Implementation of Consistency as a Service: Global Auditing for Cloud Consistency

[1]Priti Garud, [2]Prof. S.P. Vidhate

[2]HOD of Computer Engineering Dept.,VACOE,Ahmednagar

[1]*Vishwabharti Acadamy's College of Engineering, Ahmednagar,Maharashtra*

ABSTRACT

Currently, cloud is the most important part of our life. Cloud is most popular because of huge advantages such as it is portable we can able to access the cloud anywhere globally and it is used in different business purpose. Duplication technology is used to reduce storage space so that cloud service provider maintains much duplication and each piece of data are globally distributed on servers. The main issue of cloud is to handle duplication of data which is too costly to achieve powerful consistency on worldwide. In this paper we present a novel consistency service model which contains a large amount of data cloud and multiple audit clouds In the Consistency Service model. A data cloud is maintain by CSP which is stands by Cloud service Provider and the number of user constitute group and that group of user can constitute an audit cloud Which can check whether the data cloud provides the valid level of consistency or not we suggest the global auditing architecture, and this architecture requires a tightly synchronize clock for updating and download operation in the audit cloud. While loosely synchronize clock for upload operation. Then, performed global auditing by global trace of operation through randomly electing an auditor from audit cloud. Finally, we use a heuristic auditing strategy (HAS) to reveal as many violations as possible.

KEYWORDS :-*Cloud Storage, Global Consistency Auditing, Local Consistency Auditing, heuristic auditing strategy (HAS).*

1. INTRODUCTION

Cloud storage service has become more accepted due to their very huge advantages and when we say specific style of computing where everything from computing power to infrastructure business apps are provided as a service its computing service is known by cloud computing rather than product some other benefits of cloud is resource provisioning scalability, flexibility and low cost. Amazon DB, Microsoft Azure Storage DB are the e.g. Of cloud company which gives cloud services as per month or yearly basis and by using cloud storage services the customer can able to access data store any where anytime by using any device and no need of capital investment on hard-ware and access your data any time. The major problem in cloud is to handle dummies copy it is too costly to achieve strong consistency worldwide. Many cloud service provider (CSP) uses weak consistency such as eventual consistency to achieve good performance and high availability the user can able to see latest update by using ACP principle (Availability consistency and partition.) The famous popular example of eventual consistency is Domain Name System. Eventual consistency is not remedy for all difficulty for all application e.g. for interactive service the strong consistency is required. Show the figure 1 for all details regarding systems. Suppose Alice and bob are work under cloud storage service project. The data is replicated to 5 servers CS1, CS2, CS3, CS4 and CS5 respectively uploaded the latest version of the requirement analysis to CS4 Alice call bob to download latest version so here causal relationship is establish between bob read and Alice update. If the Cloud only provides eventual consistency then bob gives the permission to access old version from CS5. So from this we can say different application has different consistency by the following e.g.

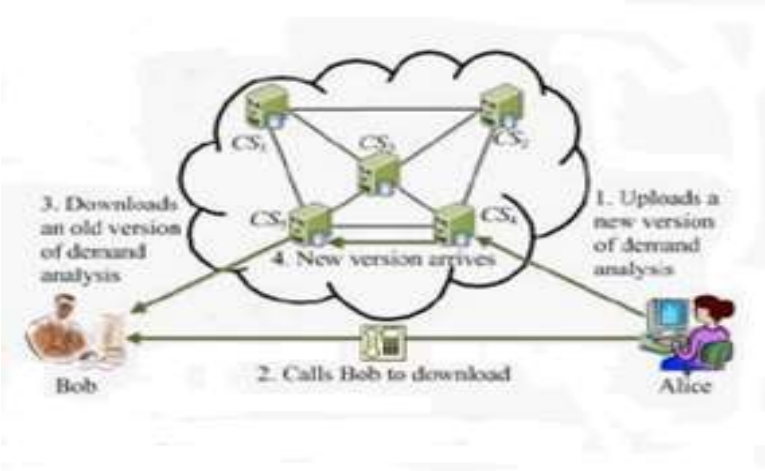

Figure -1: An Application that requires causal consistency

1) Monotonic read consistency while mail server has read your write consistency
2) Social networking services are the example of causal consistency. consistency plays important role in the cloud storage to determines correctness as well as actual cost/transaction But here we shows novel consistency service model for this situation this consistency service module contain multiple small audit cloud and large data cloud. Audit cloud contain a group of users that working on the project and service level agreement will be form between audit cloud & data cloud while Cloud service provider maintain data cloud, which will take how much will be charged if the data cloud failed to SLA and what type of consistency the data cloud should provide the implementation of data cloud is not visible to all users due to virtualization technique.

To find out each replica in data cloud is newest one or not very difficult to users. We permit the user in audit cloud to check cloud consistency by analyzing the trace interactive operation. We don't require a global clock among all users for total ordering of operation so we use loosely synchronized clock for our solution. For partial order of operation each user maintain logical vector. So here we develop 2 level of Auditing Structure.

1. Local Auditing
2. Global Auditing

Local Auditing: The Local Auditing focuses on monotonic read and read your write consistency. This can be performing by light-weight online algorithm the local auditing algorithm is online algorithm.

Global Auditing: This auditing focuses on causal consistency because causal consistency performs by constructing directed graph. The directed acyclic graph is constructed then causal consistency is obtained. Finally we propose analytical auditing strategy which applicable reads to reveal many unsuccessful results.

2. LITERATURE SURVEY

1. Don't settle for eventual: scalable causal consistency for wide-area storage with COPS:

Geo-replicated, distributed data stores that support complex online applications, such as social networks, must provide an "always-on" experience where operations always complete with low latency. Today's systems often sacrifice strong consistency to get these goals, exposing inconsistencies to their clients and necessitating complex application logic. In this system, this system identifies and defines a consistency model causal consistency with convergent conflict handling, or causal that is the strongest achieved under these constraints.

This system present implementation and the design of COPS, a key-value store that delivers this consistency model across the wide-area. A key contribution of COPS is its scalability, which can enforce causal dependencies between keys stored across an entire cluster, rather than a single server like as previous systems. In COPS- GT, this system introduces get transactions in order to obtain a consistent view of multiple keys without blocking or locking. The assessment shows that all operations are completed in a millisecond, provides throughput

similar to previous systems when using one server per cluster, and scales well as this system increase the number of servers in each cluster.

2. Axioms for memory access in asynchronous hardware systems:

Misra [2] is the first to present an algorithm for verifying whether the trace on a read/write register is atomic. Following his work, Ref. [3] proposed offline algorithms for verifying whether a key-value storage system has safety, regularity, and atomicity properties by constructing a directed graph. He presented an elegant algorithm for checking atomicity. The algorithm proposed by [2] works by reasoning about the values of the register. The observation is that, at some point during the span of an operation, the register assumes the value of the operation, (either write or read). The system stipulates that a newly added value instead of the place of old value is not allowed to re-appear in future. Therefore, if a trace violates this condition, then it is not atomic. Some- what surprisingly, if a trace does not violate this condition, then it is atomic. In contrast, our algorithms reason about the operations. We choose to reason about operations but not values because we aim to provide a common framework to check a variety of semantics, many of which (e.g., regularity and safety) were introduced after Misra's paper. It is not immediately clear to us how to extend Misra's algorithm to check, say, regularity, because for regularity, a replaced value is allowed to re-appear.

3. Two Level Auditing Architecture to Maintain Consistent In Cloud:

Confidential data in an enterprise may be illegally accessed through a remote interface facilitated by a multiple cloud, or relevant data and archives may be lost or tampered, in the condition of outside storage .So that it is necessary for CSPs to provide security techniques for managing their storage services. To overcome these problems this system presents a Consistency as a service auditing cloud scheme. This system proves the security of my scheme based data fragmentation on multiple clouds. The proposed system has data security, data fragmentation and storage on multiple cloud services. Therefore, a trusted third party is used to store the data on multiple cloud and find the data access by un-trusted cloud service providers. The system proposes, client data divided into various multiple pieces and send to the multiple clouds with help of trusted or believable third party. If any of the un-trusted cloud service providers try modify the data the alert will send to trusted third party about illegal access of un-trusted cloud service provider.

4. What consistency does your key-value store actually provide?:

In this proposed system, where they record lengthy traces with interleaved operations, and after the fact they check for cycles in several conflict graphs to find whether various properties hold. The properties they analyze are those that are important in parallel hardware design, such as safe registers or regular, rather than the properties usual in cloud storage platforms such as eventual consistency with monotonic reads. There is also work on formally defining weak consistency properties. Usually eventual consistency is described in terms of internal properties such as the state of the replicas.

5. Analyzing Consistency Properties for Fun and Profit:

Motivated by the increasing reputation of eventually consistent key-value stores as a commercial service. This system address two important problems related to the consistency properties in a history of operations on a write/read register (i.e., finish time, start time, argument, and response of every operation). First, this system considers how to detect a consistency violation as soon as one happens. This system proposed a specification for online verification algorithms, and this system presents such algorithms for several well-known consistency properties. Second, this system considers how to quantify the severity of the violations, if a history is found to contain consistency violations. This system investigates two quantities: first is the staleness of the reads, and the second is the commonality of violations. For staleness, this system further considers time-based staleness and operations-count-based staleness. These system present efficient algorithms that compute these quantities. This system believes that addressing these problems helps both key-value store providers and users adopt data consistency as an important aspect of key-value store offering.

6. Timestamps in Message-Passing Systems That Preserve the Partial Ordering:

Time stamping is a common method of totally ordering events in concurrent programs. However, for applications requiring access to the global state, a total ordering is inappropriate. This paper presents algorithms for times tamping events in both asynchronous and synchronous message passing programs that allow for access to the partial ordering inherent in a parallel system.

7. Distributed Systems: Principles and Paradigms:

A cloud is essentially a large-scale distributed system where each piece of data is replicated on multiple geographically distributed servers to achieve high performance and high availability. Thus, we first review the consistency models in distributed systems. Ref. [5], as a standard book, proposed two classes of consistency models: one is data-centric consistency and the other is client-centric consistency. Data-centric consistency model considers the internal state of a storage system, i.e., how updates goes through the system and what guarantees the system can furnish with respect to updates. However, to a customer, it really does not matter whether or not a storage system internally include any stale copies. As long as no stale data is observed from the client's point of view, the customer is satisfied. Therefore, client-centric consistency model concentrates on what specific customers want, i.e., how the customers inspect data updates. Their work also describes different levels of consistency in distributed systems, from strict consistency to weak consistency. High consistency implies reduced availability and high cost. Ref. [6] states that strict consistency is never needed in practice, and is even considered harmful.

3.PROPOSED SYSTEM

We show a novel consistency as a service (CaaS) model[1] , where a group of users that constitute an audit cloud can verify whether the data cloud provides the promised level of consistency or not. The Consistency as a service model consists of large data cloud and various audit cloud .A service level agreement (SLA) will be busy between the data cloud and the audit cloud ,which will tell what level of consistency the data cloud must provide, and how much will be charged if the data cloud violates the service level agreement .

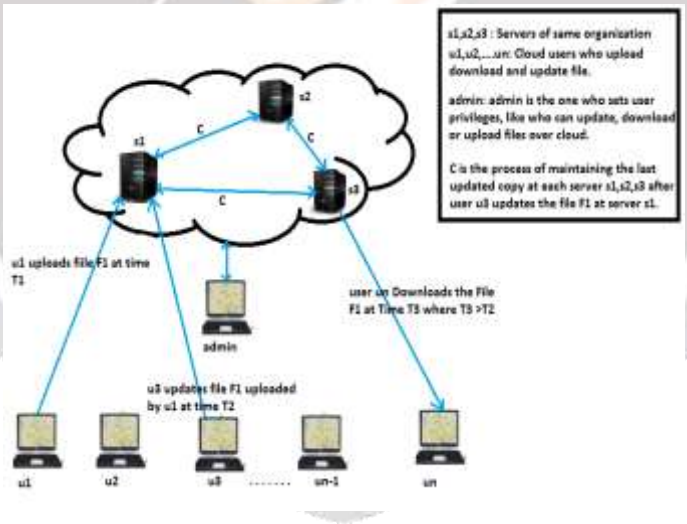

Figure -2: Proposed model

global auditing concentrate on Causal consistency.

Causal consistency: - Writes that are causally related should be seen by all processes in the similar order. Concurrent writes may be seen in a dissimilar order on different machines.

Global auditing concentrate on casual consistency, which is performed by constructing a directed graph. If the constructed graph is a directed acyclic graph then casual consistency is preserved. Specify the severity of violations can be done by two metrics for the CaaS model: commonality of violations and staleness of the value of read. Finally it was proposed a heuristic auditing strategy (HAS) which adds appropriate reads to reveal as several violations as possible.

4. MATHEMATICAL MODEL

System Description:

User Authentication

Set (C) = {c0, c1, c2, c3}

C0= Get User Id

C1=Get Cloud Id.

C2=Get Data Owner Info

C3=get the User Privilege Information

C4= Get key from hash table.

Encryption

Set (E) = {e0, e1, e2, c1, c2}

e0=get file to be encrypted

e1=get public key for encryption

e2=encryption of data.

Consistency check

Set (D) = {d0, d1, d2}

d0= check global status

d1= maintain consistency status

d2= check out

Service module

Set (S) = {s0, s1, s2, d0, d1}

s0=get user id and file request

s1=get data to be uploaded or downloaded

s2=provide service

Union and Intersection of project

Set (E) = {e0, e1, e2, c1, c2}

Set (D) = {d0, d1, d2}

Set (S) = {s0, s1, s2, d0, d1}

(C U E)= {c0,c1,c2, c3,c4, e0, e1, e2}

(C ∩ E)= {c1, c2}

(D U S)= {d0, d1, d2, s0, s1, s2}

(D ∩ S) = {d0, d1}

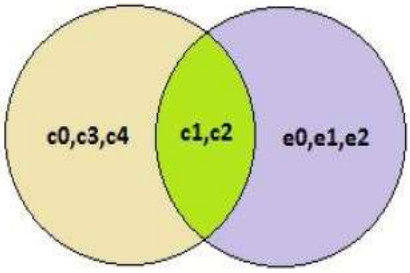

Figure -3 : C intersection E

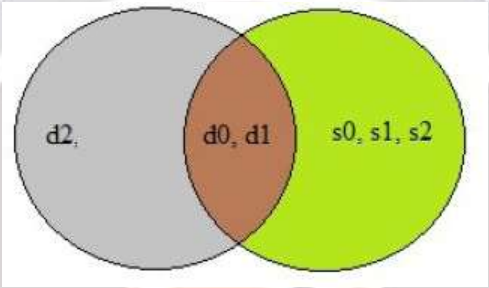

Figure -4: D intersection S

5. RESULT

System results are as follows:

Figure -5: Registration

Figure -6: User Activation

Figure- 7: Token generation

Figure -8: Admin login

Figure -9: User login

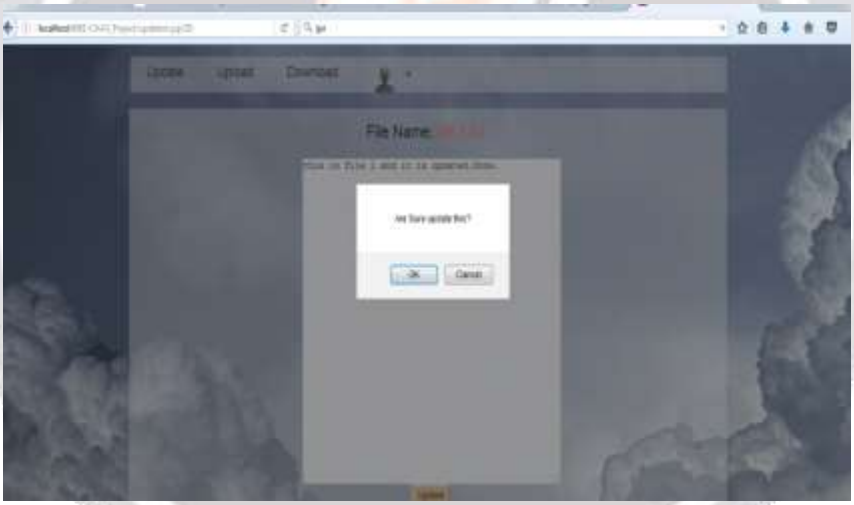

Figure- 10 : File Updating

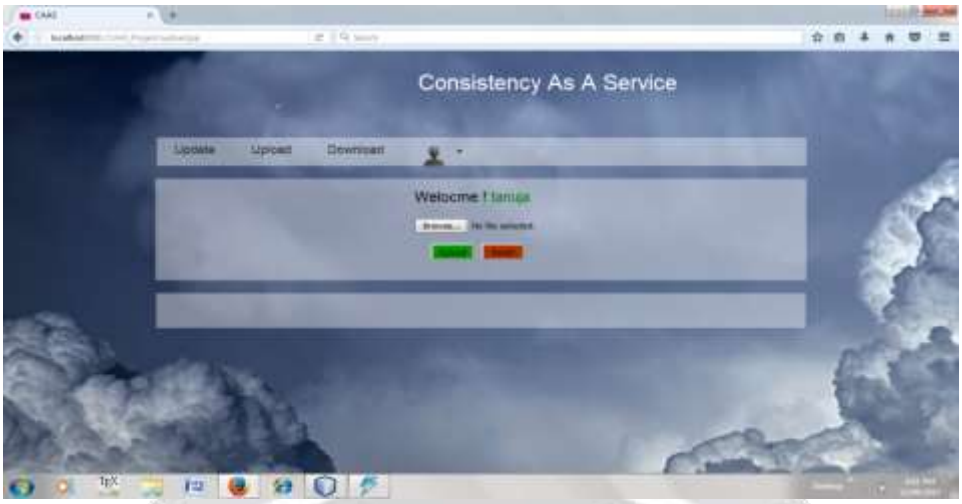

Figure -11 : File Uploading

6. RESULT ANALYSIS

If the user u1 is updating the file and the user u2 is requesting download or update request for same file, the access should be denied as the consistency is not maintained if access is provided.

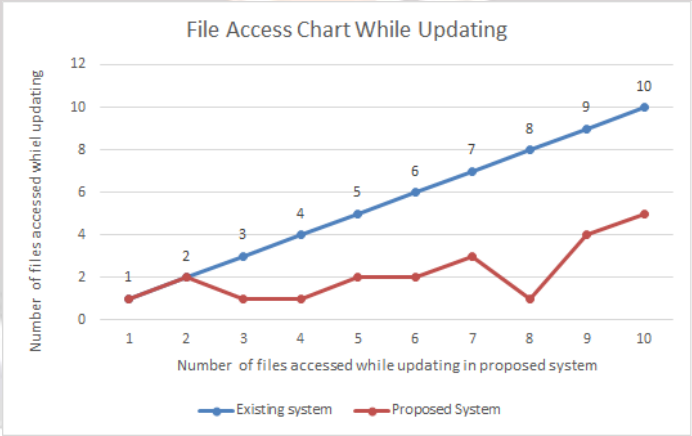

Chart -1: Result analysis

7. CONCLUSION

Consistency as a service (CaaS) model and a global auditing structure to help users validate whether the cloud service provider (CSP) is providing the promised consistency and to quantify the severity of the violations is any . With the CaaS model, the users can assess the quality of cloud services and select a right cloud service provider among various candidates, for example the least expensive one that still provides adequate consistency for the users' application.

8. REFERENCES

[1] W. Lloyd, M. Freedman, M. Kaminsky, and D. Andersen, "Don't settle for eventual: scalable causal consistency for wide-area storage with COPS," in Proc. 2011 ACM SOSP.

[2] J.Misra. Axioms for memory access in asynchronous hardware systems. ACM Transactions on Programming Languages and Systems, 8(1):142–153, January 1986.

[3] E. Anderson, X. Li, M. Shah, J. Tucek, and J. Wylie, “What consistency does your key-value store actually provide,” in Proc. 2010 USENIX HotDep.

[4] W. Golab, X. Li, and M. Shah, “Analyzing consistency properties for fun and profit,” in Proc. 2011 ACM PODC.

[5] A. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms. Prentice Hall PTR, 2002.

[6] W. Vogels, “Data access patterns in the Amazon.com technology platform,” in Proc. 2007 VLDB.

[7] M. Armbrust, A. Fox, R. Gri-th, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. , "A view of cloud computing," Commun. ACM , vol. 53, no. 4, 2010.

[8] C. Fidge, "Timestamps in message-passing systems that preserve the partial ordering," in Pro c. 1988 ACSC .

[9] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, "Consistency rationing in the cloud: pay only when it matters," in Pro c. 2009 VLDB

[10] H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu, "Data consistency properties and the tradeoffs in commercial cloud storages: the consumers' perspective," in Pro c. 2011 CIDR.