Design and Implementation of a Synchronous and Asynchronous-Based Data Replication Technique in Cloud Computing

Kirubakaran Subramani, Valarmathy Shanmugasundaram, and Kamalanathan Chandran
Department of Electronics and Communication, Bannari Amman Institute of Technology, India

Abstract: *Failures are usual rather exceptional in cloud computing environment. To access from the nearby site, the often used data should get replicated to multiple locations to compose the users to advance the system accessibility. A challenging task in cloud computing is to decide a sensible number and right location of replicas. At this point, we propose an adapted dynamic data replication approach to decide a sensible number and right location of replicas and we compare both the adapted dynamic data replication approach and normal dynamic data replication approach. The normal dynamic data replication approach has three dissimilar stages which are the recognition of data file to replicate, number of replicas to be created and placing new replicas. We adapt the popularity degree in the initial stage of normal dynamic data replication approach and also we consider the failure probability for replica factor calculation and the other two stages are related to the normal dynamic data replication approach. When we update the major data center we moreover integrate the synchronous and asynchronous updation of replica data file.*

Keywords: *Cloud computing, data replication, synchronous, asynchronous updation.*

Received May 24, 2013; accepted July 11, 2013; published online March 8, 2015

1. Introduction

The achievement of contemporary technologies awfully depends on its helpfulness of the world's norms, its simplicity of use by end users and most considerably its degree of information security and control. Cloud computing is a novel and increasing information technology that alters the way IT architectural solutions are recommended using moving towards the subject of virtualization: of data storage, of local networks (infrastructure) as well as software [2, 3, 4, 5]. It is a comprehensive distributed computing paradigm driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, highly accessible and configurable and reconfigurable computing resources (e.g., networks, servers, storage, applications and data) can be rapidly provisioned and discharged with slightest management effort in the data centers. Services are carried on demand to outdoor customers over high-speed Internet with the X as a Service (XaaS) "computing architecture, which is ruined down into three dissimilar parts: Applications, platforms and infrastructure".

Conversely, in the cloud, applications are accessible anywhere, any-time and storage turns into infinite for all intents and purposes from a sociological standpoint. Along with the users can permission the powerful applications, platforms and services distributed over Internet. Moreover, high availability, high fault tolerance and high efficiency access to cloud data centers where disappointments are usual rather than outstanding are important issues, owing to the huge data support. Data replication permits dipping user waiting time, speeding up data access and rising data accessibility by offering the user with different models of the similar service, all of them with a rational state. Reproduction is an often used method in the cloud, such as Google File System (GFS) [9], Hadoop Distributed File System (HDFS) [7]. On the other hand, cloud data centers have grown sharply in both size and number and the vigorously scalable and totally virtualized resources are presented as a service over the Internet [10].

To progress system availability (by directing traffic to a replica after a failure), avoid data loss (by recovering lost data from a replica), Reproduction is applied and develop presentation (by spreading load across multiple replicas and by making low-latency access available to users around the world). There are different strategies to replication on the other hand. Synchronous replication promises all copies are up to date, however potentially gains high latency on renews. In addition, if synchronously replicated renews can never achieve while a few models are offline accessibility may be impacted. Asynchronous replication eliminates high write latency (in exacting, making it appropriate for wide area replication) but allows models to be out of date. Besides, due to collapse, data loss may happen if renew is lost before it can be reproduced [1]. There are three crucial problems that must be worked out in order to, achieve the dynamic data replication:

1. Which data should be reproduced and when to reproduce in the cloud systems to meet the user's necessities on waiting time reduction and data access speeding up are considerable issues for further research, as the wrongly chosen and too early reproduced data will not decrease the waiting time or speed up data access.
2. How many appropriate novel models should be created in the cloud to meet a sensible system accessibility necessity is another significant issue to be comprehensively examined. The system maintenance costs will significantly increase with the number of novel models increasing and to a lot of models may not improve accessibility, however bring unnecessary spending instead.
3. Where the novel models should be placed to meet the system task thriving execution rate and bandwidth consumption necessities is furthermore a crucial issue to be investigated in detail. The models may advance system task successful execution rate by keeping all models vigorous and bandwidth consumption if the models and requests are rationally allocated. Suitable model placement in ultra-large-scale, vigorously scalable and totally virtualized data centers is much more complicated [8] on the other hand.

In cloud computing, we suggest a method for synchronous and asynchronous based data replication in this document. We at first make sure the system availability and after that we relate the adapted Dynamic Data Replication Strategy (D2RS) algorithm. The dynamic data replication approach contains three dissimilar stages. To decrease the waiting time, the initial stage is to recognize which data file should be replicated and when to replicate in cloud computing. We adapt the popularity degree in the initial stage of the D2RS algorithm [8]. The adaption of the popularity degree is based on double exponential moving average function. The next stage of the D2RS algorithm is to find the necessary number of models and the third stage is to find where to put the novel replica data files. In usual dynamic data replication approach, the popularity degree is computed between the start time and present time however in our adapted dynamic data replication approach, we computed the access frequency based on time factor and users and the access frequency based on time factor is computed by means of double exponential moving average function and also we included the failure probability in replica factor calculation. For the recently created models we furthermore integrate the synchronous and asynchronous updation. In synchronous updation, the record which we revise in the key datacenter will get revise at the same time to the models in the sub datacenters however, in asynchronous updation; the record which we revise in the key datacenter will get revise to the models after a particular time interval by the asynchronous agent.

The remaining of the document is configured as follows: section 2 demonstrates assess of associated works, section 3 demonstrates system architecture, section 4 demonstrates our suggested method, section 5 explains the attained result and section 7 terminates our suggested method.

2. Review of Related Works

In cloud computing atmosphere, a handful of researches are obtainable in the literature for data replication model. Now, assess of latest works from these topics is offered. Storage systems are vital building blocks for cloud computing infrastructures. The implementation of low-cost storage system stays an open matter even though high concert storage servers are the ultimate solution for cloud storage. The competent cloud storage system with economical and commodity computer nodes has been executed by Myint and Naing [6] to address this problem, that are arranged into PC cluster based datacenter. HDFS was an open source cloud based storage platform and planned to be arranged in low-cost hardware. By improving replication management plan, PC cluster based cloud storage system was executed with HDFS. Data objects were allocated and reproduced in a cluster of commodity nodes placed in the cloud. That system offered optimum replica number with weighting and balancing between the storage server nodes. The experimental effects demonstrated that storage was balanced depending on the presented disk space, expected accessibility and failure possibility of every node in PC cluster.

In the cloud computing atmospheres, disappointments are usual rather than exceptional. To develop system accessibility, replicating the famous data to multiple appropriate locations is a sensible option, as users can access the data from a close by site. This is, on the other hand, not the case for models which must have a permanent number of copies on numerous locations. In the cloud computing, deciding a sensible number and right sites for models has turn out to be a challenge. For allocated computing environments, Sun *et al*. [8] improved a vibrant data replication approach was put ahead with a concise review of replication approach appropriate. It comprised: Examining and modeling the correlation between system availability and the number of replicas, assessing and identifying the famous data and activating a replication operation when the reputation data passes a vibrant threshold, computing an appropriate number of copies to convene a sensible system byte effective rate requirement and placing models among data nodes in a balanced way, planning the dynamic data replication algorithm in a cloud. In a cloud, experimental effects showed the competence and efficiency of the developed system brought by the suggested approach.

In the cloud atmosphere, an adaptive replication approach has been suggested by Hussein and Mousa [4]. The approach examines the accessibility and competent access of each file in the data center, and learned how to develop the dependability of the data

files based on calculation of the user access to the blocks of every file. Using heuristic search for every replication, the suggested adaptive replication approach reorganized vigorously large-scale dissimilar files models on dissimilar data nodes with minimal cost. The suggested adaptive approach was based on a proper explanation of the problem. Based on examining the latest history of the data access to the files by means of HLES time series, the approach recognized the files which were famous file for replication. The replication signal was activated once a replication factor based on the popularity of the files was less than a particular threshold. Therefore, the adaptive approach recognizes the most excellent replication location based on a heuristic search for the best replication factor of every file. In the cloud atmosphere, experimental assessment expressed the competence of the suggested adaptive replication approach.

3. System Architecture

This part illustrate the system architecture of the cloud computing. Figure 1 illustrates the sample cloud computing system.

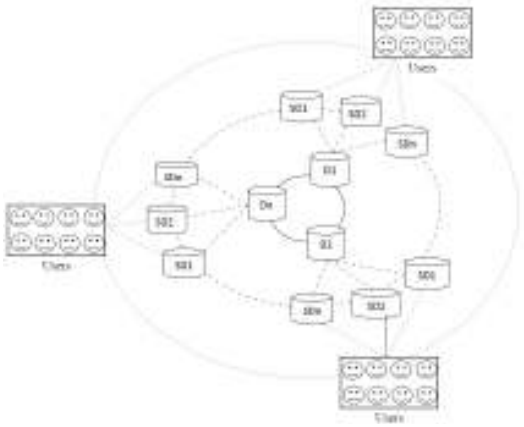

Figure 1. Sample cloud system topology.

In the Figure 1 *D*1, *D*2, ..., *Dn*, are the key data centers and *SD*1, *SD*2, …, *SDn* are sub data centers of the key data centers *D*1, *D*2, …, *Dn*. At this point, each key data centers contain *n* number of sub data centers. The sub data centers may or may not have another set of sub data centers. The users will employ the least set of sub data centers. Each key data centers are interconnected with each other and each set of sub data centers are also interconnected with each other to shift the data from one data center to another. It is not definite that all the data in the data center *D*1 would be in the data center *D*2 and furthermore it is not definite that the complete data in the sub data center *SD*1 of the key data center *D*1 is present in the sub data center *SD*2 of the key data center *D*1. If the user who uses the data center *SD*1 needs a data which is not in *SD*1 but which is present in, the data center demand a model of the data to or to its key data center. A different case is that if a data in the data center is employed by more number of users and if a novel user comes to use the similar data, he requires waiting to use that data. Thus, to keep away from the waiting time, we require making a model of that data in the data center.

Figure 2 illustrates the model cloud data center architecture. It encloses users, scheduling manager, replica manager and data centers. The assignment specified by the user is first send to the scheduling manager. In order to, access the file, the scheduling manager next offers the path to the specified data center. The path choice by the scheduling manager is based on the number of users uses a data center to keep away from congestion.

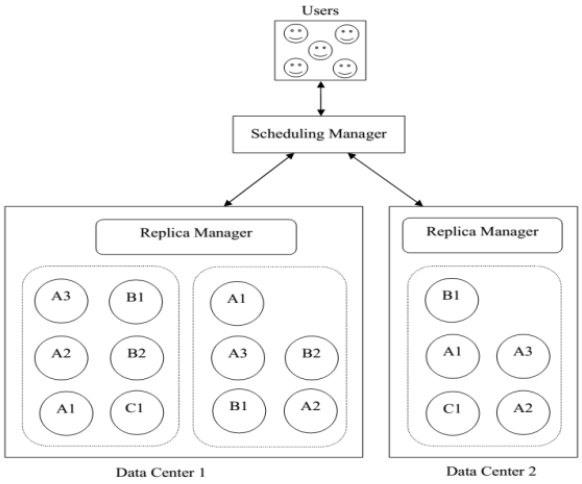

Figure 2. Sample cloud data center architecture.

4. Proposed Synchronous and Asynchronous Based Data Replication

In cloud computing, this part describes our suggested method of synchronous and asynchronous based data replication. The processes engaged in our proposed method are possibility of file accessibility and unavailability, which file and when to reproduce, total number of replication, where to put the model and the synchronous and asynchronous revise after replication. Figure 3 illustrates the block diagram of our suggested method.

To find the System Byte Effective Rate (SBER) [8], the accessibility of file and unavailability of the file is computed. After that, we employ the adapted D2RS algorithm. The usual D2RS [8] contain three steps that are: Which data file should be replicated and when to replicate in the cloud system, how many models to be created in the cloud system and where we have to put the novel models. The alteration we do is the inclusion of failure probability in replica factor calculation and the computation of the popularity degree which is the access frequency based on time factor that is applied in the initial stage of the usual D2RS. In our technique, excluding the modification, all the process is similar as [8] and we integrate the synchronous and asynchronous updation. In usual D2RS, the popularity degree is computed regarding the start time and the present time but in our adapted D2RS, the popularity degree is computed by means of access frequency

based on time factor and access frequency based on users.

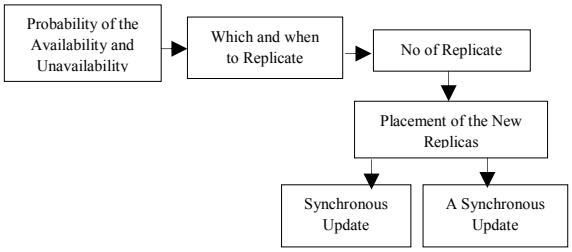

Figure 3. Block diagram of our recommended technique.

4.1. Popularity Degree

Based on the time factor and users, the popularity degree of a block is the access frequency. The popularity degree is computed from the start time to the present time. In our suggested method, based on users, the popularity degree is computed by access frequency based on time factor and the access frequency. The access frequency based on time factor is computed by double exponential moving average function. The popularity degree is computed as follows:

$$P_d = F_1 + F_2 \tag{1}$$

Where P_d: Popularity degree, F_1: Access frequency based on time factor, and F_2: Access frequency based on users. The access frequency based on the time factor F_1 is worked out based on the double exponential moving average. It is proved by the equation beneath:

$$F_1 = 2[f(an_t)] - f(f(an_t)) \tag{2}$$

Where $f(an_t)$: Access frequency based on time interval, and $f(f(an_t))$: Access frequency based on the time interval of the time interval we split.

The procedure to discover the access frequency based on the time interval we split is as follows: At first we divide the time intervals from the present time t_p to start time t_s. A model splitting of time intervals from the present time t_p to start time t_s is demonstrated in Figure 4. After that, we work out the access frequency within the time interval which we divide and ultimately, we sum all the access frequency we attained for the relevant time intervals. The access frequency based on the time interval we divide is proven by an equation beneath:

$$f(an_t)_{t \to t_s \; to \; t_p} = \alpha an_{t_p} + (1-\alpha) an_{t_{p1}} + (1-\alpha)^2 an_{t_{p2}} + (1-\alpha)^3 an_{t_{p3}} + \ldots + (1-\alpha)^k an_{t_s} \tag{3}$$

Where an_{tp}: Access frequency with respect to time interval, and α: Constant.

0 to 50	51 to 100	101 to 150	151 to 200	201 to 250	251 to 300	301 to 350
t_s		…		t_{p2}	t_{p1}	t_p

Figure 4. Sample time splitting.

Figure 4 demonstrates the model splitting of time between t_s and t_p. In this figure, t_p is the present time which takes the interval from 301 to 350 seconds and t_{p1} has the time interval from 251 to 300 seconds and t_{p2} has the time interval from 201 to 250 seconds and ultimately, t_s is the start time that has the time interval from 0 to 50 seconds.

After working out the access frequency based on time interval, we have to compute the access frequency based on the time interval of the time interval we divide i.e., we have to divide the time interval of t_p, t_{p1}, t_{p2}, …, t_s. The equation specified beneath demonstrates the formula to find the access frequency based on the time interval of the time interval we divide by the access frequency based on time interval $f(an_t)$ and the access frequency based on time interval of the time interval $f(f(an_t))$, we can discover the value of the access frequency based on time factor $F1$.

$$f\left(f\left(an_t\right)\right) = f\left(an_{t_p}\right) + f\left(an_{t_{p1}}\right) + f\left(an_{t_{p2}}\right) + \ldots\ldots\ldots + f\left(an_{t_s}\right) \tag{4}$$

The access frequency based on the user is computed as follows: At first we require to find the access frequency of each user individually from the start time to the present time. After that, the access frequency of each user is multiplied with the weight value of the relevant users and ultimately, we require summing all the values. It is proven by the equation below:

$$F_2 = \sum_{i=1}^{N} an_{ui} * W_{ui} \qquad t \to t_s \; to \; t_p \tag{5}$$

Where An_{ui} : Access frequency of i^{th} user, W_{ui}: Weight value of i^{th} user, and N: Total number of users.

4.1.1. Replica Factor

Consider t_1, t_2 and t_3 are the three different time intervals that has the replica numbers, the calculation of the failure probability at t_3 is the replica numbers in t_2 divided by the replica numbers in t_1. Except the popularity degree calculation and failure probability calculation, all the calculations are same as [8]. The replica factor of [8] is modified by including the failure probability. The modified replica factor is the ratio of product of popularity degree and failure probability to the product of number of replicas and file size of data file. It is shown by Equation 6:

$$R_f = \frac{P_d \times F_p}{R_n \times F_s} \tag{6}$$

Where R_f : Replica factor, P_d: Popularity degree, F_p; Failure probability, R_n: Number of replicas and F_s : File size of the data file.

4.2. Synchronous and Asynchronous Update

This part makes clear the synchronous update and asynchronous update of our suggested method. Figure 5 demonstrates a model diagram for the synchronous and asynchronous update.

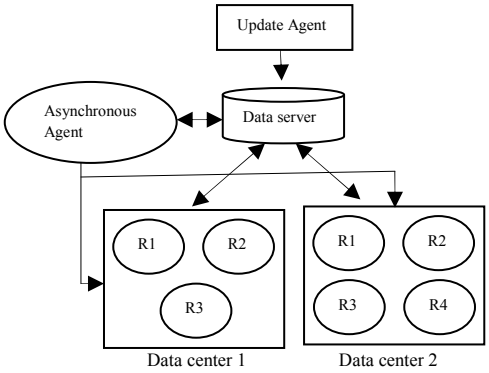

Figure 5. Sample block diagram for synchronous and asynchronous update.

4.2.1. Update Agent

In the data server, this is the agent who revises i.e., adding or deleting any information. The data in the data server will not be similar for every time we open a page. It would get change when the revise agent updates the page in the data server. The update agent is the authorized agent to change any information in the data server and excluding the update agent no one can do any modification in the data server.

4.2.2. Data Server

The data server is the key data center that contains many sub data centers. The key data center encloses a compilation of information and it allocates that information through the cloud and the users can employ that information from the cloud network. Therefore, any modification of information in the key data server would alter the similar copy present in the sub data centers which offers the information to the users.

4.2.3. Data Center

The data centers are the sub data centers which are linked with the key data center and share the information from the key data center. In the cloud network, the users employ the data from the sub data center and the files which are reproduced are present in the sub data center. The sub data centers have the data which is in the key data center. If a file is insisted by more number of users, it would produce a replica of that file to keep away from congestion and waiting time. The models created in the sub data center should get modify when we change the source file of that model in the key data center.

4.2.4. Asynchronous Agent

After a particular time interval, the asynchronous agent is an agent that revises the model in the sub data center. The modification made in the key data center should get revised in the model present in the sub data centers. Frequently, when we revise the information in the key data center, the suitable models present in the sub data centers would in addition get revised automatically. But, we can never be definite that all the models present in the data centers would get revised. The usual concurrent modification of data in the sub data centers when the source file is modified in the key data center is called synchronous update. The asynchronous agent revises all the models in the sub data centers for every particular time interval. Therefore, each model in the sub data centers would get revised in the particular time interval even if the model was not revised by means of synchronous update.

5. Results and Discussions

This part describes the results we attained for our suggested method and compare the performance with the presented method and furthermore makes clear the comparison of synchronous and asynchronous updation.

5.1. Experimental Setup

The experimental set of our method employs java that employs java development kit version 1.6 and the system configuration for the setup: i5 processor with 4GB RAM. We employed three dissimilar datasets for processing our suggested method and the datasets are Financial, Medical and RDB. We compare our adapted D2RS with the usual D2RS by those three datasets we employed.

5.2. Experimental Procedure

This part demonstrates the experimental process of our suggested method. Figure 6 demonstrates a model dataset in the data center with its substances.

In Figure 6, the 'replication server' window demonstrates that the financial dataset encloses the tables as account, card, client, etc., and the 'column' section demonstrates the records in the report table. The 'server content' window in Figure 6 demonstrates the total number of records in each table there in the financial dataset after adding a record in the report table of the financial dataset; Figure 7 demonstrates the screenshot for synchronous updation.

In 'replication server' window the option for updation we chosen are synchronous. As a result when we revise a record in the server, it will get revised in the replica at the same time. We can see it in the 'Syn_Asyn_Replication Viewer' and in the 'Server Content' window. The 'server content' window demonstrates the features in the server and the 'Syn_Asyn_Replication Viewer' window demonstrates the features in the model. In both windows we can see the total number of records in the report table as 4504 however before revising the server it was 4503 that is proven in Figure 6.

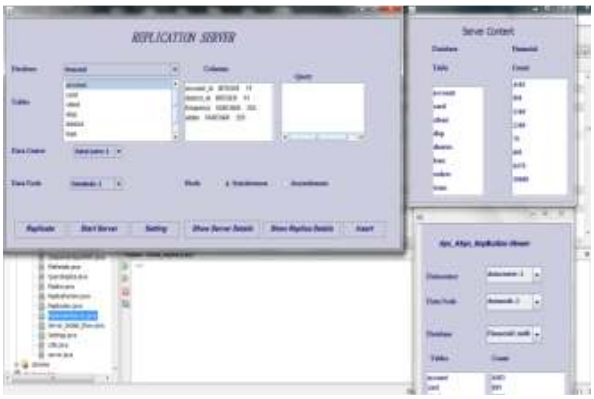

Figure 6. Sample dataset in the datacenter with its contents.

Figure 7. Synchronous updation.

Figure 8 demonstrates the screenshot for asynchronous updation after adding a record in the report table of the financial dataset. In 'replication server' window the option for updation we chosen are asynchronous. Hence, it will not get revised in the model at the same time when we revise a record in the server. The 'server content' window demonstrates the report table has 4505 records after adding a record and the 'Syn_Asyn_Replication Viewer' window demonstrates the report table has 4504 records only. In the model, it implies that the added record is not revised concurrently. Figure 9 demonstrates the screenshot for the query specified by the user.

Figure 8. Asynchronous updation.

Figure 9. Client query.

5.3. Performance Comparison

This part demonstrates the presentation of our method compared with the usual D2RS based on the number of model regarding time interval. Based on the implementation time, we moreover compare the synchronous and asynchronous updation.

Figure 10 demonstrates the total number of replicas created for our suggested and presented method regarding different time intervals when the variable parameter α [6] is 0.2. Now, when the time interval is ten seconds, the replica numbers created for the suggested method and the presented method is one and the replica numbers generated for the presented method is two; when the time interval is twenty seconds, the model created for the suggested method and the presented method is one. When the time interval is thirty seconds, the replica numbers created for the suggested method is one and it is three for the presented technique. When the time interval is forty, the replica numbers created for both the suggested and the presented method is one. The replica numbers created is two for the suggested method when the time interval is fifty and it is one for the presented method. When the time interval is sixty, the replica numbers created for the suggested method is two and it is one for the presented technique. When the time interval is seventy, the model created is one for the suggested method and it is two for the presented technique.

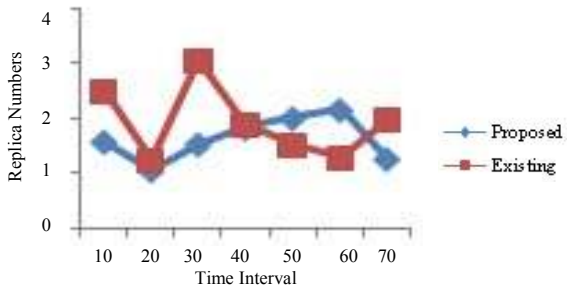

Figure 10. Total replica when α=0.2.

Figure 11 demonstrates the comparison of total number of model created for our suggested method and the presented method when the adjustable parameter α is 0.3.

Figure 12 demonstrates the comparison of the total number of model created for our suggested method and the presented method when the adjustable parameter is 0.4.

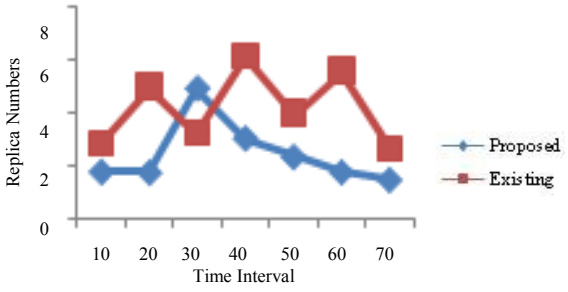

Figure 11. Total replica when α=0.3.

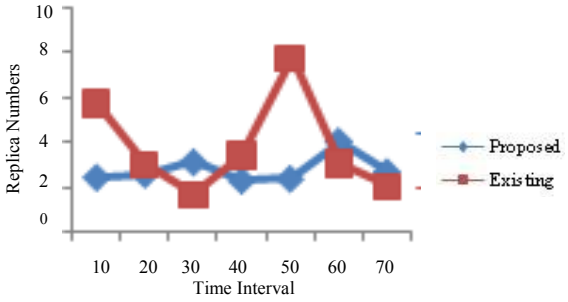

Figure 12. Total replica when α=0.4.

When the adjustable parameter is 0.8, the presentation of our suggested method and the presented method is illustrated in Figure 13. Here, when the time interval is ten, the SBER of our proposed technique is 0.596 and it is 0.33 for the existing technique; and when the time interval is twenty, the SBER of our proposed technique is 0.601 and it is 0.365 for the existing technique; and when the time interval is thirty, the SBER of our proposed technique is 0.632 and it is 0.318 for the existing work. Similarly for all the varied time intervals, the SBER of our proposed technique is higher than the existing technique which means the system availability is good in our proposed technique.

Based on the execution time, Figure 13 demonstrates the presentation comparison among the synchronous and asynchronous updation. At this point, when the queries given to carry out is hundred, the time taken for execution is 2024 ms based on synchronous updation and it is 1873 ms based on asynchronous updation. When we give 200 queries, the time taken for implementation is 5743ms in synchronous updation and 3245ms in asynchronous updation. When the query given is 300, the implementation time is 6734ms for synchronous updation and 4256ms for asynchronous updation.

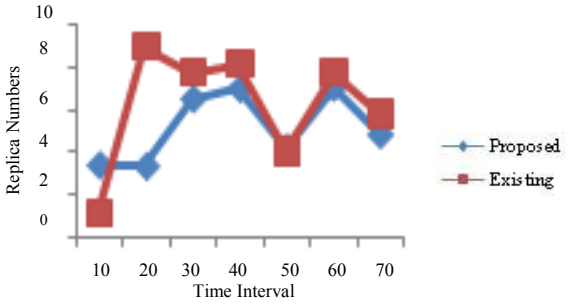

Figure 13. Total replica when α=0.8.

The record we alter in the main server will get revised concurrently to the client servers in synchronous updation however in asynchronous updation, it will not get revise in the client servers at the same time. As a result after the synchronous updation, when we give the query the processing time is high compared to the asynchronous updation since the data in the client server based on synchronous updation would be more compared to the data in the client server based on asynchronous updation. The updation in asynchronous technique will happen after a definite time period. So, after the updation is made in asynchronous mode and if we give the query, the implementation time would be similar for both modes of updating.

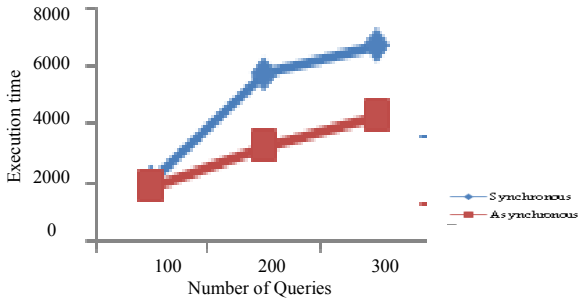

Figure 14. Execution time for synchronous and asynchronous updation.

6. Conclusions

We have suggested an adapted D2RS algorithm with synchronous and asynchronous updation in this document. Based on time factor, we have adapted the popularity degree which is the access frequency. Commonly, the D2RS contain three steps which are identification of data file to replicate, number of replicas to be created and where to place the models. The popularity degree is applied in the initial stage of the D2RS and we have adapted the initial stage of the vibrant data replication strategy. The alteration is based on the double exponential moving average function and the access frequency based on users. Based on the replica numbers, we have compared the modified D2RS and the normal D2RS in terms of system availability. We have moreover compared the implementation time of synchronous and asynchronous updation.

References

[1] Cooper B., Silberstein A, Tam E., Ramakrishnan R., and Sears R., "Benchmarking Cloud Serving Systems with YCSB," *in Proceedings of the 1st ACM Symposium on Cloud Computing*, Indian, pp. 143-154, 2010.

[2] Gao K., Wang Q., and Xi L., "Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment," *the*

International Arab Journal of Information Technology, vol. 11, no. 3, pp. 268-275, 2014.

[3] Ghemawat S., Gobioff H., and Leung S., "The Google File System," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29-43, 2003.

[4] Hussein M. and Mousa M., "A Light-weight Data Replication for Cloud Data Centers Environment," *International Journal of Engineering and Innovative Technology*, vol. 1, no. 6, pp. 169-175, 2012.

[5] Leavitt N., "Is Cloud Computing Really Ready for Prime Time?," *Computer*, vol. 42, no. 1, pp. 15-20, 2009.

[6] Myint J. and Nain G., "Management of Data Replication for Pc Cluster Based Cloud Storage System," *the International Journal on Cloud Computing: Services and Architecture*, vol. 1, no. 3, pp. 31-41, 2011.

[7] Shvachko K., Hairong K., Radia S., and Chansler R., "The Hadoop Distributed File System," available at: http://zoo.cs.yale.edu/classes/cs422/2014fa/readings/papers/shvachko10hdfs.pdf, last visited 2010.

[8] Sun D., Chang G., Gao S., Jin L., and Wang X., "Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments," *the Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 256-272. 2012.

[9] Weinhardt C., Anandasivam A., Blau B., and Stober J., "Business Models in the Service World," *IT Professional*, vol. 11, pp. 28-33, 2009.

[10] Wang S., Yan K., and Wang S., "Achieving Efficient Agreement within a Dual-Failure Cloud-Computing Environment," *Expert System with Applications*, vol. 38, no. 1, pp. 906-915, 2010.

Kirubakaran Subramani obtained his BE (electronics and communication engineering) from Bharathiyar University, Coimbatore in 2004 and ME (network engineering) Anna University of Technology-Coimbatore in 2009. Currently, he is pursuing his PhD degree from Anna University, Chennai in the area of cloud computing. He is presently working as Assistant Professor in the Department of Electronics and Communication at Bannari Amman Institute of Technology, Sathyamangalam. He is having a total of 6 years of teaching experience in various engineering colleges. His research interest includes P2P networks, overlay networks, data management in cloud computing and distributed systems. He has published 7 papers in International and National Conferences.

Valarmathy Shanmugasundaram received her BE (electronics and communication engineering) degree and ME (applied electronics) degree from Bharathiar University, Coimbatore in 1989 and 2000 respectively. She received her PhD degree at Anna University, Chennai in the area of Biometrics in 2009. She is presently working as Professor and Head in the department of Electronics and Communication Engineering, Bannari Amman Institute of Technology, Sathyamangalam. She is having a total of 21 years of teaching experience in various engineering colleges. Her research interest includes Biometrics, image processing, soft computing, pattern recognition and neural networks. She is the life member in Indian Society for Technical Education and Member in Institution of Engineers. She has published 33 papers in International and National Journals, 68 papers in International and National Conferences.

Kamalanathan Chandran received his BE (electronics and communication engineering) degree from Anna University, Chennai in May 2005 and M.Tech. (applied electronics) degree from Dr.MGR University, Chennai, in 2008. He is pursuing his PhD degree at Anna University, Chennai in the area of Cloud Computing. He is presently working as Assistant Professor in the department of Electronics and Communication Engineering, Bannari Amman Institute of Technology, Sathyamangalam. He is having a total of 6 years of teaching experience in engineering colleges. His research interest includes cloud computing, wired and wireless networks and network security. He is the life member in Indian Society for Technical Education. He has published 12 papers in International and National conferences.