Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

Cloud Storage Service with Auditing Cloud Consistency

Sruthymol K.S[1], Aswathy R[2]
[1,2] *Department of Computer Science, IIET, Nellikuzhi*

Abstract—Cloud storage service maintains and manage large amount of data and makes that data accessible over a network, usually the internet. Because of this enormous advantages cloud storage services have become very popular. To provide always-on access all over the world, a cloud service provider (CSP) maintains multiple replicas of data on multiple geographically distributed servers. One of the main problem of this replication technique is that it is very difficult and expensive to achieve strong consistency on a world wide scale. So in this paper, present a novel consistency as a service (CaaS) model. The CaaS model consists multiple audit clouds and one large data cloud. The data cloud is controlled by the CSP and the small audit cloud contains multiple users and they can able to verify whether the data cloud provides the promised consistency or not. Here propose a two level auditing architecture to verify the consistency. If the data cloud is not provide the promised level of consistency, then a gossip based system is used to avoid the consistency to a maximum level. One more method is also used to avoid the consistency to a maximum level, a locking protocol.

Keywords— Cloud storage, consistency as a service (CaaS), two level auditing

I. INTRODUCTION

Now a days cloud computing has become very popular, because it promises high scalability, elasticity, availability at low cost. One of the most important service in cloud computing is cloud storage service, which deliver the data storage as a service, including network attached storage and database-like services. Most important examples are Amazon simpleDB, Microsoft Azure storage and so on. The users can access data stored in a cloud anytime and anywhere with any device by using the cloud storage service, without caring about the capital investment. To provide the always-on access all over the world, the cloud service provider (CSP) stores the replicas of data on multiple servers. One of the main problem of using the replication technique is that it is very difficult to achieve strong consistency on a world wide scale. But many CSPs only provide eventual consistency, such as weak consistency for high availability and performance, in this case a user can read old version of data for a period of time. One of the most popular application that provide eventual consistency is domain name system (DNS). Changes to the names will not be visible immediately, but eventually all clients see them.

For many applications, especially for the interactive applications , stronger consistency assurance is required. Consider an example, suppose that Alice and Bob using a cloud storage service and both are cooperating on a project. There are five cloud servers, CS_1,….,CS_5. The data is replicated to this five cloud servers. After updating the requirement analysis to CS_4, Bob calls Alice to download the latest update of requirement analysis. Here, after Bob calls Alice there exist a causal relationship between Bob's update and Alice's read. So Alice can able to download the latest version of requirement analysis from the cloud server CS_5. That means the data cloud should provide causal consistency, which ensures that Bob's update is replicated to all of the cloud servers before Alice read the update. If the cloud provide only eventual consistency, then Alice is allowed to access an old version from CS_5. In this example, if the data cloud provides only eventual consistency, that not satisfy the requirements of customers.

Actually, different applications require different consistency levels. For example, social network services require causal consistency, mail services require read-your-write consistency and monotonic-read consistency. In cloud storage , for consistency not only need to consider correctness but also need to consider the actual cost per transaction. So in this paper, present a novel consistency as a service (CaaS) model. The CaaS model consists of multiple audit clouds and one large data cloud. The audit cloud consists of a group of users, they cooperate on a job and the data cloud is maintained by a CSP. There will be a service level agreement between the data cloud and the audit cloud, which will include the consistency level provided by the data cloud, and how much will be charged if violates the SLA. But for the users it is very difficult to verify whether the data cloud provide the promised level of consistency or not. In this paper, the audit cloud is allowed to verify cloud consistency by analyzing a trace of operations, a loosely synchronized clock is used for this solution. Here each user need to maintain a logical vector and physical vector for each operation. A two level auditing architecture is adopted here: local auditing performed by each user and the global auditing performed by an elected auditor. Local auditing focuses on read –your-write and monotonic-read consistencies. Global auditing focuses on causal consistency, which is performed by implementing a directed graph. If the graph is a directed acyclic graph (DAG), we ensure that causal consistency is preserved.

II . RELATED WORKS

A cloud is a large-scale distributed system, the cloud service providers (CSP) are responsible for keeping the data available and accessible at any time. So to achieve high availability and high performance each piece of data is replicated on multiple geographically distributed servers in cloud. Thus, first review the consistency models in distributed systems. Ref. [2], which proposed two types of consistency models: data-centric consistency and client-centric consistency. Data-centric consistency model includes the internal processing of a storage system, i.e., how updates flow through the system and what guarantees are provided with respect to updates. The client-centric consistency model considers the customers point of view, that means a customer not need to bothered about whether a storage system internally contains any stale copies or not. Therefore the client-centric consistency model concentrates on customers satisfaction. This work also describes different consistency levels in distributed systems, from strong consistency to weak consistency. If the consistency is increased then the cost is also increased and the availability is reduced. In reality, many distributed systems sacrifice strict consistency for high availability.

Then review the work that describes different levels of consistency in a cloud. Ref. [3] describes various consistency properties provided by commercial clouds. Existing commercial clouds only provide weak consistency guarantees to stored data. In Ref. [4], describes depending on the availability of data the consistency requirements are vary over time. In this work the authors provide techniques that make the system dynamically adapt to the consistency level according to the state of the data. This work proposed a consistency rationing approach. In this approach data is divided into three consistency categories: A, B, and C. The A category ensures strong consistency guarantees, the C category ensures session category and the B category is handled with either strong or session consistency depending on the data. Ref. [5] proposed a novel consistency model, so automatically adjust the consistency levels for different data. Finally review both Ref. [6] and Ref. [7], describes benchmark-based verifications. That means focus on benchmarking staleness in a storage system. Both work evaluated consistency in Amazon's S3, but different results are provided.

III. PROPOSED WORK

A. Consistency as a Service (CaaS) model

The proposed method is shown in Figure 1, which illustrate a consistency as a service (CaaS) model, which includes two audit clouds and one data cloud. The data cloud maintained by the cloud service provider (CSP), each piece of data is replicated on geographically distributed multiple cloud servers. An audit cloud consist of multiple users that cooperate on a job. Each user is identified by a unique ID. The data cloud and audit cloud engage in a service level agreement (SLA), which state clearly the consistency level promised by the data cloud. The audit cloud can able to verify whether the data cloud violates the SLA or not.

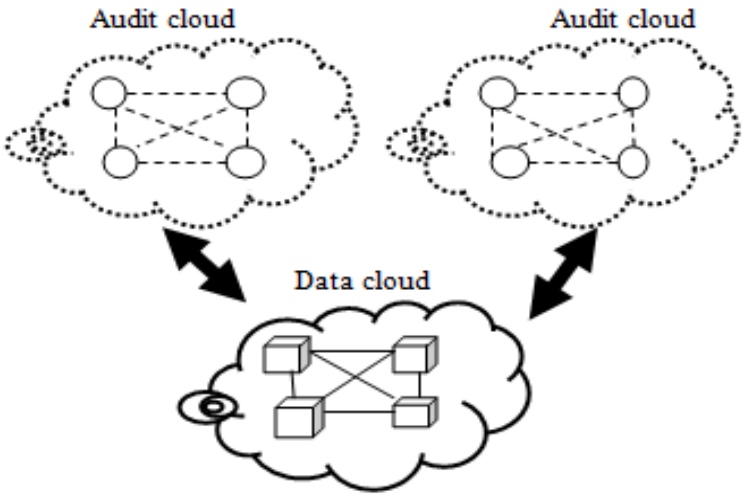

Fig 1 Consistency as a service model

B. User Operation Table

A user operation table (UOT) is maintained by the user for recording local operations. This records in UOT includes three elements: operation, logical vector, physical vector. The physical clock in the physical vector continuously increased by the user and the logical clock in the logical vector is increased by the user only when an event happens. An auditor is elected from the audit cloud, then all other users send their UOTs to the auditor, which will perform the global auditing.

C. Two-Level Auditing Structure

In CaaS model a two-level auditing structure is used to verify the consistency properties. In the first level, using the UOT each user independently performs local auditing. The following consistencies are verified in this level:

Monotonic-read consistency: a user must read either the same value or a newer value.

Read-your-write consistency: a user always reads his latest updates.

A local consistency auditing algorithm is used to verify both the above consistencies, which is an online algorithm.

Local Consistency Auditing Algorithm:

- Initialize UOT with ϕ

- Op1 = W (a) is the first operation, then record W (a) in UOT
- W (b) is the last write in UOT
- If op2 = r (a), then Read-your-write consistency is violated
- If r (a) is the current read, r (c) is the last read in UOT
- Then W (a) happens before W (c), Monotonic-read consistency is violated.

In the second level an elected auditor can perform global auditing, which is performed after obtaining a global trace of all users operations. In this level causal consistency should be verified:

Causal consistency: all causally related read/write operations were executed in an order that reflects their causality, all concurrent operations may be seen in different orders.

A global consistency auditing algorithm is used to verify the causal consistency, which is an offline algorithm.

Global Consistency Auditing Algorithm:

- In global trace each operation is denoted by a global trace
- Two operations op1 and op2, if op1 happens before op2 then a time edge is added from op1 to op2
- Two operations op1 and op2, if op1 = W (a) and op2 = R (a), two operations are from different users then a data edge is added from op1 to op2
- If two operations op1 = W (a), op2 = W (b), from different users, and W (a) is on the route from W (b) to R (b) then a causal edge is added from op1 to op2.
- Topological sorting is used to check whether the graph is a directed acyclic graph (DAG) or not.

To observe whether a directed graph is a DAG or not, topological sorting is performed on the graph. Topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering.

D. Gossip Based System

If the topological sorting is not possible, we can able to understand that the graph is not a DAG. From this the users can verify that the data cloud not provide the promised level of consistency specified in the SLA. In this situation to avoid the consistency to a maximum level a gossip based system is used. In this system a gossip time protocol is implemented in all the system. So one user can able to know the state of many other users, through this it is possible to avoid concurrent operations, thereby reduce the conflict, also able to reduce inconsistency to a maximum level.

- User1 initializes gossiping by sending a request message to user2
- After the reception of a request by user2, it sends a response back to user1
- User1, based on the response, may either synchronize its clock to the clock of user2, send a feedbackmessage to user2, or ignore the message.
- If user2 receives the feedbackmessage, it may use it to synchronize its clock to the clock of user1.

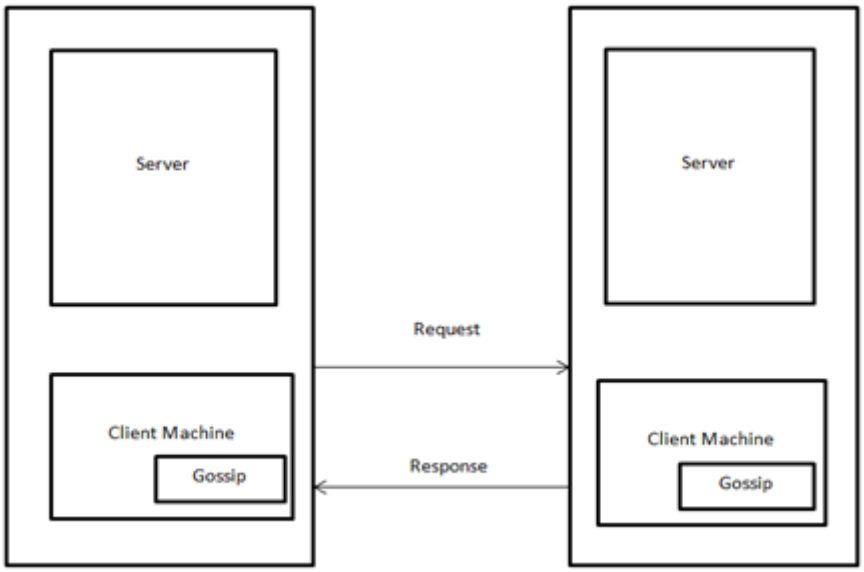

Fig 2 Cloud storage system with gossiping time protocol

E. A Locking Protocol

Another method for avoiding consistency to a maximum level is a locking protocol. In database and transaction processing, two phase locking (2PL) is a concurrency control method that guarantees serializability. It is also the name of resulting set of database transaction schedules. The protocol utilizes locks, applied by a transaction to data, which may block other transactions from accessing the same data during the transaction's life.

A lock is a system object associated with a shared resource such as data item of an elementary type, a raw in a database or a page of memory. In a database, a lock on a database object may need to be acquired by a transaction before accessing the object. Correct use of locks prevents undesired, incorrect or inconsistent operations on shared resources by other concurrent transactions. When a database object with an existing lock acquired by one transaction needs to be accessed by another transaction, the existing lock for the object and the type of the intended access are checked by the system. If the existing lock type does not allow this specific attempted concurrent access type, the transaction attempting access is blocked. Thus, with a locking mechanism, needed operation blocking is controlled by a proper lock blocking scheme. If one user lock the item, then another users can not able to update the item until the user unlock the item. So in this case another users can only able to read the item.

IV. RESULTS

In this section compare the distributed system without gossiping time protocol (old) and the distributed system with gossiping time protocol (new). The proposed method is experiment through JAVA running on a local machine.

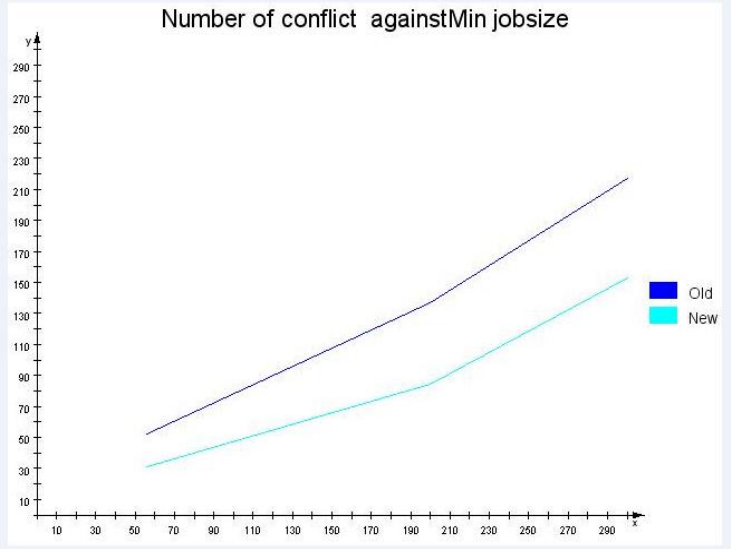

Fig 3 Comparison of cloudstorage system without gossiping and with gossiping

V. CONCLUSION

In this paper, presented a consistency as a service (CaaS) model and a two level auditing structure to help users verify whether the data cloud maintained by the cloud service provider (CSP) is providing the promised level of consistency or not. Two consistency auditing algorithms are presented to check different types of consistency properties. A gossip based system is used to avoid the consistency to a maximum level. Through the CaaS model with gossiping time protocol the users can determine the quality of cloud services and select a right CSP from various CSPs, and also avoid the inconsistencies (not completely) with gossip based system. A locking protocol is also used to avoid the consistency to a maximum level, so it is possible access the data with high consistency.

REFERENCES

[1] W. Golab, X. Li, and M. Shah, "Analyzing consistency properties for fun and profit," in Proc. 2011 AC PODC.
[2] A. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms. Prentice Hall PTR, 2002.
[3] "Eventually consistent," Commun. ACM, vol. 52, no. 1, 2009
[4] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, "Consistency rationing in the cloud: pay only when it matters," in Proc. 2009 VLDB.
[5] S. Esteves, J. Silva, and L. Veiga, "Quality-of-service for consistency of data geo-replication in cloud computing," Euro-Par 2012 Parallel Processing, vol.7484, 2012.
[6] H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu, "Data consistency properties and the trade-off in commercial cloud storages: the consumers' perspective," in Proc.2011 CIDR.
[7] D. Bermbach and S. Tai, "Eventual consistency: how soon is eventual?" in Proc. 2011 MW4SOC.
[8] M. Rahman, W. Golab, A. AuYoung, K. Keeton, and J. Wylie, "Toward a principled framework for benchmarking consistency," in Proc. 2012 Workshop on HotDep.
[9] D. Kossmann, T. Kraska, and S. Loesing, "An evaluation of alternative architectures for transaction processing in the cloud," in Proc. 2010 ACM SIGMOD.
[10] A. Aiyer, L. Alvisi, and R. Bazzi, "On the availability of non-strict quorum systems," Distributed Computing, vol. 3724, 2005.