International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 24083-24091

http://www.ripublication.com

Nmdd2rs: Increase The Data Availability For Cloud Environment

S.Kirubakaran[1], Dr.S.Valarmathy[2], C.Kamalanathan[3]
[1]Assistant Professor(Sr.G),ECE Dept, Bannanari Amman Institute of Technology, Sathy, e-mail:skirubame@gmail.com,
[2]Professor & Head, ECE Dept, Bannari Amman Institute of Technology Sathy, e-mail:atrmathy@gmail.com,
[3]Assistant Professor(Sr.G),ECE Dept, Bannanari Amman Institute of Technology, Sathy, e-mail:kamalanadhan@gmail.com

Abstract

In cloud computing, data replication provides a new way of sharing information to the target audience by keeping the redundant resources to further improve the reliability, fault-tolerance, or accessibility. Due to the advancement happened in cloud computing, researchers have developed different techniques to access from the nearby site, the often used data should get repeated to multiple locations to compose the users to advance the system accessibility. Here, a challenging task is to decide a sensible number and right location of redundant resource to be kept considering both objectives like, cost and time. In order to achieve this, Neural Modified Distributed D2RS (NMDD2RS) algorithm will be developed based on concept of *distributed computing.* The distributed algorithm can handle the updation and locating the resources frequently by giving only the most important information to other cloud nodes. Also, the distributed algorithm makes use of neural network to predict the error probability in finding the replica factor.

Keywords: Cloud Computing, Data Replication, NMD2RS Algorithm, Neural Network

Introduction

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 GFS (Google file system) [5]. 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 an update is lost before it can be reproduced.

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 users' 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 too 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 [2] on the other hand.

In this paper we propose a distributed data replication strategy for data replication in cloud computing using neural network and modified D2RS algorithm [1]. In modified D2RS algorithm [1], the popularity degree calculation in the first stage of the normal D2RS algorithm [2] is altered. Here, we do two modifications in [1] which are splitting the whole queries by the scheduler to process the task by reducing congestion and including the failure probability (error probability) in the calculation of the replica factor. The failure probability is a predicted value by means of neural network. The neural network is trained based on the historical data to predict the failure probability. The inclusion of failure probability in the replica factor calculation would improve the system availability by replicating the right file which is more demanded by the users. The structure of this paper is organized as follows: the second section shows Proposed Distributed Data Replication Strategy and the third section explains the Failure Probability Calculation and the fourth section shows the results obtained for our work and the fifth section concludes our work.

Proposed NMDD2RS

This section delineates the proposed distributed data replication strategy using modified dynamic data replication strategy [1] and neural network. The Fig.1 shows the sample cloud data center architecture of our proposed technique.

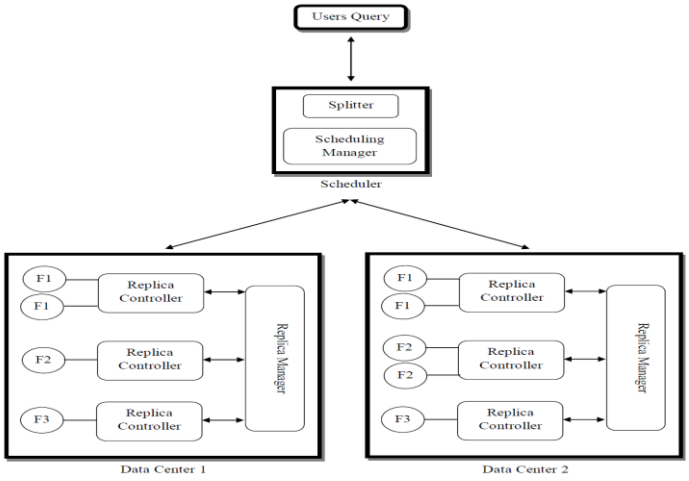

Figure 1: Sample cloud data center architecture

Generally, there would be more number of sub datacenters which are interlinked and connected to a master datacenter in cloud system. The Fig.1 shows two sub datacenters which are represented as Data Center 1 and Data Center 2. The queries from the users are given to the scheduler. The scheduler contains splitter and scheduling manager. The splitter would split the queries at a particular time and give to the scheduling manager to perform the job. By means of splitting a large number of queries, the congestion would be avoided and the processing time would be reduced. Thereafter the scheduling manager will send the queries to the corresponding datacenters to complete the job (task). In the datacenter, for each type of file there would have replica controller and the replica controllers are managed by replica manager. There has one replica manager for each datacenter.

To replicate a more demanded file, the replica factor calculation in D2RS algorithm [2] is an essential factor that uses popularity degree, existing number of replicas and file size. In [1] we modified the popularity degree calculation of [2] and all the other process are same as [2]. The popularity degree calculation used in the first stage of D2RS [2] is modified in [1] and showed better system availability than the normal D2RS [2] in our previous work [1]. Here we do two modifications on [1] by splitting the queries given by users in the scheduler to reduce the time consumption and including the failure probability (error probability) factor in the calculation of replica factor used in the first stage of [1, 2] to improve the system availability by replicating the more demanded file or data. In [1, 2] the replica factor of a file is calculated by means of three factors which are popularity degree of the file, existing replica numbers and file size. In this work, the replica factor is calculated using four factors which are popularity degree of the file, failure probability, file size and

existing replica numbers. Except the two modifications, all the other processes are same as [1].

Failure Probability

The failure probability (error probability) is obtained using neural network. To get the failure probability, the neural network is trained using historical data. The historical data contains system byte effective rate (SBER), replica numbers and popularity degree for different time intervals. The popularity degree and replica numbers are calculated for each file and it is summed and stored in the historical data. The Fig.2 shows sample historical data.

In this Fig.2, the 'Time' denotes the time intervals; and the 'SBER' denotes the system availability at corresponding time interval; and the 'RN' denotes the total replicas generated at corresponding time interval

Time	SBER	RN	PD	NN Training Target
t1	0.776179	1	0.767614	0
t2	0.811226	1	0.717022	0
t3	0.742927	1	0.80498	1
t4	0.749741	1	0.753878	0
t5	0.7636	1	0.652915	0
t6	0.796746	1	0.56785	0
t7	0.735563	1	0.766307	1
t8	0.760901	1	0.529768	0
t9	0.783398	1	0.517882	0

Figure 2: Sample Historical data

and the 'PD' denotes the total popularity degree value at corresponding time interval and the 'NN Training Target' denotes the target set to train the neural network. The target is based on the SBER of each time interval i.e. if the SBER value of the current time interval is greater than the SBER value of the previous time interval, the target would be 'zero' otherwise it would be 'one'. The SBER values, the replica numbers and the popularity degree in the historical data are calculated based on [1]. The historical data are given as input to the neural network to train it to obtain the failure probability value. The Fig.3 shows the feed forward neural network that gets input from the historical data.

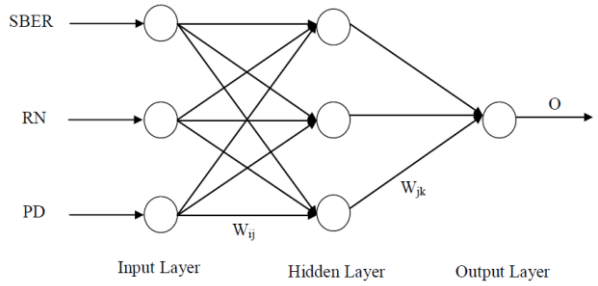

Figure 3: Feed Forward Neural Network

In general, the neural network would have three layers which are input layer, hidden layer and output layer. In between each input and hidden layer and in between each hidden and output layer, there would have weight values. Initially, the weight values are randomly generated. While training the neural network based on the target, the weight values would be changed to adjust the output from each neuron (node). In this Fig.3, the SBER, the RN and the PD from the historical data are given as input to the neural network. The W_{ij} denotes the weight values between the i^{th} neuron in the input layer and j^{th} neuron in the hidden layer. The W_{jk} denotes the weight values between j^{th} neuron in the hidden layer and k^{th} neuron in the output layer. The neural network is trained based on the 'NN Training Target' of the historical data. The weight values between each layer while training is adjusted based on back propagation algorithm. An example of back propagate on algorithm is as follows:

Example of Back Propagation Algorithm
The Fig.4 shows a neural network with hidden and output layer to understand the process of back propagation algorithm.

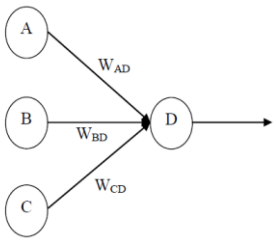

Figure 4: Sample Neural Network with hidden and output layer

Consider A, B and C in Fig.4 are the neurons (nodes) in the hidden layer and D is the neuron in the output layer. The W_{AD} represents the weight values between the nodes A and D, the W_{BD} represents the weight values between the nodes B and D, the W_{CD} represents the weight values between the nodes C and D. Initially, the error from the output neuron D is calculated as follows:

$$E_{D=}O_D(1-O_D)(T-O_d)$$

In the above equation, E_D denotes the error from the node D; and O_D denotes the output from the node D; and T denotes the target. The error from the neuron D in the output layer is used to change the weight values between the neurons in the hidden layer and the output layer. The weight values are adjusted as follows:

$$W_{AD}^{+} = W_{AD} + \left(E_D * O_A\right)$$

$$W_{BD}^{+} = W_{BD} + \left(E_D * O_B\right)$$

$$W_{CD}^{+} = W_{CD} + \left(E_D * O_C\right)$$

The W_{AD}^{+}, W_{BD}^{+} and W_{CD}^{+} in the above equations are new weight values; and W_{AD}, W_{BD} and W_{CD} in the above equations are existing weight values; and O_A denotes the output from the neuron A; and O_B denotes the output from the neuron B; and O_C denotes the output from the neuron C. The errors from the neurons in the hidden layer cannot be calculated directly as like the neuron of the output layer. Therefore, we have to back propagate it from the output layer. The errors from the neurons in the hidden layer are calculated as follows:

$$E_A = O_A\left(1 - O_A\right)\left(E_D * W_{AD}\right)$$

$$E_B = O_B\left(1 - O_B\right)\left(E_D * W_{BD}\right)$$

$$E_C = O_C\left(1 - O_C\right)\left(E_D * W_{CD}\right)$$

The errors from the neurons in the hidden layer are used to change the weight values between the neurons in the input layer and neurons in the hidden layer. The process would be repeated to train the neural network. After training the neural network, while testing the SBER, RN and PD calculated based on [1] is given as input to the neural network and the neural network will give the failure probability FP as output.

A. Replica Factor

The replica factor calculation of [1, 2] is modified by including the failure probability factor in it. The DMD2RS 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 the equation below:

$$RF = \frac{PD \times FP}{RN \times FS}$$

Where RF the replica factor; PD is the popularity degree [1]; FP is the failure probability; RN is number of replicas; and FS is the file size of the data file.

Performance Comparison

The performance of the proposed technique is compared with the existing technique [1] in terms of system availability. The system availability is checked for different

time intervals by setting different adjustable parameter α [2] which is based on different system performance.

Performance based on system availability

In the proposed technique the replica factor calculation is altered by including the failure probability which is obtained using neural network over the existing technique [1]. Therefore we compare the system availability of the proposed technique with the existing technique. The system availability is calculated based on system byte effective rate (SBER). The SBER is calculated for different time intervals by varying the adjustable parameter α [2]. The Fig.7 shows the SBER and replica numbers comparison when $\alpha = 0.2$.

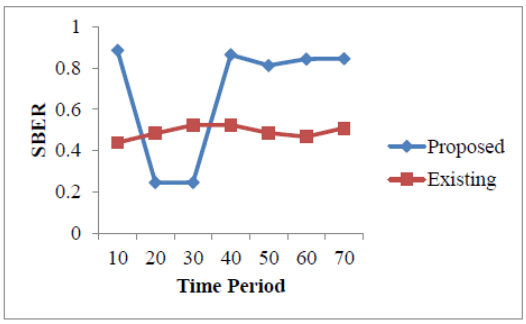

Figure 5(a): SBER Comparison when α=0.2

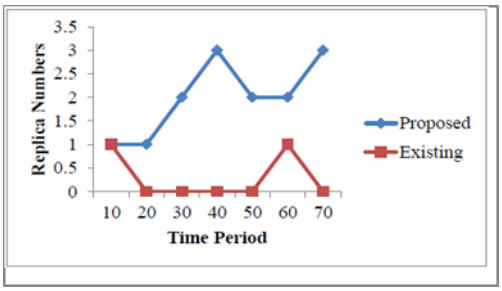

Figure 5(b): Replica Number Comparison when α=0.2

The Fig.5(a) shows the SBER comparison and the Fig.5(b) shows the replica number comparison for different time intervals when the adjustable parameter is 0.2. Here, the system availability is better compared to the existing technique for almost all the time intervals except the time intervals 20 and 30. The Fig.8 shows the comparison of SBER and replica numbers generated for different time intervals when $\alpha = 0.4$.

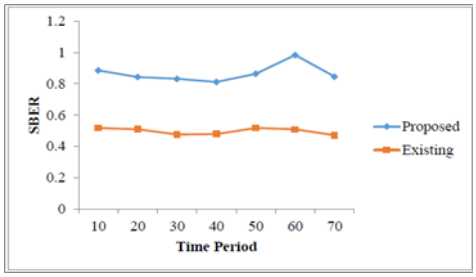

Figure 6(a): SBER Comparison when $\alpha = 0.4$

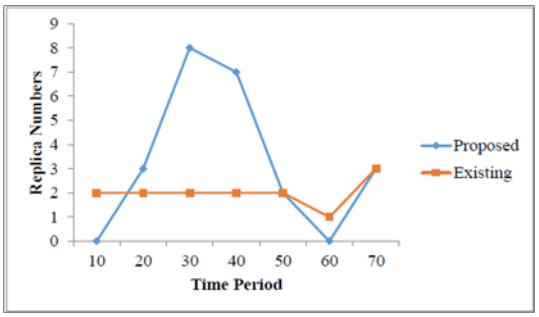

Figure 6(b): Replica Number Comparison when $\alpha = 0.4$

The Fig.6(a) shows the SBER comparison and the Fig.6(b) shows the comparison of replica numbers generated for different time intervals when the adjustable parameter is 0.4. Here, the system availability of the proposed technique is high for all the time intervals compared to the existing technique. At the time intervals 10 and 60, the replica numbers generated for our proposed technique is lesser than the existing technique. Even then the SBER of our proposed technique is higher than the existing technique at the time intervals 10 and 60. This denotes that our proposed technique generated replica for the right file than the existing technique.

Conclusion

In this paper we proposed a distributed data replication strategy for data replication in cloud computing using neural network and modified D2RS algorithm. Here, the queries given by the users are split and distributed by the scheduler to the corresponding data center in the cloud system to perform the task by reducing time consumption. We also modified the replica factor calculation of [1, 2] by including the failure probability (error probability) which is predicted using neural network to improve the system availability by replicating the more demanded file or data. The performance of the proposed technique is compared with existing technique [1] in terms system availability. The proposed technique outperformed the existing technique.

References

[1] S.Kirubakaran, Dr.S.Valarmathy and C.Kamalanathan, 2013,"Data Replication Using Modified D2RS In Cloud Computing For Performance Improvement", Journal of Theoretical and Applied Information Technology (JATIT), Vol. 58, No. 2, Pp. 460-470.

[2] Sun DW, Chang GR, Gao S., 2012,"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 Mar.

[3] Leavitt N, 2009, "Is Cloud Computing Really Ready for Prime Time?," Computer, Vol. 42, pp. 15-20.

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

[5] Ghemawat S, Gobio H, Leung S T. "The Google file system, 2003,"ACM SIGOPS Operating Systems Review,vol.37,no.5,pp.29-43..

[6] Foster I, Zhao Y, Raicu I, Lu S Y,2008 "Cloud computing and grid computing 360-degree compared", In Proc. Grid Computing Environments Workshop, Austin, TX, USA, Nov. 12-16, pp. 1-10.

[7] Buyya R, Yeo C S, Venugopal S, Broberg J, Brandic I. 2009, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, 25(6): 599-616.

[8] Armbrust M, Fox A, Gri±th R, Joseph A D, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M., 2010, "A view of cloud computing", Communications of the ACM, 53(4): 50-58.

[9] Mell P, Grance T. "The NIST definition of cloud computing", Communications of the ACM, 2010, 53(6): 50.

[10] Iosup A, Ostermann S, Yigitbasi N, Prodan R, Fahringer T, Epema D H J, 2011, "Performance analysis of cloud computing services for many-tasks scientific computing", IEEE Transactions on Parallel and Distributed Systems, 22(6): 931-945.