A Data Replication Using HQFR In Cloud Data Center

Dr.N.Murali [1], Mr.J.Noorul Ameen[2], Ms.B.Abarnadevi[3], Ms.K.Vinothini [4]

Department of Computer Science And Engineering

[1] Professor, E.G.S.Pillay Engineering College, Nagapattinam, Tamilnadu, India.

[2] Assistant Professor, E.G.S.Pillay Engineering College, Nagapattinam, Tamilnadu, India.

[3,4] E.G.S.Pillay Engineering College, Nagapattinam, Tamilnadu, India.

Abstract- *Cloud computing is a large scale parallel and distributed computing system. It consists of a collection of inter connected and virtualized computing resources that are managed to be one or more unified computing resources. The aim of the survey is an automated analysis or interpretation of ongoing events in data replication system in cloud. However, replica management systems usually need to migrate and create a large number of data replicas over time between and within data centers, incurring a large overhead in terms of network load and availability. The performance of these published works is analyzed efficiently with the evaluation metrics such as Throughput, Execution time, Storage utilization, Replica number, Network usage, Replication cost, Recovery time and Hit ratio. Higher throughput value indicates that the proposed approach achieves better result. To support such type of applications continuously performing data replication on the basis of QoS requirements of the corresponding application. For performing such type of data replication developing an algorithm using some concepts of HQFR [High QoS First Replication] algorithm. Along with the QoS requirement main goal is to minimize data replication cost. So developing another algorithm which is inspired from MCMF [Minimum Cost Maximum Flow] algorithm. At the end will propose an efficient scheme for data replication on the basis of QoS requirement.*

Keywords- Cloud storage, Data availability, Data migration, Replica management

I. INTRODUCTION

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. It gives clients with most adaptable services in a clear manner and with less expensive and most powerful processors. 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. 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. With reference to Replication strategies, replica optimization is one of the performance enhancement techniques for software system.

A dynamic data replication with placement algorithm strategy to enhance the performance of software system . Here use dynamic data replication using popularity degree and replica factor and placement algorithm.

Proposed approach consists of three phases:

1) Selecting Replica using PD,
2) Creating replica using RF and
3) Placing replica.

In the first phase; find the file to be replicated by calculating popularity degree (PD). In the second phase; create replicas with the help of popularity degree and replica factor (RF) in the third phase; place these replicas to the suitable location. Thus the simulation results shows that by using proposed approach it provides users with a system that has higher data availabilities, lower data transmission delays, and less bandwidth consumption for data access. The main contributions are summarized as follows:

- The compute popularity degree to find the files needed to replicate.

• And 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.

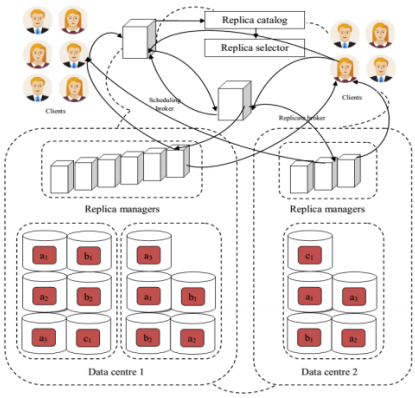

Fig : 1.1 Cloud data center

II. OVERVIEW OF DATA REPLICATION SYSTEM

Replication is used to advance system availability (by aiming traffic with a replica following a failure), prevent data reduction (by recovering lost files from a replica), and along with improve performance (by scattering load around multiple reproductions and by means of making low-latency access offered to users about the world). On the other hand, there are usually diverse ways to replication. Synchronous replication assures just about all copies are informed, but perhaps incurs excessive latency on updates. In addition, availability may be impacted in the event synchronously duplicated updates can't accomplish although some people might replicas are usually offline. Asynchronous replication excludes excessive write latency (in accurate, making the item appropriate pertaining to wide area replication) but permits replicas being stale. In addition, data loss normally takes place in the event an update is lost caused by breakdown just before it is usually replicated.

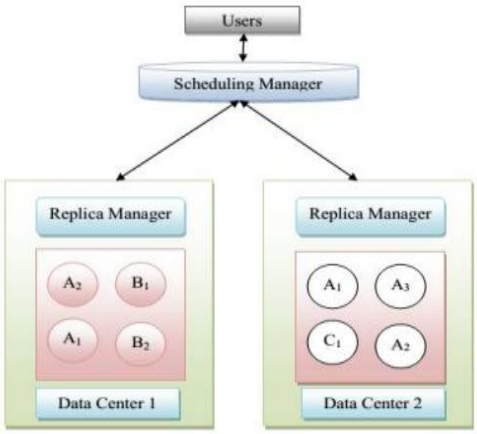

Fig : 2.1 Data replication system

III. FRAMEWORK

To introduce proposed replica placement solution, provide in this Section a motivating example to highlight some limitations of distributed storage systems. Let us consider a cloud system composed of two data centers (DC1 and DC2) located at different geographic regions and connected through a backbone network. It use HQFR to manage the storage distributed over the two data centers. It assume that have 4 partitions A, B, C and D with sizes 300 GB, 100 GB, 500 GB and 200 GB, respectively. Each partition has 4 replicas that are placed by the as unique as possible algorithm that strives to increase data availability. The initial mapping of the replicas across the infrastructure. For instance, the four replicas of partition A (denoted by A1, A2, A3 and A4) are distributed across the two data centers. The same applies to the other partitions. When a new data center is added to the infrastructure (i.e., DC3), replicas are relocated again according to the as-unique-as-possible algorithm used by HQFR [28]. The optimal locations of the replicas according to the as-unique-as possible algorithm. During this relocation, two issues could arise. Firstly, the amount of exchanged data to create the replicas could be huge and could overload the network. Secondly, the replicas that are not yet created or are in the process of being created are unavailable, and thus cannot process clients' requests. Indeed, the management tool that directs user requests to the appropriate locations of data should have an updated view of all replica placements. In HQFR, the new placement is used to direct clients' requests, even before the migration finishes. That is, the management tool becomes oblivious to the old placement of replicas. Therefore, some client requests may be directed to the new placement, even if some replicas haven't yet wholly arrived at their final destination, thus negatively impacting availability. Moreover, to ensure availability of data during migration, the management tool limits the number of migrating replicas of each data for a time interval. Indeed, a new placement of replicas is computed after 1-hour delay to move only one replica of each data in the respective interval, with the assumption that the availability of replicas will be ensured (i.e., all clients' requests will be accommodated).

HQFR algorithm:

As the name indicates High QoS First Replication algorithm. The main thing is that are considering the QoS requirement from the aspect of request information and its access time only. In HDFS the data is divided into 64MB data blocks. The replication factor is two in HDFS. There are two numbers of copies of data block other than the original one. And that two copies are stored on different Data Nodes or different data racks. And the Name Nodes keeps track of all the replicas other than original copy and the mounted on different data racks to avoid rack failure.

IV. EXPERIMENTAL RESULT

As the name indicates the applications with high QoS should be replicated first. According to knowledge the high QoS application have stricter requirements in the response time of a data access time than the normal applications. High QoS requirement application should take precedence over the low QoS requirement application to perform data replication. To sort all the applications according to their QoS requirement in a way, the application with high QoS should come first and then the lower one. If the data replication space is limited then first stores the data replicas of high QoS applications. When the high QoS application reads a corrupted data replica , its QoS requirements can be supported continuously by retrieving the data replica from high performance node.

Input: A set of requested nodes S_r.
Output: QoS-aware data replica placement.
1: Sort the requested nodes in S_r based on their associated access time.
2: for each requested node r_i in the sorted list of the requested nodes do
3: $S_q{}^{r_i} \leftarrow$ Find the corresponding qualified nodes of r_i using Eq. (1) and (2) to verify all storage nodes.
4: Select the r_f qualified nodes from $S_q{}^{r_i}$ which have smaller data replica access time than other qualified nodes in $S_q{}^{r_i}$.
5: for each selected qualified node q_j do
6: Store one data replica from r_i to q_j.
7: Update the available replication space of q_j.
8: end for
9: if $|S_q{}^{r_i}| < r_f$ then
10: $S_{uq}{}^{r_i} \leftarrow$ Find the corresponding un-qualified nodes of r_i only using Eq. (1).
11: Select the $r_f - |S_q{}^{r_i}|$ un-qualified nodes from $S_{uq}{}^{r_i}$ which have smaller data replica access time than other qualified nodes in $S_{uq}{}^{r_i}$.
12: for each selected un-qualified node uq_j do
13: Store one data replica from r_i to uq_j.
14: Update the available replication space of uq_j.
15: end for
16: end if
17: end for

While processing these replication request have to find the qualified node's list which helps to satisfy the QoS requirements of the appropriate application while running. The QoS requirement is given in the form of access time of that data block which is requested by an application. Note that while finding qualified node it should satisfy two conditions: • The requested node Ri and its qualified node Qj should not be mounted in the same rack. It should belong two different racks.

$$Rack(Ri) \neq Rack(Qj)$$

Where Rack() is the function to determine in which rack a node is located.

• The total data replica access time from qualified node Qj to request node Ri (T access(Ri,Qj))should be smaller than the QoS requirement of running application in Ri which is Tqos. T access (Ri,Qj) ≤ Tqos

After finding the qualified nodes by using these two conditions the data block can store its one data replica in each qualified nodes and the qualified nodes update their replication space respectively.

V. REPLICATION COST

A replica with high replication cost is not a suitable candidate for replacement because if the grid site needs that replica in the future, it should pay a high cost for replicating it again, and this is not economical. Therefore, the RPV will be greater if Replication Cost of that replica is high.

$$Replication\ cost = \frac{Size}{Bandwidth} * Propogation\ delay\ time$$

VI. CONCLUSION

Data replication has been widely adopted to improve data availability and to reduce access time. However, replica placement systems usually need to migrate and create a large number of replicas between and within data centers, incurring a large overhead in terms of network load and availability. In this concept, proposed HQFR, an efficient Replica migration scheme for distributed cloud Storage systems. HQFR complements replica placement algorithms by efficiently managing replica creation by minimizing the time needed to copy data to the new replica location while avoiding network congestion and ensuring the required availability of the data. 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, showed that proposed technique outperforms than the existing technique. An efficient scheme to solve the QoS aware problem in data replication. First algorithm is inspired from HQFR algorithm. This algorithm cannot give optimal solution to the QoS aware problem. So proposed another algorithm which gives optimal solution in polynomial time. This algorithm also helps in achieving both the objectives of which is minimization of replication cost and minimization of QoS violated data replicas. In future, going to find out an efficient energy optimization algorithm to energy consumption problem of nodes while performing data replication in cloud computing system.

REFERENCES

[1] 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.

[2] 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.

[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] 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.

[5] 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.

[6] 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

[7] 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.

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

[9] 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.

[10] 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.