR	OS= FS * NR	Cost= OS * 0.00067	RR_TD	NR	OS= FS * NR	Cost= OS * 0.00067	RR_TD	NR	OS= FS * NR	Cost= OS * 0.00067	RR_TD
1	>10	**3**	**2.31**	**0.0015477**	**3**	**2**	**1.54**	**0.0010318**	**3**	**2**	**1.54**	**0.0010318**	**3**
2	15-40	**3**	**2.31**	**0.0015477**	**3.5**	**3**	**2.31**	**0.0015477**	**3.5**	**3**	**2.31**	**0.0015477**	**3.5**
3	41-60	**3**	**2.31**	**0.0015477**	**4.2**	**3**	**2.31**	**0.0015477**	**4.2**	**4**	**3.08**	**0.0020636**	**3.7**
4	61-80	**3**	**2.31**	**0.0015477**	**7**	**3**	**2.31**	**0.0015477**	**7**	**5**	**3.85**	**0.0025795**	**3.9**

In table 5 is shown the replica placement process of file 5. The file 5's replicas are residing in DC2 and DC7 along with their RAF is 13, and 19 respectively. Here, the highlight point regarding this file is, the replica of this file is reduced by DRRRA approach (2-replica strategy) which is discussed in chapter 6. But, the need for this file is increased. So, it is going to be replicated.

Here, the DC1, DC3, DC4, DC5, and DC6 doesn't have the replica of file 5 as well as their AS is 4.5, 4.6, 4.1, 4.7 and 4.9 respectively. The DC6 has the maximum of AS. So, the replication will be stored in DC6 which is shown in table 6. It is shown in Table (5) is after ranking the process replicated to that files based on server memory.

5. Result and discussion

The reflections in the parameter which involved in research due to DRCAES approach is presented in table 7. There are 4 parameters and their value calculation shown in this table such as NR (number of Replicas), OS (occupied Space), Cost and RR_TD (Request-Response Time Delay).

Finally, when the research comparing the proposed algorithms with the existing algorithm the DRCAES give better performance in all aspects such as the number of replicas, Occupied Space (OS), Cost and Request-Response Time Delay (RR_TD) based on File Accessing Frequency Finally, when the research comparing the proposed algorithms with the existing algorithm the DRCAES give better performance in all aspects such as the number of replicas, Occupied Space (OS), Cost and Request-Response Time Delay (RR_TD) based on File Accessing Frequency (FAF) for file of File Size 0.77 GB. It is shown in the table 7.

The change in NR value will be reflected to OS, Cost, and RR_TD parameter values. In the existing system, the NR value decided based on disk failure rate benchmark of NR is 3 which is the convention strategy.

For an example, the above table represents the file with File Size (FS) 0.77 in GB is uploaded and accessed in different scenarios.

- In the proposed system, the NR is decided based on DRRRA and DRCAES approaches.
- The Occupied Space (OS) is calculated using following way,
  Occupied Space (OS) = File Size (FS) * Number of Replicas (NR)
- The cost is calculated in the following way,
  Cost = Occupied Space (OS) * 0.00067
  (0.00067 is the amount incurred for 1 GB per day which is adopts based on Google Drive Cost plan. This is only for testing purpose)
- The RR_TD values are obtained by the use of MATLAB tool.

These values are calculated based on different File Accessing Frequency (FAF). The table values are presented in graphical representations.

Fig. 3 shown the comparison of changes in Number of Replicas parameter in different File Accessing Frequency (FAF). From the graph, we can clearly understand the NR is standard in existing PRC, either 2 or 3 in DRRRA approach, and it is vary based on FAF in DRCAES approach.

Fig. 4 presents the comparison of Occupied Space (OS) and different File Accessing Frequency (FAF) range. The graph is boons for the clear understanding the reflections done due to the NR. The OS is also standard in existing PRC because of Standard NR, minimized for rarely accessed files in

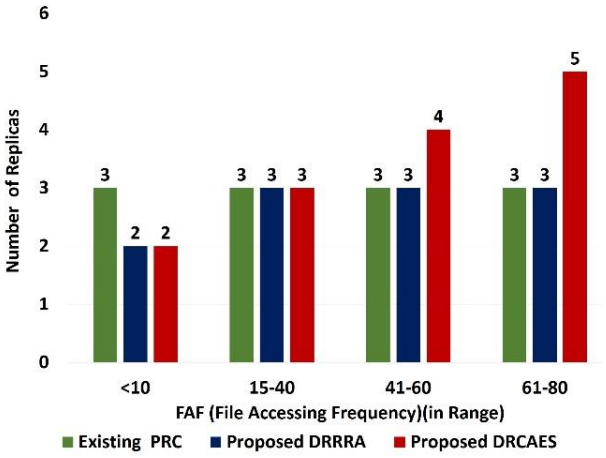

Figure.3 FAF vs. No. of replica

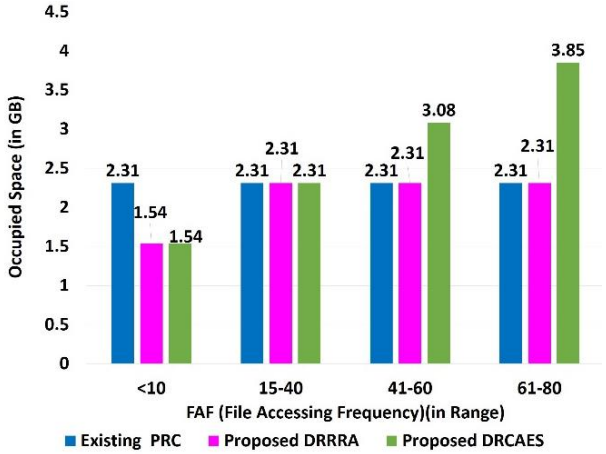

Figure.4 FAF vs. Occupied space

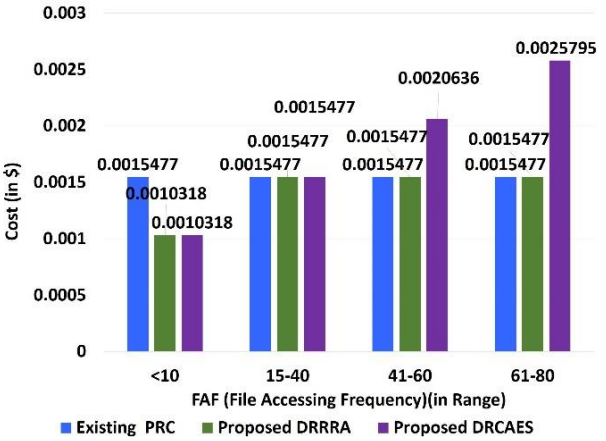

Figure.5 FAF vs. Cost

DRRRA approach, but optimized based on FAF in DRCAES approach.

Fig. 5, the comparison of Cost in dollars ($) and different File Accessing Frequency (FAF) range is presented. The graph is determines the changes in cost due to the Occupied Space (OS) modification. The cost is also a regular in existing PRC because of Standard OS, minimized for rarely accessed files in

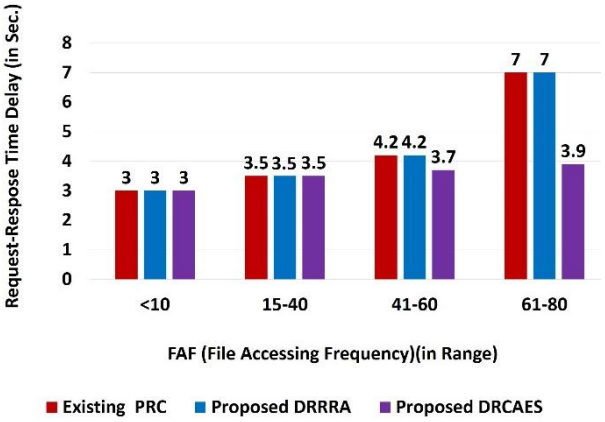

Figure.6 FAF vs. Request-response rime delay.

DRRRA approach, but in DRCAES approach, it is optimized based on FAF.

Fig. 6 shows the Request-Response Time Delay (RR_TD) in the sec. and different File Accessing Frequency (FAF) range. The graph values represented based on MATLAB tool result and the reflection of the RR_TD values is determined by the value of NR. The RR_TD is worst in existing PRC and DRRRA approach for high FAF range because of Standard NR, but it is reduced in DRCAES approach, because of optimized NR based on FAF.

Similarly, the table 7 presents the comparison of parameters such as Number of Replicas (NR), Occupied Space (OS), Cost, Request-Response Time Delay (RR_TD) along with reliability and availability concern of the proposed DRCAES with DRRRA and exiting PRC algorithm. The optimized cost obtained without affecting the existing reliability assured percentage.

From the table, we can understand the DRCAES provides an efficient data storage on cloud computing environment with reliability, availability concerns in a cost-effective manner.

The benefits of DRCAES approach is listed below which stated in section 1. That all are proved,

- Dynamically predict rarely accessed files and most frequently accesses the file using FAFR model.
- Through DRRRA dynamically reduce the number of replicas of that rarely accessed files, so that, the cost and occupied space is minimized. For reduction of the replica, it finds minimum available Space of DC among DC' where that file exists (Removal).
- In DRCAES, dynamically create and place the new replica for the most accessed file if the frequency of each replica is equally accessed otherwise it won't replicate. The new replica placed in the data center that has

Table 8. Comparison of parameters for existing and proposed algorithms

	Number Replica	**Occupied Space**	**Cost**	**Reliability**	**Availability**	**Request-Response Time Delay**
Existing PRC [3]	1 or 2 or 3 Decide Based on Disk Failure Rate	Minimized Based on Replica	Minimized Based on Replica	1-Replica →No reliability 2-replica→95% Assured 3-replica→99%	Not Considered	Increased for more request
Proposed DRRRA	**2 or 3 decide Based on FAF**	**Minimized for Rarely Accessed File.**	**Minimized for Rarely Accessed File.**	**2-replica →95% Assured [3]**	Not Considered	Increased for more request
Proposed DRCAES	**2-Replica is Minimum and maximum is decided Based on SLA**	**optimized**	**optimized**	**2-replica→95% Assured [3]**	**Enhanced**	**Decreased**

more available space and that file does not exist.

- Balanced storage retained during removal and new replica placement. Because, it analyses all aspects of Storage system like available Space, SLA, Accessing frequency of all existing replica.

6. Conclusion

To minimize the request response time and delay of data placement for time-varying workload applications, user necessity optimally makes use of the time difference between storage and network services across multiple cloud service provider. The previous work of this research dynamically predicts rarely accessed files with a help of FAFR algorithm and reduces the number of replicas for that file, if it satisfies the time limit using DRRRA algorithm. Similarly, the proposed DRCAES algorithm dynamically predicts most frequently accessed files with the help of the FAFR algorithm. Then, it creates a new replica for that file and finds the data center that has more available space and doesn't have that file. However, this work achieves the optimizing occupied space, cost, server performance, increased server's service delivery speed and decreased request-response time delay. Thus, ultimately the proposed DRCAES provide an efficient data storage with an optimized cost without affecting reliability, availability concerns for the cloud also by optimizing the number of replicas based on the user need and SLA. The proposed algorithm achieves better result when compared to the existing algorithms. In future replica management during the disaster could be considered without affecting the reliability and availability concerns with minimum replica.

References

[1] R. Han, M.M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications", *Elsevier, Future generation power systems in Science Direct*, Vol.32, No.1, pp. 82-98, 2014.

[2] M. Du and F. Li, "ATOM: Efficient Tracking, Monitoring, and Orchestration of Cloud Resources", *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No.8 , pp. 2172-2189, 2017.

[3] W. Li, Y. Yang, and D. Yuan, "Ensuring Cloud Data Reliability with Minimum Replication by Proactive Replica Checking", *IEEE Transactions on Computers*, Vol. 65, No. 5, pp. 1494-1506, 2016.

[4] S. Souravlas, and A. Sifaleras, "Binary-Tree Based Estimation of File Requests for Efficient Data Replication", *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No. 7, pp. 1839-1852, 2017.

[5] R. Han, S. Huang, Z. Wang, and J. Zhan, "CLAP: Component-Level Approximate Processing for Low Tail Latency and High Result Accuracy in Cloud Online Services", *IEEE Transactions on*

Parallel & Distributed Systems, Vol. 28, No.8 , pp. 2190-2203, 2017.

[6] U. Tos, R. Mokadem, A. Hameurlain, T. Ayav, and S. Bora "A Performance and Profit Oriented Data Replication Strategy for Cloud Systems", In: *Proc. of IEEE Conferences on Ubiquitous Intelligence & Computing*, *Toulouse, France*, pp. 780-787, 2016.

[7] D.T. Nukarapu, B. Tang, L. Wang, and S. Lu, "Data Replication in Data Intensive Scientific Applications with Performance Guarantee", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 8, pp.1299-1306, 2011.

[8] A. Kumar, R. Tandon, and T.C. Clancy, "On the Latency and Energy Efficiency of Distributed Storage Systems", *IEEE Transactions on Cloud Computing*, Vol. 5, No 2, pp. 221- 233, 2017.

[9] M. Hadji, "Scalable and Cost-Efficient Algorithms for Reliable and Distributed Cloud Storage", *Springer International Publishing Switzerland*, Vol.581, No.1, pp. 3-12, 2016.

[10] S.Q. Long, Y.L. Zhao, and W. Chen, "MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster", *Journal of Systems Architecture*, Vol. 60, No.1, pp. 234-244, 2014.

[11] S. Rampersaud and D. Grosu, "Sharing-Aware Online Virtual Machine Packing in Heterogeneous Resource Clouds", *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No. 7, pp. 2046-2059, 2017.

[12] Y. Lin and H. Shen, "EAFR: An Energy-Efficient Adaptive File Replication System in Data-Intensive Clusters", *IEEE Transactions on Parallel and Distributed Systems* , Vol. 28, N, pp. 1017-1030, 2017.

[13] L. Shi, Z. Zhang, and T. Robertazzi, "Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud", *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No. 8, pp. 1607-1620, 2017.

[14] S.A.E. Selvi and R. Anbuselvi, "Ranking Algorithm Based on File's Accessing Frequency for Cloud Storage System", *International Journal of Advanced Research Trends in Engineering and Technology*, Vol. 4, No. 9, pp. 29-33,2017.

[15] S.A.E. Selvi and R. Anbuselvi, "Optimizing the Storage Space and Cost with Reliability Assurance by Replica Reduction on Cloud Storage System", *International Journal of Advanced Research in Computer Science*, Vol. 8, No.8, pp.327-332, 2017.