International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Efficient Placement Algorithm for Data Replication and To Improve System Availability in Cloud Environment

Sasikumar Karuppusamy[1*] **Madiajagan Muthaiyan[1]**

[1]*Birla Institute of Technology & Science, Pilani, Dubai Campus, UAE*
Corresponding author's Email: sasikumark1175@gmail.com

Abstract: The IT systems sent to satisfy their business functionalities and to provide good Quality of Service (QoS) parameters such as availability, scalability and performance. The failures are normal rather than exceptional in case of cloud computing. To improve system availability enhancing the system performance is an important factor. For that replicating the popular data to multiple sites is a best choice as the users can access the data from a nearby site. In this paper, we proposed a dynamic replication and placement algorithm to enhance the performance of the software system. We calculate popularity degree and replica factor to identify the suitable file to replicate. From this the most accessed file and the file need to be replicated is found and the replicate of the file is created. Then the proposed algorithm is used to place the replicas in the suitable location. We compare the performance of the technique with the existing technique.

Keywords: Data Replication; Cloud Computing; Popularity Degree; system availability.

1. Introduction

In our daily life, quick dispersal and access of information are becoming an essential part with the development of computer networks, especially the internet. Cloud computing is a technology that utilizes the internet and central remote servers to provide adaptable services for its users. The integration product between the traditional computing models such as distributed computing, virtualization and the developing network technology was termed as cloud computing. The computing, platform, storage and service resource pools can be realized, which are abstract, dynamic extending and manageable based on the above computing model. In addition, the resource pools can be given to external clients on demand via internet [1]. In cloud, the services are conveyed on demand to external users over high speed internet with 'XaaS' where X as a services is a computing architecture which is broken down in to three segments: "infrastructure", "platforms", and "applications". It gives clients with most adaptable services in a clear manner and with less expensive and most powerful processors [2]. The IT systems sent to satisfy their business functionalities and to provide good Quality of Service (QoS) parameters such as availability, scalability and performance. Performance of a software system is one of the most important QoS parameter used for client satisfaction. If for any application that fulfills all business requirements but fail to satisfy the performance quality, it leads to greater dissatisfaction of software application end users. After the introduction of cloud, clients are opting for cloud based infrastructure in order to provide infrastructure needs of high performing software applications and this is true even for applications based on data replication strategies.

Data replication is a technique of creating identical copies of data (files, databases, etc.) in geographically distributed sites. Each copy is called a replica [3].Data replication is a general and simple approach to achieve these goals. It has been widely used in many areas, such as the Internet, peer-to-peer systems, and distributed databases [4-8]. With reference to Replication strategies, replica optimization is one of the performance enhancement techniques for software system. One of the objectives of replica optimization is to lessen file access times by guiding access requests to corresponding replicas and pro-actively replicating repeatedly used files based on access statistics accumulated. A well defined data replication method should meet the requirements of being able to determine the appropriate time to replicate files, decide which file should be replicated and to store these replicas in exact location [9].To design efficient dynamic data replication scheme the critical steps have been the analyzing of data access pattern [10-12]. In cloud computing, replication is used to reduce user waiting time, increasing availability of data and minimizing cloud bandwidth consumption by offering the user multiple replicas of a specific service on different nodes [13]. In a replicated environment, copies of files or fragments are hosted in multiple system or sites. When the number of replicas is increasing it enhances the system performance by improving the locality; however it involves additional data transmission to keep replicas updated to achieve data consistency. Since this increases cost with excessive replication of fragments, there is a need to have a replication strategy which can control the number of replicas for each fragment.

In this paper, we proposed a dynamic data replication with placement algorithm strategy to enhance the performance of software system. Here we use dynamic data replication using popularity degree and replica factor and placement algorithm. Our proposed approach consists of three phases:1) Selecting Replica using PD, 2) Creating replica using RF and 3) Placing replica. In the first phase; we find the file to be replicated by calculating popularity degree (PD). In the second phase; we create replicas with the help of popularity degree and replica factor(RF) in the third phase; we place these replicas to the suitable location. Thus the simulation results shows that by using our proposed approach it provides users with a system that has higher data availabilities, lower data transmission delays, and less bandwidth consumption for data access.

Our main contributions are summarized as follows:

- We compute popularity degree to find the files needed to replicate.
- And we calculate replica factor to find the exact file to replicate and to create the replicas.
- Finally to place the replicas in a suitable location placement algorithm is used.

The remaining paper is organized as follows: In section 2 related works is discussed based on the data replication strategies in order to enhance the availability of data in the cloud. In section 3, the system definition is explained. In section 4 the proposed data replication and placement algorithm is evaluated. In section 5 is followed by experimental results.

2. Review of Related Works

In the literature survey, several methods have been proposed using data replication strategy to improve system availability. Among them the most recently published works are presented as follows:

In [15], W. K. Lin, C. Yes, D. M. Chiu have discussed about the P2P replication system. Here several heuristic algorithms are used to achieve good system level file availability. They showed the difference among the performance of different algorithm s is significant when the storage space is abundant. In [18], to avoid the drawbacks in the year 2008 they discussed about the dynamic data replication strategy called Latest Access Largest Weight to reduce job execution time. The popular file is found and replicated to suitable sites to achieve a system load balance. In[16],K. Sashi,A. S. Thanamani proposed a Modified BHR Region Based Replication Algorithm which increases the data availability by replicating files within the region and also reduces unnecessary replication. They are stored in particular site instead of storing in many sites. Thus mean job execution time can be minimized, the network is used more effectively and storage space is saved.

In [17], S. Kamali, P. Ghodsnia and K. Daudjee discussed about the new framework for dynamic fragment allocation to. By using this technique they find near optimal allocation scheme within the framework which provide efficient solution to the fragment allocation problem. In [22], N. Mansouri proposed a Modified Latest Access Largest Weight to overcome the drawbacks from the above mentioned paper in 2011. Here they compared with the eight existing algorithms and improve Mean Job Time and Effective Network under all of the access patterns, especially under the different random file access patterns. In [2], they discussed about the dynamic centralized data replication algorithm MinDmr. It minimizes the data access time and network load by creating and spreading replicas from the super data centers to main data centers.

In [19], Albert discussed about the data replication technique for cloud computing data centers which optimizes energy consumption, network bandwidth, and communication delays which can be applied in both the geographically distributed data centers as well as inside each individual data center. In [20], they discussed about the QoS aware data replication problem in cloud system. The data replication cost and the number of QoS-aware data replicas cannot be minimized. To solve this problem, high-QoS first-replication (HQFR) is used. In [21], Mohamed-K HUSSEIN, Mohamed-H MOUSA proposed an adaptive replication strategy in order to enhance the availability of new replica factor. But in this, to find the best replica it takes some time.

3. System Model

This section describes about the series of availability of the system, replicating data, file allocation to the desired system. In cloud computing, cloud storage is considered to be an important factor. Cloud Storage is a service where data is remotely maintained, managed, and backed up. The service is usually available to the users over the network. It allows the user to store files online so that the user can access them from any location via the internet. The providers make them accessible to the client online by keeping the uploaded files on an external server. Many of these services are free up to a specific number of gigabytes, with additional storage available for a monthly fee. Availability is said to be the operable or committable state of the operation. Availability should be increased in order to improve the system performance. Availability of a system is defined as

$$A = MTTF/ (MTTF + MTTR)$$

Where MTTF is the Mean Time too Failure of the business system and MTTR is Mean Time to Repair. The architecture for the cloud data service center is shown in fig1.

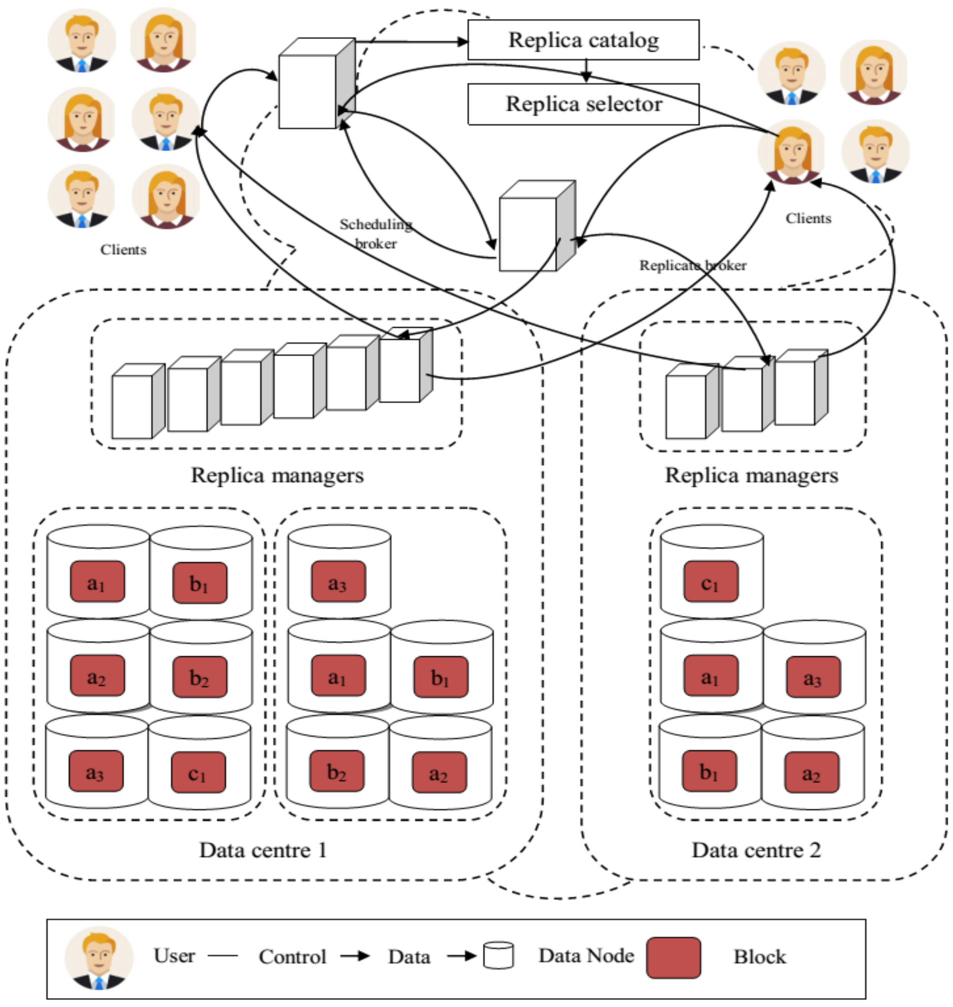

Figure.1 Cloud data center architecture

The cloud data service center consists of

- Replica Manager
- Replica Selection
- Replica Catalog
- Scheduling and replica Broker

❖ Replica Manager

The replica manager controls the overall operation of replication management system. It helps in creation and managing replicas based on the demand of the user. It increases the storage space and a directory to keep track of all the replicas and their locations. The directory information is configured by the replica manager. The log in replica manager is used to schedule the request and redirects the suitable replica copies. It holds the general information about the datacenter and replica location in the region. It provides replica services and manages the replica access, consistency, core replica creation, deletion and authentication. Replica manager supports the data management and the transfer of data between cloud and the creation of new replicas. It also tracks the user access pattern, monitors data popularity and also determines whether the local creation is required or not based on their availability. The manager and the catalog communicate with each other synchronously to maintain successful replication process.

❖ Replica Selection

Replica selection requires information about the capabilities and performance characteristics of a storage system. Replica selection is done using the replica selector. It is based on the demand of the user and failure occurs during access time. By selecting the reasonable replicas it can improve the response time of the service and also decreases the cost of the service. The caching function in replica selector is used to create corresponding service end point and the copy of the requested replica. The service end point has space to store the data set.

❖ Replica Catalog

Each newly created replica is registered in the replica catalog table. Replica catalog is also responsible for locating requested data and maintains the number of user bases, datacenter, replicas in region, number of request at certain time period and availability. When each time, the site stores a new replica it send a file register request to replica catalog and then replica catalog add this site to the list of sites and hold the replica. The catalog is then queried by applications to discover to discover the locations of available replicas of a particular replica location in each datacenter. A replica catalog contains information about locations of replica and associated replicas and the metadata associated with these replicated data. When a user request a file, replica broker has query the catalog using attributes to make operations such as locating the nearest replica of a particular database by using index. The data search and the location process starts at the local data catalog to check if the data stored and available locally. If it is not find, it redirect to other region.

❖ Scheduling and Replica Broker

The scheduling broker is the central managing broker. The scheduler reads the service list, and uses the service description file as input to deploy each new service. The replica broker is used to manage the activities of the replica files. The control will first move from the user to the scheduling broker where it schedules the user request and pass the control to the replica broker. Here it indicates the replica files to be created and sends it the replica manager to decide where to place these replicas.

4. Proposed Technique to Improve the Performance of Software System

In our proposed approach, dynamic data replication and placement algorithm is used to enhance the performance of the software system. The process of our proposed data replication and placement is shown in fig 2. To achieve dynamic data replication there are three important problems to be solved. 1) Which data should be replicated and when to replicatethe data wrongly selected and too early replicated data will not reduce the waiting time or speed up data access. 2) How many suitable new replicas should be created. With the number of new replicas increasing, the system maintenance cost will significantly increase.3) Where the new replicas should be placed to meet the system task successful execution rate and bandwidth consumption requirements. To enhance the performance, we use three phases:

4.1. Selecting replica using PD
4.2. Creating replica using RF
4.3. Placing replica

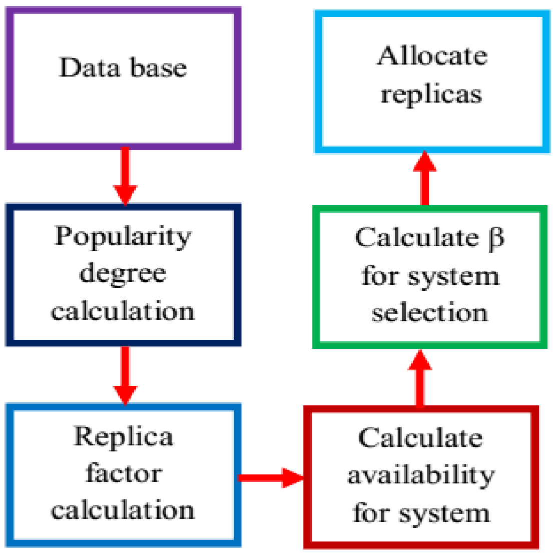
Data base
Allocate replicas
Popularity degree calculation
Calculate β for system selection
Replica factor calculation
Calculate availability for system

Figure.2Process of our proposed data replication and placement of replicas

We calculate the PD and RF to find the file to replicate and also to create the replicas. Then we use placement algorithm for placing the file in appropriate location. For this we calculate the availability factor to find whether the particular location is available for placing the replica.

4.1 Selecting Replica using PD

For selecting replica, a popular data file is identified by analyzing the access histories and setting different weights for different accessed data. Basically, the more recently accessed data is more pertinent to the analysis, and is thus set to a highest priority by giving high weight. We can find this by calculating popularity degree. The popularity can be calculated based on three factors and the formula to calculate PD is shown below:

$$PD = Q1 + Q2 + Q3 \quad (1)$$

Where, PDdenotes the popularity degree of a file, $Q1$denotes the first factor, $Q2$denotes the second factor and $Q3$denotes the third factor. The equation to calculate $Q1$ is given below.

$$Q1 = \sum_{t_i = t_s}^{t_p} FA_{(t_i, t_{i+1})} \log\left(\frac{nu}{nr}\right)_{(t_i, t_{i+1})} \quad (2)$$

In the above equation, FA denotes the Access Frequency betweent_i andt_{i+1} and nu denotes the number of unique users between t_i and t_{i+1} andnr denotes the number of repeated users between$t_i$and t_{i+1} and $t_s$is the Start time and $t_p$is the Present time. The first factor iscalculated by taking the product of access frequency and the inverse access frequency.The formula to calculate the second factor is given below.

$$Q2 = \sum_{t_i = t_s}^{t_p} FA_{(t_i, t_{i+1})} W_t \quad (3)$$

Where, W_t is Total weight of tables in a file. The second factor is calculated by taking the product value of the access frequency in a time interval and the total weight value of each table in a file. We then sum the values obtained on each interval to calculate the value for the second factor. In each file, there has number of tables and each table has weight values. The total weight value is adding the weight values of each table in a file. The third factor is the same that used to calculate the popularity degree is shown by an equation below.

$$Q3 = \sum_{t_i = t_s}^{t_p} an_k\left(t_i, t_{i+1}\right) \times \omega\left(t_i, t_p\right) \quad (4)$$

After calculating the first factor$Q1$, second factor$Q2$ and the third factor$Q3$ of a file, the popularity degreePD of a file can be calculated. Similarly, we have to find the popularity degree for each file.

4.2 Creating Replica using RF

The replica factor calculation is used to find whether the data file should be replicated or not and to create replicas. We calculate the replica factor by calculating the Positive Factor (PF) and negative factorNF. The purpose of Positive Factor is to identify how important is a file for replication. The formula to calculate the positive factorPF is given below:

$$PF = \frac{PD_{current} - PD_{\min}}{PD_{\max} - PD_{\min}} \quad (5)$$

In the above equation, PF represents Positive Factor and$PD_{current}$ represents the Popularity degree of the current file and PD_{min} represents Minimum popularity degree and PD_{max}represents Maximum popularity degree. The positive factor $PF$is calculated by identifying the difference between the popularity degree of the current file$PD_{current}$ and the minimum popularity degree value$PD_{min}$and dividing the solution by the difference between the maximum popularity degree value $PD_{max}$and the minimum popularity degree value PD_{min}. The positive factor is calculated to find the importance of the file to replicate. Thereafter, we have to evaluate the Negative Degree (ND) for each file. The negative degree calculation is used to find the Negative Factor (NF). NF of a file specifies if a file should not be replicated. The equation to evaluate negative degree *ND*for each file is given below:

$$ND = M \times R \times QRT \tag{6}$$

Where, *ND* denotes Negative degree and *M* denotes the Memory size of a file and *R* denotes the number of replica exist and *QRT* denotes Query response time. The negative degree *ND*is then used to calculate the negative factor*NF*of a file. The formula to calculate the negative factor*NF*is as follows:

$$NF = \frac{ND_{current} - ND_{\min}}{ND_{\max} - ND_{\min}} \tag{7}$$

In the above equation, *NF* represents Negative Factor and $ND_{current}$represents Negative Degree value of current file and ND_{min}represents Minimum negative degree value and ND_{max}represents Maximum negative degree value. The negative factor *NF* value is calculated by taking the difference amid the negative degree value of the current file$ND_{current}$ and the minimum negative degree value $ND_{min}$and dividing the solution with the difference amid the maximum negative degree ND_{max} and the minimum negative degree ND_{min}. The negative factor is calculated to find the importance of not to replicate the file. The replica factor is then calculated using the equation given below:

$$RF = \frac{\gamma PF + \alpha(1 - NF)}{\gamma + \alpha} \tag{8}$$

Here, *RF* denotes replica factor and *PF* denotes the positive factor of a file and *NF* denotes negative factor of a file andγ, α are the constant values which we assigned as 1 by checking performance with different values. The number of replica is then generated by the following condition,

$$no.\,of\ replica = \begin{cases} RF_t + 2 \times RF_{t-1}; & if\ RF_t > RF_{t-1} \\ 0 & ;\ if\ RF_t < RF_{t-1} \end{cases} \tag{9}$$

Where, RF_tdenotes Replica factor at a time interval*t*and RF_{t-1}denotes Replica factor at previous time interval*t-1*. It compares replica factor with the threshold value to decide whether to create replica for the particular file. Ifthe replica is greater than the threshold value then the replica will be created. These replicas will be considered when allocating the node.

4.3Placing replica

In this section, our aim is to ensure the desired availability with minimum replicas without degrading system performances. This goal is possible with a placement strategy that takes into account: the desired availability, the stability of nodes in the system and the failures. We need a location to place the replicas which we get from the above step. For that first we have to consider the list of nodes in the data center. Then these nodes are arranged in descending order using the classification criteria as explained below. The placement of several replicas of same data in the same node does not improve the availability or fault tolerance. For this reason, it will be useful to store a single replica of the same data in a node. In our system, we can predict failures in node. So in case of suspecting failure, the node places all its data in other node to assure the desired availability.

Consider some list of nodes and calculate the availability factor for these nodes.The availability factors can be calculated to find whether the particular node is available to place the data. The availability factor can be calculated as follows.

$$AF(Ni) = \frac{STAB(Ni)}{FR(Ni)} \tag{10}$$

If a node is with good AF , then it has good stability*STAB(Ni)*and low failure degree*FR*.The stability is in the range of 0≤ *STAB(Ni)*≤1.The failure degree can be calculated as

$$FR(Ni) = \sum_{j=1}^{k} Size(Dij) \tag{11}$$

Where *Ni*a node, *k* is is the number of data in the node *Ni* and *Dij*is the data *j* stored in node *Ni*.To increase the distance between the identical replicas, no similarity function $\overline{\beta}$. It can be defined as

$$\overline{\beta}(n) = ((LDn \cup DNn) - (LDn \cap DNn)) \tag{12}$$

Where '*n*' is a node, '*LDn*' is the list of local data in node n and '*DNn'* is the list of data of all neighbors of the node '*n*' in the data center. Here $\overline{\beta}(n)$ is the size of data difference between the data list of node n and the data list of its neighbors. For example if $\overline{\beta}(n) = 0$ then all the data in the node n also exist in the neighborhood. To avoid this non-deterministic case $\overline{\beta}(n) = 0$, we classify the node according to the following criteria.

$$classification\ criteria = AF\left(1/\left(1 + \overline{\beta}\right)\right) \tag{13}$$

After calculating classification criteria and arranging it in descending order. Check whether the replicated file can be placed in the node or not. If possible then store the replicated file to the corresponding node or else proceed to the next node and check if that node satisfies the condition. This process continues till all the replicas are placed. For each node it verifies in the list *N_List* if adding the replica increases the non-similarity of the node. If so, then it stores the data in the node else it tests the next node in the *N_List*. For example, consider if the *N*_List contains five nodes and we want to store a new replica of the data R and all nodes have the same AF. If *N_List*= {2, 5} and the replica R we created does not increases the no-similarity in node 2i.e. $\left(\left(\overline{\beta}'(2)\right)=\left(\overline{\beta}(2)\right)\right)$ because this data already exists in its neighborhood.

Table 1. placement algorithm

```
Create a list of nodes N_List
Sort the list in descending order using eqn (13)
If rep $\leftarrow$ false
While(N_List $\neq$ null & Rep=false)
    If $\overline{\beta}'(n) > \overline{\beta}(n)$ then
Store replica of the data in node n
$\overline{\beta}(n) \leftarrow \overline{\beta}'(n)$
  rep $\leftarrow$ true
    else go to the next node in N_List
    end if
  end while
if rep-=false then
  store the replica in the first node n in N_List
$\overline{\beta}(n) \leftarrow \overline{\beta}'(n)$
End if
```

But in addition to this data in node 5 increases its no-similarity. (i.e) $\left(\left(\overline{\beta}'(5)\right)>\left(\overline{\beta}(5)\right)\right)$, so the replica R can be placed in node 5. If it goes through a whole list *N_List* without replacing the replica. Then it chooses the best node in terms of *AF*. Thus the node minimizes the recovery time and keeps the distance between different data replication. The placement of replicas in the suitable node can be done using this following procedure. The placement algorithm is shown in table1.

5. Result and Discussion

In this section, our proposed method of dynamic data replication and proposed algorithm method discussed in the previous section is implemented.The evaluation of the results is discussed below.

5.1Experimental setup

Our proposed technique is implemented using Java (jdk 1.6) which is installed in a system that has following configuration: i5 processor with 3.20GHZ clock speed, 4GB RAM. Here, we are using the sample data set which is shown below in table 2.The data set consist of the ID, Sex, Birthday, Description, First date, Admission and Diagnosis.

Table 2. Sample database

ID	SEX	Birthday	Description	First Date	Admission	Diagnosis
2110	F	13-Feb-34	94.02.14	93.02.10	+	RA susp.
11408	F	02-May-37	96.12.01	73.01.01	+	PSS
12052	F	14-Apr-56	91.08.13	55.02.11	+	SLE
14872	F	21-Sep-53	97.08.13	12.04.09	+	MCTD
27654	F	25-Mar-36	92.02.03	04.08.16	+	RA, SLE susp
30609	F	13-Jul-44	91.08.13	29.06.12	-	SLE, MCTD
43003	M	24-Nov-37	94.03.08	15.02.15	-	Reynaud's phenomenon
48473	F	07-Oct-48	97.08.13	13.01.13	+	SLE
52199	F	16-Mar-54		14.09.14	-	PM/DM
57266	M	25-Jul-23	97.02.03	23.02.05	+	RA

5.2Experimental Results

In our proposed approach, to improve the performance of our system dynamic data replication and placement algorithm is used. For that the replicas are created and they are placed in an appropriate node using the placement algorithm. The query generated by the user is compared with the datasets and the generated query should be available to the user within a short period of time. We compare the performance of our proposed technique with the existing technique in terms of system byte effective rate (SBER) and execution time. The query generated for the process is shown in table 3. The query generated is searched in the database and the result related to the query should be available to the user within a short period of time. The Fig 3 and 4 explains about the availability factor versus number of replicas and queries. When the number of queries increases then there is increase in the availability and also by the increase in the number of availability then we can increase the creation of number of replicas. Thus the related data can be placed in a cloud and the time taken to access the query related data can be done quickly and effectively.

Table 3. Query generated for the process

1.SELECT A.[ID], A.[First Date], A.[SEX], A.[Birthday], A.[Description], A.[Admission], A.[Diagnosis] FROM A
2.SELECT B.[ID], B.[Examination Date], B.[aCL IgG], B.[aCL IgM], B.[ANA], B.[ANA Pattern], B.[aCL IgA], B.[Diagnosis], B.[KCT], B.[RVVT], B.[LAC], B.[Symptoms], B.[Thrombosis] FROM B
3.SELECT C.[ID], C.[Date], C.[GOT], C.[GPT], C.[LDH], C.[ALP], C.[TP], C.[ALB], C.[UA], C.[UN], C.[CRE], C.[T-BIL], C.[T-CHO], C.[TG], C.[CPK], C.[GLU], C.[WBC], C.[RBC], C.[HGB], C.[HCT], C.[PLT], C.[PT], C.[APTT], C.[FG], C.[PIC], C.[TAT], C.[TAT2], C.[U-PRO], C.[IGG], C.[IGA], C.[IGM], C.[CRP], C.[RA], C.[RF], C.[C3], C.[C4], C.[RNP], C.[SM], C.[SC170], C.[SSA], C.[SSB], C.[CENTROMEA], C.[DNA], C.[DNA-II] FROM C
4.SELECT A.ID, A.SEX, A.Birthday FROM A
5.SELECT C.Date, C.ALP FROM C
6.SELECT A.Birthday, A.Admission FROM A
7.SELECT B.ANA, B.[aCL IgA] FROM B
8.SELECT A.ID, A.Birthday FROM A
9.SELECT C.CENTROMEA, C.HGB, C.GPT FROM C
10.SELECT A.ID, A.Diagnosis FROM A
11.SELECT B.[Examination Date], B.Thrombosis FROM B
12.SELECT A.[First Date] FROM A

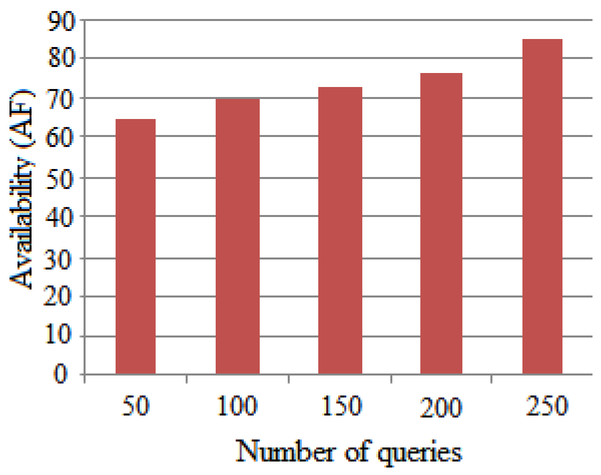

Figure.3 Availability versus Number of queries

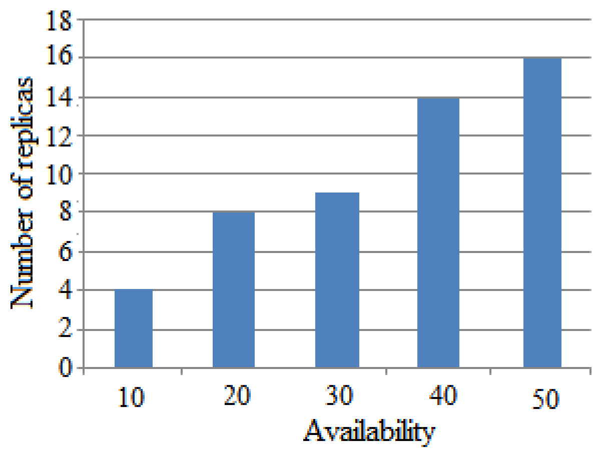

Figure.4 Number of replicas versus availability

5.3.Comparative Results

The performance of our proposed data replication methodis compared with the existing system [2] to show that our proposed technique has better performance than the previous approaches. In [2], their system suffers to increase data availability and improve cloud system task successful execution rate.

❖ Effectiveness of Creating Replica

In our proposed approach, we find the data that need to be replicated and then for that data the replica is created using popularity degree and replica factor. If the replica generated is not for the right file, the system performance will not get improve. This leads to increase in waiting time and also slows down the process. So selecting the right file to replicate is much needed for the performance of the system. To choose the right file to replicate PD and RF is used. This helps to reduce the waiting time and also speed up our process. By using our proposed technique it shows that creating replicas reduces the time taken by the query to execute the process.

❖ Effectiveness of Placing Replica

After finding the number of replica, the replica is created and then this replica is placed in corresponding nodes. This is calculated using the availability factor and classification criteria. With the help of the classification criteria the nodes are arranged based on the availability of nodes. Thus it reduces the overload in the network and improves the bandwidth utilization.

The performance of SBER is compared with the existing and our proposed approach and our result shows that the value of SBER is high compared to our existing approach. Thus our proposed approach gives better result than our previous approaches. The comparison of our proposed and the existing method using the system byte effective rate (SBER) is shown in figure 5. The figure 5 shows the value of SBER at each time interval. At each time interval the value of SBER increases or decreases. Overall, our system achieves that a suitable number of copies to meet a reasonable system byte effective rate requirement and placing replicas among data nodes in a balanced way.

The performance based on the queries is shown in fig 6. When the number of query increases the execution time also increases.

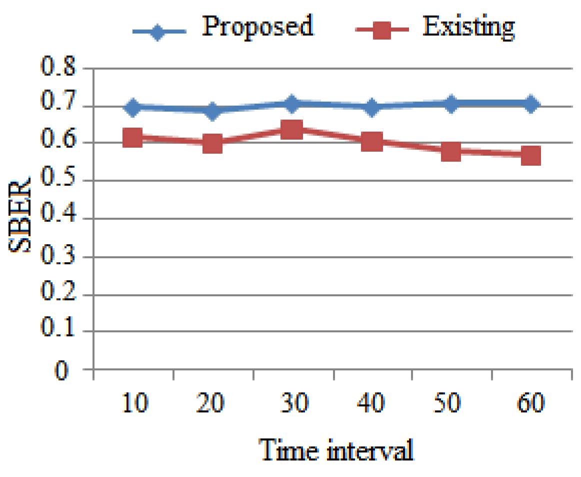

Figure.5 Performance of SBER

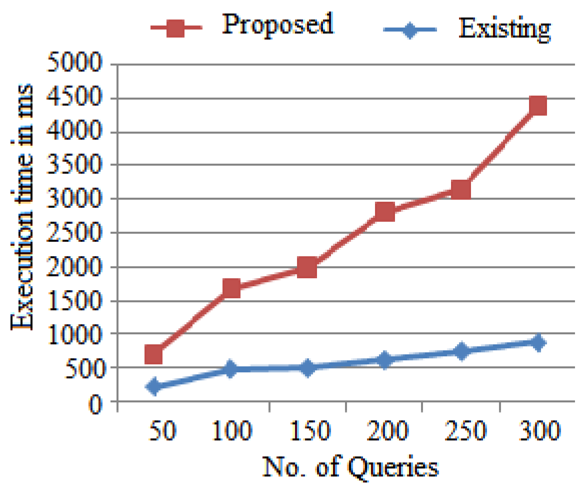

Figure.6 Performance of execution

The performance of a system depends on the time taken to process the query. If the query is processed soon and the result is arrived to the user in a short period of time then the performance of the system also increases. In fig 6, the number of query given to process and the time taken to execute that query is compared with the existing and our proposed approach. While comparing with our proposed approach, it takes less amount of time to execute each amount of query compared to our existing approach. Thus it shows that the performance of the system increases.

6. Conclusion

In this paper, we proposed a dynamic replication strategy and placement algorithm to improve the performance of the software system.In this technique the files to be replicated and the number of replicated files are identified using the popularity degree and the replica factor. It increased data availability by providing users with different replicas of the same service. Thereafter we used placement algorithm and placed the replicas in the corresponding nodes or to the identified system. This is calculated using the availability factor and it is classified by classification criteria and the nodes are arranged in an order. Thus the replicas are placed in the system. We also proposed the placement of replicas in anefficient manner which increased the performance of the system without overloading the system nodes. It strives to increase data availability, improve cloud system task successful execution rate and minimize cloud system bandwidth consumption. The performance is compared with the existing system. From the experimental results, we showed that our proposed technique outperforms than the existing technique.In future, some significant improvements like further reducing the user waiting time, speeding up data access, and further increasing data availabilitywill be proposed.

Reference

[1] W. Kong, Y. Lei and J. Ma, "Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism", *Journal homepage*, Vol. 127, No. 12, pp. 5099-5104, 2016.

[2] D.W. Sun, G.R. Chang, S.Gao, L.Z.Jin and X.W.Wang, "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments", *Journal of computer science and technology*, Vol. 27, No. 2, pp. 256-272, 2012.

[3] H. Lamehamedi,"Decentralized data management framework for data grids", *PhD thesis, Faculty of Rensselaer Polytechnic Institute, New- York*, Vol. 23, No. 1, pp. 109-115, 2007.

[4] O. Wolfson, S. Jajodia, Y. Huang, "An adaptive data replication algorithm", *ACM Transactions on Database Systems*, Vol. 22, No. 2, pp. 255-314, 1997.

[5] M. Rabinovich, I. Rabinovich and R. Rajaraman, "Dynamic replication on the internet", *Technical Report*, HA6177000–980305-01-TM, AT&T Labs, March 1998.

[6] J. M. Perez, F. G. Carballeira, J. Carretero, A. Calderon and J. Fernandez, "Branch replication scheme: a new model for data replication in large scale data grids", *Future Generation Computer Systems*, Vol. 26, No. 1, pp. 12-20, 2010.

[7] M. Vrable, S. Savage and G. M. Voelker, "Cumulus: file system backup to the cloud", *ACM Transactions on Storage*, Vol. 5, No. 4, pp. 14, 2009.

[8] H. Shen, "An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21,No. 6, pp. 827-840, 2010.

[9] M. C. Lee, F. Y. Leu and Y. P. Chen, "PFRF adaptive data replication algorithm based on star-topology data grids: An", *Future Generation Computer Systems*, Vol. 28, No. 7, pp. 1045-1057, 2012.

[10] S. Y. Ko, R. Morales, I. Gupta, "New worker-centric scheduling strategies for data-intensive grid applications", *In Proc.ACM/IFIP/USENIX Int'l Conference on Middleware*, pp. 121-142, 2007.

[11] L. Meyer, J. Annis, M. Wilde, M. Mattoso and I. Foster, "Planning spatial workflows to optimize grid performance", *In: Proc. ACM Symp. Applied Computing*, pp. 786-790, 2006.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, pp. 1345-1359, 2010

[13] A. Rajalakshmi, D. Vijayakumar and Dr. K .G. Srinivasagan, "An Improved Dynamic Data Replica Selection and Placement in Cloud", *Int. Conference on Recent Trends in Inf. Tech.,*, pp. 1-6, 2014.

[14] Z. Wang, T. Li, N. Xiong and Y. Pan, "A novel dynamic network data replication scheme based on historical access record and proactive deletion", *The Journal of Supercomputing*, Vol. 62, No. 1, pp. 227-250, 2012.

[15] W. K. Lin, C. Ye and D. M. Chiu, "Decentralized replication algorithms for improving file availability in P2P networks", *In 2007 5th IEEE International Workshop on Quality of Service*, pp. 29-37, 2007.

[16] K. Sashi and A. S. Thanamani, "Dynamic replication in a data grid using a Modified BHR Region Based Algorithm", *Future Generation Computer Systems* Vol. 27, No. 2, pp. 202-210, 2011.

[17] S. Kamali, P. Ghodsnia and K. Daudjee, "Dynamic data allocation with replication in distributed systems", *In 30th IEEE Int. Performance Computing and Communications Conference*, pp. 1-8, 2011.

[18] R. S. Chang and H. P. Chang, "A dynamic data replication strategy using access-weights in data grids", *The Journal of Supercomputing*, Vol. 45, No. 3, pp. 277-295, 2008.

[19] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry and A. Y. Zomaya, "Energy-efficient data replication in cloud computing datacenters", *Cluster Computing*, Vol. 18, No. 1, pp. 385-402, 2015.

[20] J. W. Lin, C. H. Chen and J. M. Chang, "QoS-aware data replication for data-intensive applications in cloud computing systems", *IEEE Transactions on Cloud Computing*, Vol. 1, No.1, pp. 101-115, 2013.

[21] M. K. Hussein And M. H. Mousa, "A Light-weight Data Replication for Cloud Data Centers Environment", *Int. Journal of Innovative Research in Com. and Comm. Engg*, Vol. 1, No. 6, pp. 169-175, 2012.

[22] N. Mansouri, "An effective weighted data replication strategy for data grid", *Australian Journal of Basic and Applied Sciences*, Vol. 6, No. 10, pp. 336-346, 2012.