Runtime Replica Consistency Mechanism For Cloud Data Storage

Mohammed Radi
Computer science department, Faculty of applied science
Alaqsa University Gaza
Moh_radi@alaqsa.edu.ps

Abstract
A cloud computing is becoming increasingly popular; cloud storage services attract more attentions for their high security and availability with a low cost. Cloud storage is expected to become the main force of the future storage market. As a key technology of cloud computing, replication faces new challenges, especially replica consistency. The intrinsic characteristic heterogeneous of cloud applications makes their consistency requirements different where the consistency requirement of certain application changes continuously at runtime. This paper presents a Runtime Replica Consistency Mechanism for cloud data storage to achieve a dynamic balance between consistency and performance. Evaluation result show that the propose mechanism guaranteeing the consistency and decrease the overhead.

***Keywords*—; replica consistency; cloud storage.**

1. Introduction

Cloud computing is becoming a very familiar word in industry and is receiving a large amount of attention from the research community.

Cloud storage is emerging as a powerful paradigm for sharing information across the Internet, which satisfies people's mobile data demand anywhere and anytime. Rather than relying on a few central large storage arrays, such a cloud storage system consolidates large numbers of geographically distributed computers into a single storage pool and provides large capacity, high performance storage service at low costs in unreliable and dynamic network environment [1].

Replication is one of the performance enhancing techniques for cloud storage system that has been widely used. Files are distributed across data nodes to achieve availability fault tolerance, scalability and performance. Unfortunately, the more replication increases the problem of inconsistent replicas. To solve the replica consistency problem, a replica consistency mechanism is needed. There are two traditional approaches that can generally be used as to how implement consistency management in large scale systems. The first approach is *lazy-copy based protocol (pull- based)* which transfers the updates from the original resource to the replicas only when accessing the replica. This way can save a lot bandwidth resource because update transferring occurs only when accessing to the replicas but it cuts down the availability of replicas and increase replicas access time. The second approach is *aggressive-copy based (push - based)* in which insures all replicas to be updated immediately by transferring, the updates to all replica once the original replica is updated. This way can grantee the availability of up-to date data all the time but the maintain cost increase significantly. Lazy based and aggressive based are only suitable for particular scenes.[2]

In the cloud computing environment the customer of cloud storage is homogeneous. Some application need lazy consistency and some need aggressive consistency, and the consistency requirement is inconsistent and change at runtime. The mechanism for replica consistency should be suitable for different application and consider this consistency requirement changes at run time. [1,3,4,5]. This paper focus on introducing a Runtime Based Replica Consistency Mechanism (RBRC) for cloud storage to achieve a dynamic balance between consistency and performance.

The rest of the paper is organized as follows. In Section 2, we present the related works. Section 3 a runtime based replica consistency mechanism. the evaluation is presented in section 4 Finally , we conclude the paper in Section 5.

2. Related works

There are many consistency models proposed in the distributed systems and database literature. Common references in the DB literature include [6,7,8] Also In distributed systems, [9] is the standard textbook that describes alternative consistency models as well as their trade-offs in terms of consistency and availability. In data grid environment many replica consistency models have been proposed[2,10, 11, 12]. Our work extends these established models by allowing levels of consistency to be defined and adapting the consistency guarantees at runtime.

Strong consistency is expensive not just in the transaction cost, but also in terms of replicas' availablity and system's performance. Not all applications need strong consistency guarantees. However, eventual consistency may result in high penalty cost caused by false operations. Therefore, researchers pay attentions to the balance between consistency, availablity and performance.

Ximei Wang propose an application-based adaptive mechanism of replica consistency in cloud data storage they divide the consistency of applications into four categories according to their read frequencies and update frequencies, and then design corresponding consistency strategies. The results show that the mechanism decreases the amount of operations while guaranteeing the application's consistency requirement.[1]

Kraska proposes a strategy that system can switch the level of consistency between serializability and session consistency dynamically according to running condition in the cloud[4]. It divides the data into three categories, and treats each category differently depending on the consistency level provided. The consistency level will be changed accordingly while the data's impact changes continuously at runtime.

Islam proposes a tree-based consistency approach that reduces interdependency among replica servers by introducing partially consistent and fully consistent states of cloud databases. The tree is formed such a way that the maximum reliable path is ensured from the primary server to all replica servers [5].

Ruay-Shiuang proposes an adaptive replica consistency service for data grid[10]. The strategy treats replicas differently according to the access frequency during the initializing process. The original replica and first level replicas can be updated immediately. If access frequency exceeds a predefined threshold, the second level replicas are updated immediately, too. If not, the replica is only updated when it is accessed.

Dongmei Cao proposes an adaptive consistency model for grid according to access frequency[13]. Compared to [10], the most important improvement is allowing system to switch consistency level automatically at runtime.

Based on asynchronous aggressive update propagation technique, Radi, M propose a scalable replica consistency protocol to maintain replica consistency in data grid. In the propose protocol the high access weight replicas updated faster than the others.[12]

Ghalem compares pessimistic consistency with optimistic consistency, and combines these two existing approaches [2]. It divides replicas into several sites. Optimistic principals are used to ensure replica consistency within each site. Whereas, global consistency is covered by the application of algorithms inspired from the pessimistic approach. Some of the above researches don't allow the system to change consistency level automatically at runtime, so they can't achieve the dynamic balance between consistency, availability and performance. Some partition the consistency level continuously, so the switch transaction cost is high. And in some works, the metric is selected unilaterally, so it can't be the very representative for an application. In order to avoid the above problems, the adaptive consistency mechanism proposed in this paper is based on read frequency and update frequency. System can select a suitable strategy dynamically according to these two metrics at runtime.

3. Runtime consistency mechanism in cloud

3.1 Model structure

In this paper the management of replica adapts a single master nodes for each data item or file in which there exist single master copy which is the origin of the file and the other replicas are secondary replicas. Figure 1 show the overview of the system architecture.

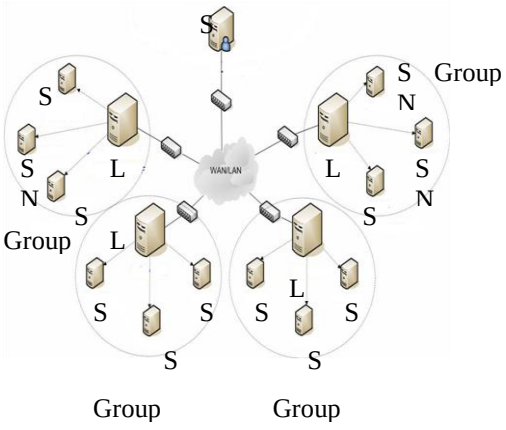

Figure 1: Over view of the system

We assume the SS, LS and SN are three main nodes of cloud sites that have more systems and storage resources. The super server (SS), where the original data are stored, can be modified by

end users through data intensive applications. Several replica nodes located closely are organized into one group. Each group has one server consider as Local Server (LS) and other nodes in the group is consider as a Secondary Node (SN). LS is responsible for the consistency service within its group. A LS is responsible for executes a corresponding operation according to the replica consistency mechanism. The SN's are the replica holders, one of the secondary node in each group will act as a LS node when the previous one breaks down.

In our model both LS and SN can receive a read request from end user and only A SS can be modified by end users and if a LS or SN receive update request it forward it to the SS to process it.

3.2 Replica consistency architecture

In order to provide the required functionalities of the single master replication, Replica Consistency Service (RCS) architecture is proposed. Local Replica Consistency Service (LRCS) and Replica Consistency Catalogue (RCC) are the main components of the architecture:

- Local Replica Consistency Service (LRCS): it is responsible for updating its local replica and relay the update propagation process if necessary.
- Replica Consistency Catalogue (RCC) is used to store the metadata inclides information of all SN including the physical poison, update and read frequencies.; this metadata will used by the RCS.

The interactions between the above components are shown in Figure 2.

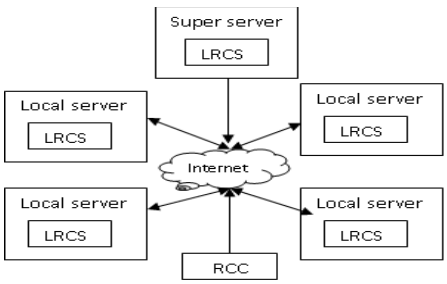

Figure 2: Replica Consistency Service Architecture

This interaction can be explained through a simple case user wishing to update a master file Fi. In a basic scenario, a user passes the update request and the target file to SS. The LRCS updates the local replica, and reflect a consistent view to its user, run the first step of the run time algorithm in which SS LRCS inquires the RCC about the replica LSs . After that SS send the update information to all LS. If the local server receives the update request it only passes this update request to the SS.details of the first step is shown in figure 3.

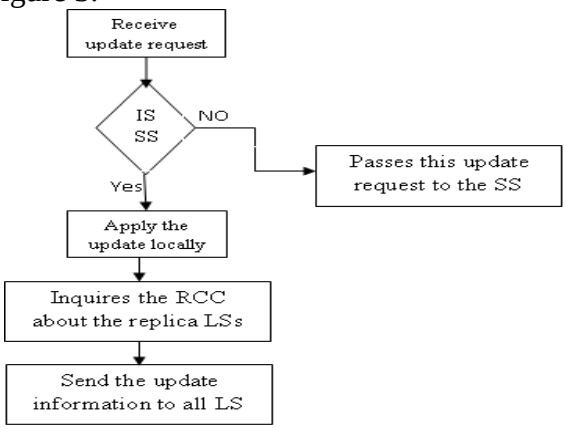

Figure 3: first step of the consistency algorithm

When the LS receive the update information, first it applies it at local replica and reflects the updates to its users, then it runs the second step of Runtime algorithm as show in figure 4. In the second step each local server inquires the RC about the read frequency of each replica in its group. Then it divides the replicas into two sup group depending on the read frequency. The replicas with high read frequency will be updated aggressively and the replicas with low frequency will be updated in lazy.

If the SS or the LS receive any read request from the user it directly allow the user to access the file, but if a SN receive a read request it first will check its read frequency, if the read frequency is high it directly allow the user to access the file but if the read frequency is low, then the LRCS pull the LRCS at LC for the last update, and it allow the user to access the replica only after it apply all the missing updates locally as shown in figure 5.

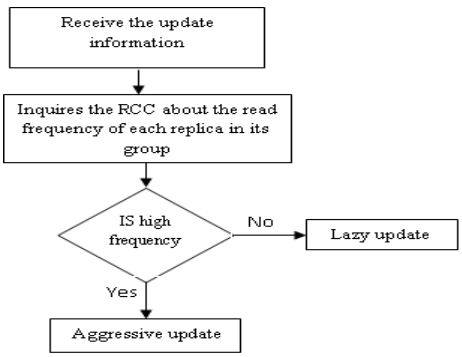

Figure 4: second step of the consistency algorithm

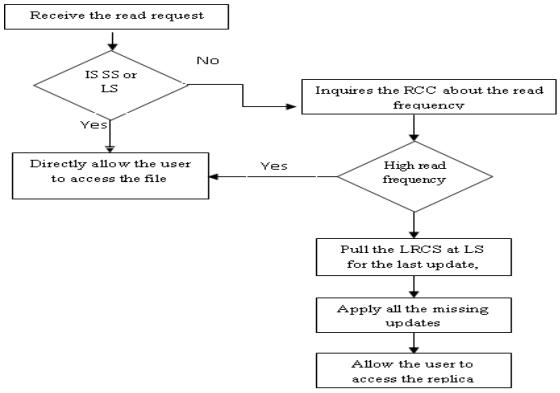

Figure 5: first step of the consistency algorithm

The runtime consistency mechanism divides the nodes into two categories according to the read frequency. The secondary nodes with high read frequency will follow the aggressive copy, and the low read frequency will follow the lazy- copy, while all master nodes will be updates in aggressive- copy. The ready frequency is classified as high if it exceeds a threshold value, and it classify as low if its less that a threshold value. Threshold value can be determined by cloud administrator.

4. Evaluation

We have implemented a runtime based replica consistency mechanism using OptorSim [14], a simulator for Data Grids. OptorSim was developed by the European Data Grid (EDG) project. It provides users with the Data Grids simulated architecture and programming interfaces to evaluate and validate their replication strategies. There are several critical omponents designed and implemented in OptorSim, including computing element (CE), storage element (SE), resource broker (RB), replica manager (RM), and replica optimister (RO), and so on. CEs and SEs are used to execute grid jobs and store files respectively.

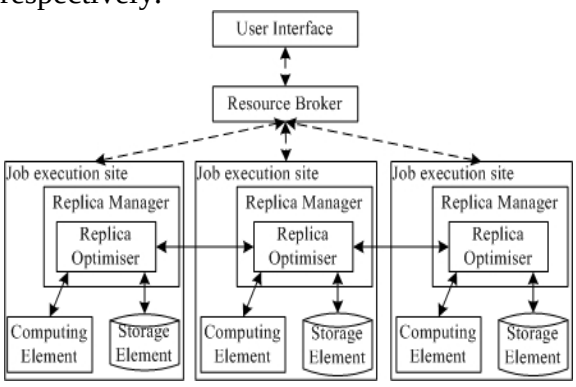

Figure 6. Basic architecture of OptorSim

In order to study the consistency we modify Optersim to satisfy our demand, and then compile our consistency mechanism on it. Each group is connected through the Internet. The Intra-region and inter-region network bandwidth are 1000Mb/sec and 500Mb/sec respectively.

Parameter	**Value**
Number of Jobs	500
Job Delay (ms)	25000
Max CE Queue size	200
File Processing Time (ms)	100000
Number of experiments	100
Each File Size (GBytes)	10
Number of Replica Modifications	100
Access Threshold	30

In order to study the runtime replica consistency mechanism we choose to compare our mechanism with lazy-copy based protocol aggressive-copy based in term of average file access time , number of replication, percentage of requesting up-to-date date . File access time is defined as the real time duration that a CE spends for accessing one file including file replication time and file processing time. The number of replication is the number of replications needed to run the replica consistency mechanism. The higher number of replications means the more file transmissions may be taken place. It may consume a considerable amount of network bandwidth. percentage of requesting up-to-date date is defined percentage that the application accesses up-to-date data in time interval τ to be the representative of consistency requirement of an application, and the overall update amount to be the representative of transaction cost.

Recall that in the mechanism with higher number of replications file transmissions may be taken place. Also It may consume a considerable amount of network bandwidth. Figure 7 represents the number of replications for the three consistency mechanisms. For lazy protocol the number of replication is very small and the aggressive-copy based protocol the number of replication is very high and it may waste too much network resources on invalid replications because some replicas may never be accessed. Compared with Aggressive-copy based protocol, our mechanism could lower the number of replications without wasting valuable network bandwidth. And it take not so much number of replication than lazy- copy mechanism.

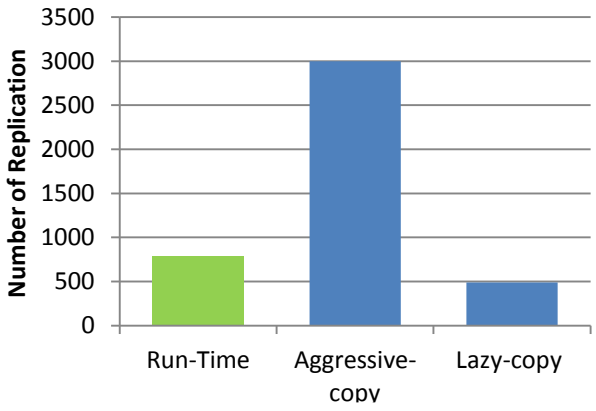

Figure 7: Number of replication

Figure 8 shows the comparison of the percentage that read a latest data every interval τ between lazy-copy, aggressive–copy and our consistency mechanism. Aggressive-copy almost guarantees that every access read the latest data. Lazy-copy mechanism guarantees weaker consistency, so the percentage is lower than strong consistency obviously. The run-time mechanism give a percentage in between the lazy-copy and aggressive –copy.

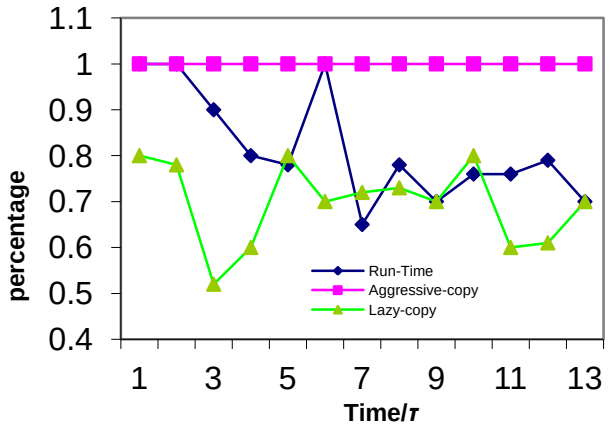

Figure 8: Percentage of accessing the up-to-date

The average file access time have evaluated as shown in Figure 9. Compared with the Lazy-copy based protocol, run-time can reduce file access delay time significantly because of our shorter replication time. As for Aggressive-copy based protocol, it copies the up-to-date replica in its region all along, therefore the file access delay time is equal to the file processing time without suffering form the long replication delay time due to the consistency problems.

Compared to aggressive copy mechanism, the run-time consistency mechanism proposed decreases number of replication significantly while the needs of application for consistency are mainly satisfied. And our consistency mechanism guarantees higher percentage of read a latest data than lazy-copy and decrease the average file access time. Consequently, we get a better balance between consistency and performance.

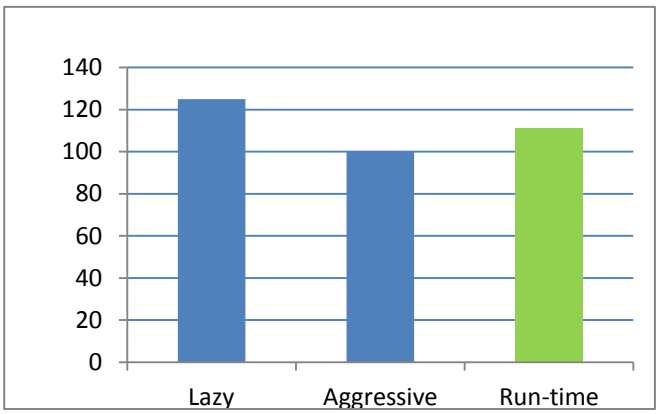

Figure 9: .Average file access time

5. Conclusion

This paper presents a Runtime Replica Consistency Mechanism for cloud data storage aiming achieve a dynamic balance between consistency and performance. The runtime consistency mechanism divides the nodes into two categories according to the read frequency. The mechanism maintain the replica consistency of some nodes in an aggressive way and some other node in a lazy way according to the read frequency. Evaluation result show that the propose mechanism guaranteeing the consistency and decrease the overhead

References

[1] Qingsong Wei, Bharadwaj Veeravalli, Bozhao Gong,Lingfang Zeng, Dan Feng, CDRM: A Cost-effective Dynamic Replication Management Scheme for Cloud Storage Cluster. 2010 IEEE International Conference on Cluster Computing, 188 – 196.

[2] Ghalem, B, S. Yahya. A Hybrid Approach for Consistency Management in Large Scale Systems. Proceedings of the International conference on Networking and Services, Page(s): 71

[3]Ximei Wang, Shoubao Yang, Shuling Wang, et, al, An Application-Based Adaptive Replica Consistency for Cloud Storage. 2010 Ninth International Conference on Grid and Cloud Computing.

[4]Kraska, T, M. Hentschel, et al. Consistency Rationing in the Cloud: Pay only when it matters. Proceedings of the VLDB Endowment, 2009, 2(1): 253-264.

[5]Islam, M.A.; Vrbsky, S.V.Tree-Based Consistency Approach for Cloud Databases Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on Nov. 30 2010-Dec. 3 2010, 401 - 404

[6] P. Bernstein, V. Hadzilacos, and N. Goodman. Concurrency Control and Recovery in Database Systems. Addison Wesley, 1987

[**7]** M. T. Ozsu and P. Valduriez. Principles of Distributed Database Systems. Prentice Hall, 1999.

[**8]** G. Weikum and G. Vossen. Transactional Information Systems.

[9] A. Tanenbaum and M. van Steen. Distributed Systems: Principles and Paradigms. Prentice Hall, 2002.

[10] Ruay-Shiung Chang, Jih-Sheng Chang. Adaptable Replica Consistency Service for Data Grid. Third International Conference on Information Technology: New Generations (ITNG'06). 2006.

[11] Cao DongMei. The Research of Replica selection and consistency for Grid[D]. WuHan: Huazhong University of Science and Technology,
2007.

[12] Mohammed Radi ; Ali Mamat ; M. Mat Deris ; Hamidah Ibrahim ; Subramaniam Shamala, Access Weight Replica Consistency Protocol For Large Scale Data Grid, Journal of Computer ScienceYear: 2008 Volume: 4 - Issue: 2

[13] Cao DongMei. The Research of Replica selection and consistency for Grid[D]. WuHan: Huazhong University of Science and Technology, 2007.

[14] OptorSim: A Replica Optimiser Simulation.
http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html.