Efficient Replica Migration Scheme for Distributed Cloud Storage Systems

[1]S.Malathi [2]Mr.N.Naveen Kumar

[1]M.Tech Student, School of Information Technology JNTUH, Kukatpally, Medchal-Malkajigiri, Hyderabad.

[2]Assistant Professor, School of Information Technology JNTUH, Kukatpally, Medchal-Malkajigiri, Hyderabad.

Abstract—with the wide adoption of large-scale internet services and big data, the cloud has become the ideal environment to satisfy the ever-growing storage demand. In this context, data replication has been touted as the ultimate solution to improve data availability and reduce access time. 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. In this paper, we propose CRANE, an efficient Replica migration scheme for distributed cloud Storage systems. CRANE complements any replica placement algorithm by efficiently managing replica creation in geo-distributed infrastructures in order to (1) minimize the time needed to copy the data to the new replica location, (2) avoid network congestion, and (3) ensure the minimum desired availability for the data. Through simulation and experimental results, we show that CRANE provides a sub-optimal solution for the replica migration problem with lower computational complexity than its integer linear program formulation. We also show that, compared to OpenStack Swift, CRANE is able to reduce by up to 60% the replica creation and migration time and by up to 50% the inter-data center network traffic while ensuring the minimum required data availability.

1. INTRODUCTION

With the wide adoption of large-scale Internet services and the increasing amounts of exchanged data, the cloud has become the ultimate resort to cater to the ever-growing demand for storage, providing seemingly limitless capacity, high availability and faster access time. Typically, cloud providers build several large-scale data centers in geographically distributed locations. They then rely on data replication as an effective technique to improve fault-tolerance, reduce end-user latency and

minimize the amount of data exchanged through the network. Consequently, effective replica management has become one of the major challenges for cloud providers [1]. In recent years, a large body of work has been devoted to address this challenge and, more specifically, to address the problem of replica placement considering several goals, such as minimizing storage costs, improving fault-tolerance and access delays [2], [3], [4], [5], [6]. However, replica placement schemes may result in a large number of data replicas created or migrated over time between and within data centers, incurring significant amounts of traffic exchange. This might happen in several scenarios requiring the creation and the relocation of a large number of replicas: when a new data center is added to the cloud provider's infrastructure, when a data center is scaled up or down, when recovering from a disaster or simply when replicas are relocated to achieve performance or availability goals.

2. RELATED WORK

[4] A dynamic data replication strategy using access-weights in data grids

Data grids deal with a huge amount of data regularly. It is a fundamental challenge to ensure efficient accesses to such widely distributed data sets. Creating replicas to a suitable site by data replication strategy can increase the system performance. It shortens the data access time and reduces bandwidth consumption. In this paper, a dynamic data replication mechanism called Latest Access Largest Weight (LALW) is proposed. LALW selects a popular file for replication and calculates a suitable number of copies and grid sites for replication. By associating a different weight to each historical data access record, the importance of each record is differentiated. A more recent data access record has a larger weight. It indicates that the record is more pertinent to the current situation of data access. A Grid simulator, OptorSim, is used to evaluate the performance of this dynamic replication strategy. The simulation results show that LALW successfully increases the effective network usage. It means that the LALW replication strategy can find out a popular file and replicates it to a suitable site without increasing the network burden too much.

In this paper, we propose a dynamic replication strategy called Latest Access Largest Weight. At constant time intervals, the dynamic replication algorithm collects the data access history, which contains file name, the number of requests for file, and

the sources that each request came from. Then these historical tables are given different weights according to their ages. By calculating the product of weight and the number of accesses for a file in different tables, we have a more precise metric to find out a popular file for replication. According to access frequencies for all files that have been requested, a popular file is found and replicated to suitable sites to achieve a system load balance. In order to evaluate the performance of our dynamic replication strategy, we use the Grid simulator OptorSim to simulate a realistic Grid environment. The simulation results show that the total job execution time of LALW is similar to LFU. However, LALW excels in terms of Effective Network Usage and storage usage.

[5] CDRM: A Cost-effective Dynamic Replication Management Scheme for Cloud Storage Cluster

Data replication has been widely used as a mean of increasing the data availability of large-scale cloud storage systems where failures are normal. Aiming to provide cost-effective availability, and improve performance and load-balancing of cloud storage, this paper presents a cost-effective dynamic replication management scheme referred to as CDRM. A novel model is proposed to capture the relationship between availability and replica number. CDRM leverages this model to calculate and maintain minimal replica number for a given availability requirement. Replica placement is based on capacity and blocking probability of data nodes. By adjusting replica number and location according to workload changing and node capacity, CDRM can dynamically redistribute workloads among data nodes in the heterogeneous cloud. We implemented CDRM in Hadoop Distributed File System (HDFS) and experiment results conclusively demonstrate that our CDRM is cost effective and outperforms default replication management of HDFS in terms of performance and load balancing for large-scale cloud storage.

In this paper, we design a cost-effective dynamic replication management scheme for large-scale cloud storage system refereed to as CDRM. We first build up a model to capture the relationship between availability and replica number. Based on this model, lower bound on replica reference number to satisfy availability requirement can be determined. CDRM further places replicas among cloud nodes to minimize blocking probability, so as to improve load balance

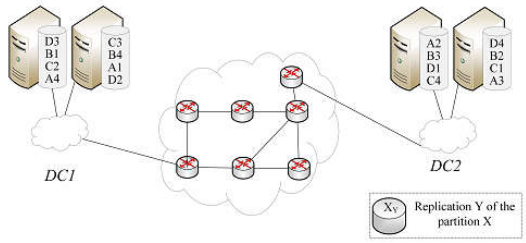
and overall performance. We implemented CDRM in HDFS and experiments demonstrate that CDRM can adapt to the changes of environment in terms of data node failure and workload changes and maintains a rational number of replica, which not only satisfies availability, but also improves access latency, load balance, and keeps the whole storage system stable.

3. FRAMEWORK

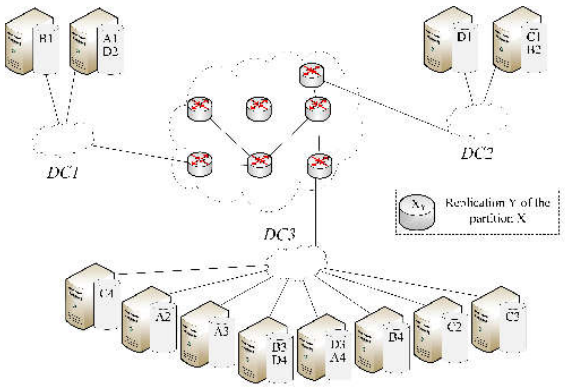
To introduce our proposed replica placement solution, we provide in this Section a motivating example to highlight some limitations of distributed storage systems. Let us consider a cloud system composed of two data centers (DC1 and DC2) located at different geographic regions and connected through a backbone network, as shown in Fig. 1(a). We use Swift to manage the storage distributed over the two data centers. We assume that we 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. Fig. 1(a) shows 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 Swift [28]. Fig. 1(b) shows 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 Swift, 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).



(a) Initial replica mapping with two data centers



(b) Final replica mapping after adding a third data center

4. EXPERIMENTAL RESULTS

We consider only 5 data centers. After that, two new data centers are connected to the infrastructure, which triggers the Swift placement algorithm in order to re-optimize the location of replicas. Then we use either Swift migration algorithm or CRANE algorithm to migrate replicas to the computed optimal locations. We deployed four scenarios considering 64 partitions with 3 replicas each one like recommended by some OpenStack providers [34]. And, we varied the average size of replicas. Table 6 depicts these scenarios.

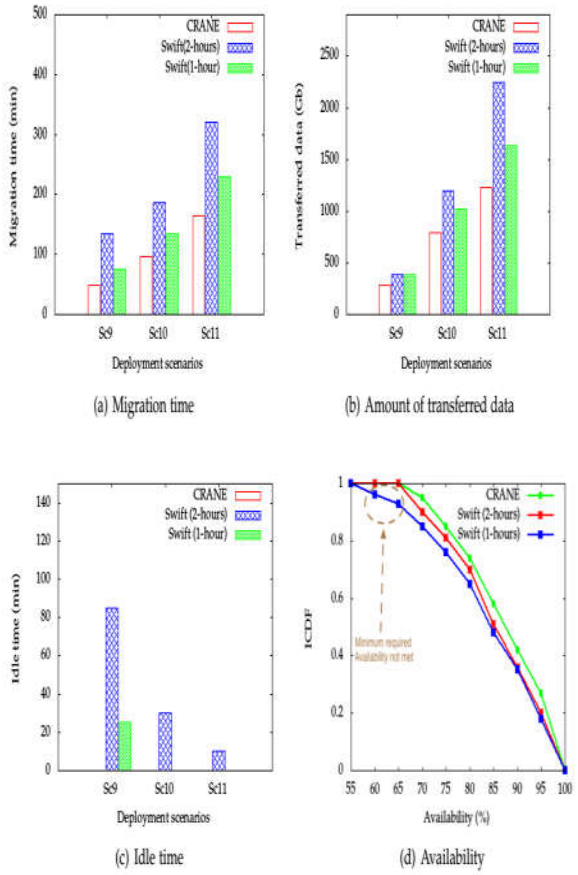


Fig.2.Experimental performance comparison between CRANE and traditional Swift

5. 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 paper, we proposed CRANE, an efficient Replica migration scheme for distributed cloud Storage systems. CRANE 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. In order to evaluate the performance of CRANE, we compare it to the optimal solution for the replica migration problem considering availability and to the standard swift, the OpenStack project for managing data storage. Results show that CRANE has sub-optimal performances in terms of migration time and an optimal amount of transferred data. Moreover, experiments show that CRANE is able to reduce up to 60% of the replica creation time and 50% of inter-data center network traffic and provide better data availability during the process of replica migration. In our future work, we will perform larger scale simulations to further scrutinize the performance of our heuristic. Other improvements will also consider addressing reliability and consistency requirements.

REFERENCES

[1] B. A. Milani and N. J. Navimipour, “A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions,” Journal of Network and Computer Applications, vol. 64, pp. 229–238, 2016.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, The Google file system. ACM, 2003, vol. 37, no. 5.

[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop distributed file system,” in IEEE Symposium on Mass Storage Systems and Technologies (MSST), 2010.

[4] 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, 2008.

[5] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, “CDRM: A cost-effective dynamic replication management scheme for

cloud storage cluster," in IEEE International Conference on Cluster Computing (CLUSTER),, 2010, pp. 188–196.

[6] 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.

[7] Y. Chen, S. Jain, V. Adhikari, Z.-L. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! datasets," in IEEE INFOCOM, April 2011, pp. 1620–1628.

[8] O. foundation. (2015) Swift documentation. [Online]. Available: http://docs.openstack.org/developer/swift/

[9] A. Mseddi, M. A. Salahuddin, M. F. Zhani, H. Elbiaze, and R. H. Glitho, "On optimizing replica migration in distributed cloud storage systems," in Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on. IEEE, 2015, pp. 191–197.

[10] (2016) IBM CPLEX Optimizer. [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/