Consistency as a Service: Providing Data Cloud Consistency Using Auditing

Priti Garud[1], D. B. Zine[2]

[1]M.E. Student, Computer Engineering Department, VACOE Ahmednagar

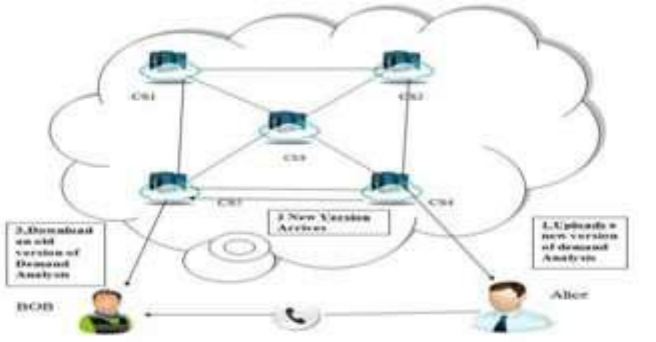
[2]Assistant professor & Head of Computer Engineering Department, VACOE Ahmednagar

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 two level auditing architecture, and this architecture requires a loosely synchronize clock in the audit cloud. Then, design algorithms to quantify the commonality of violations metrics, and the staleness of the value of a read metrics. Finally, we organise a heuristic auditing strategy (HAS) to reveal as many violations as possible. Extensive experiments were performed using a combination of real cloud deployments and simulations to validate heuristic Auditing Strategy.***

Keywords: Cloud Storage, Global Consistency Auditing, Local Consistency Auditing, heuristic auditing strategy (HAS)

1. Introduction

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 3resource provisioning scalability, flexibility and low cost. Amazon DB, Microsoft Azure Storage DB are the eg. 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 eg.


CS1
CS2
CS5
CS3
CS4
2.New Version Arrives
1.Uploads a new version of demand Analysis
3.Download an old version of Demand Analysis
BOB
Alice

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 perform 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

To maintain consistency in cloud computing have a big issue, so here we firstly discuss consistency of model. Mainly cloud consistency can be classified in two types, data centric consistency and cloud centric consistency as shown in fig. 2

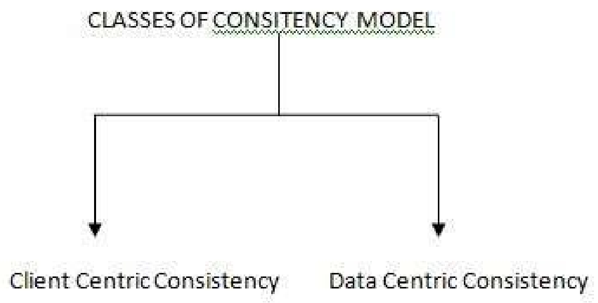


Figure 2: Classes of consistency Model

Client Centric Consistency
The purpose of this consistency to concentrates on specifies customer's requirement, i.e., the way to customers notice data updates. Their work also indicates consistency from strict consistency to weak consistency. Maximum consistency denotes maximum cost and reduced availability.

Actual availability of the data depends on the consistency requirements and the authors provide techniques which make the system dynamically adapt to the consistency level by tracing the state of the data Ref. [1] from the users' point of view we examine the level of consistency provided by cloud service provider existing solution can be derived into 2 types benchmark-based verifications [5]–[8] and traced base verification [2], [4],. Trace-based verifications comprises three consistency semantics, Lamport who propose these 3 semantic regularity, atomicity and safety. If a register is safe if read that is not simultaneously with any write returns the value of the most recent write and a read that is equal to a write can return any value.

If register is regular read that is not simultaneously with any write returns the value of the most recent write, and a read that is simultaneous with a write returns the value of the most recent write, or the value of the concurrent write. If every read returns the value of the most recent write then register is atomic. Checking whether the trace on a read/write register is atomic by Misra to present an algorithm. He Ref [2] proposed offline algorithms for verifying whether a key-value storage system has regular register, atomic register and safe register properties by constructing a directed graph. For online verification algorithm Ref. [4], by using the GK algorithm [7] and various metrics used to quantify the severity of unsuccessful result. The main dis-advantages of the existing trace based verifications is that a global clock is required among all users. Our result refers to trace based verifications. To remove this problem so we used loosely synchronize clock.

Data Centric Consistency
Let us considers the internal state of a storage system. Which checks update flow through the system and what guarantees the system can provide with respect to updates.

Now we describe the consistency service model. Then, we describe the structure of the user operation table (UOT).

A) Consistency Service Model
It contains data cloud and multiple audit cloud in fig 2. In figure 3 the data cloud maintain by Cloud Service Provider (CSP). Unique key is assign to each piece of data, because of data cloud is key value data storage system. Cloud Service Provider maintain data cloud and audit cloud contain a group of users that working on that project And service level Agreement will be form between data cloud and audit cloud which will decide 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 possible to all users due to virtualization technique. It is very difficult for user to check whether each replica in data cloud is newest one or not. We permit the user in audit cloud to check cloud consistency by analyzing the trace interactive operation.

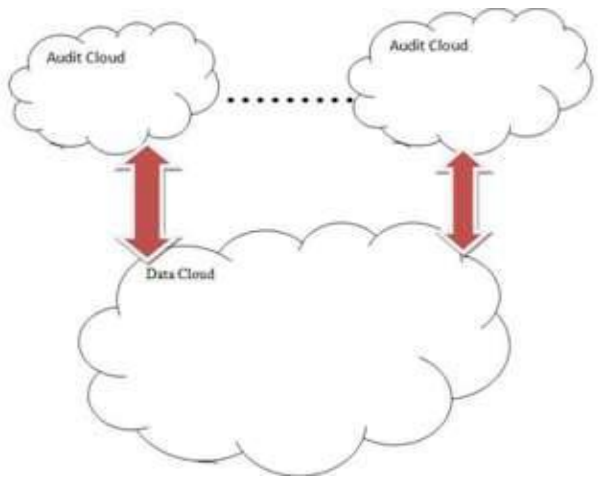


Figure 3: Consistency as a service model

Not need to 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 that, here we develop 2 level of auditing Structure. The two level auditing structure basically contain 2 auditing.

B) User Operation Table (UOT):
A user Operation Table maintains by every users to record logical operation elements logical vector; physical vectors as well as operation are inserted into user operation table. Every operation has write operation or read operation.

Let us consider operation as op, write W (K, a), and read R (K, a).where W (K, a) is nothing but writing the value as to data which is identified by key K. R (K, a) stands reading data which is recognized by key K and whose value is a. Let us consider W (K, a) as R (K, a)'s dictating write, and R (K, a) as W (K, a)'s dictated read. We have the following properties: A read have a unique dictating write is necessary. A write may have either zero or more dictated reads. From the value of a read, we can know the logical and physical vectors of its dictating write. Suppose there are N users in the audit cloud, and A logical per physical vector is a vector of N logical per physical clocks, 1 clock / user, sorted in ascending order of user ID. For a user with I Di., where $1 \leq i \leq N$ and logical vector is < LvC1, LvC2... LvCN >, where LvCi is logical clock , and LvCj is the latest logical clock of user j to his best knowledge, his physical vector is < Pv C1, Pv C2,..., Pv CN > where the term Pv Ci is his physical clock, and Pv Cj is the latest physical clock of user j to the best of his knowledge. Logical vector is modernize by using physical vector and vector clock algorithm also gets modernize in the similar way as logical vector excluding physical clock rises as time passes. Update process is defined below:

Initially all clocks are zero for two vector the users continuously rises his own physical clock in physical vector as sell as rises his one logical clock in logical vector , by one the moment action take place . Two vectors will be sent with message, as soon as user get message he modernize every elements in the vector with maximum value in his own vector along with value in receive vector. Consider, there are three user in audit cloud A, B and C respectively where I DA < I DB < I Dc Each user update vector the details working of vector is shown in the fig 3.

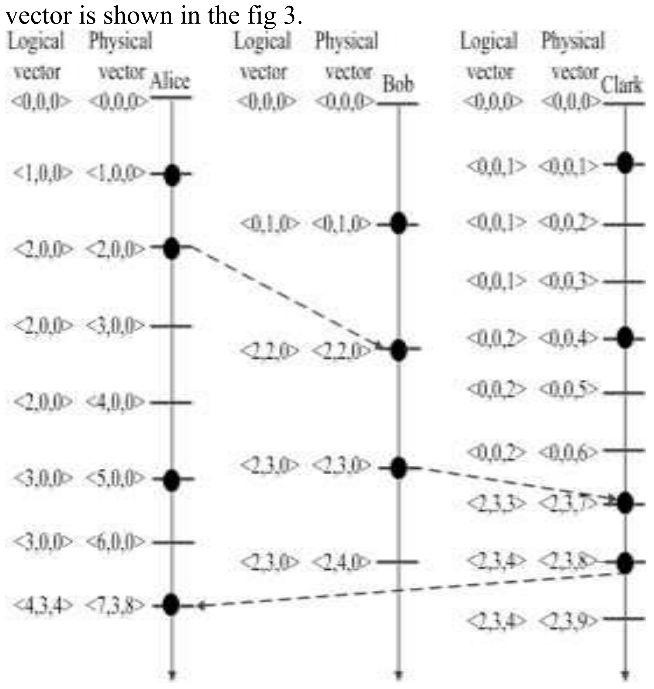


Figure 4: Logical and Physical Vector

As shown in fig. 4. as A w (k,a) is <2,0,0><2,00>hence here logical and physical vector. Following table 1 shows details regarding operations performed on user.

Table 1: User operation table

Alice o**peration log**

Operation	Logical vector	Physical Vector
W(a)	<1,0,0>	<1,0,0>
W(b)	<3,0,0>	<5,0,0>
R(b)	<5,3,5>	<8,3,7>

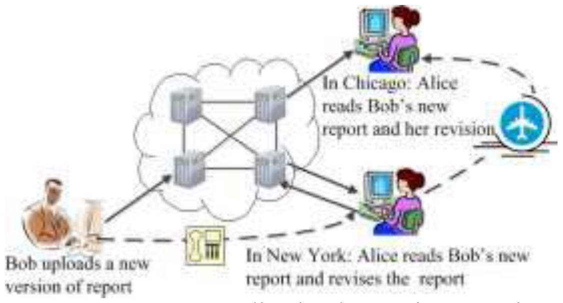
Bobs Operation log

Operation	Logical vector	Physical Vector
W(c)	<0,1,0>	<0,1,0>
W(c)	<2,4,0>	<2,5,0>
R(d)	<2,5,0>	<2,6,0>

Clarks Operation log

Operation	Logical vector	Physical Vector
R(c)	<0,0,1>	<0,0,1>
R(d)	<0,0,2>	<0,0,4>
R(a)	<2,3,5>	<2,3,10>

General review of Two Level Auditing Structure
Local consistency is checked in this section. Every user performs local auditing separately with his own user operation table.


In Chicago: Alice reads Bob's new report and her revision
Bob uploads a new version of report
In New York: Alice reads Bob's new report and revises the report
Figure 5: An application has various consistencies

Here we discuss three consistencies

- Monotonic read consistency
- Read your write consistency
- Causal consistency

Monotonic read consistency
If any process read the value of data X and successive read on data X then same value or more result value is obtain.

Read your consistency
If write of process on data X will be seen by the successive reads on data X by the same process.

Causal Consistency
Write those are causally related then it must be seen to all processes in the same arrangement simultaneous writes may be seen in different arrangement and different machines.

3. Literature Survey

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

Geo-replicated, distributed data stores that supp ort 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. The central approach in COPS is tracking and explicitly checking whether causal dependencies between the keys are satisfied in the local cluster before exposing writes. Further, in COPS- GT, this system introduces get transactions in order to obtain a consistent view of multiple keys without locking or blocking. The evaluation shows that COPS completes operations in less than 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. It also shows that COPS-GT provides similar latency, throughput, and scaling to COPS for common workloads.

***B.* 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. His algorithm 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). Atomicity stipulates that if a value is replaced by another value, then the old value is not allowed to re-appear in the 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.

C. 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 with when they are stored into an uncertain storage pool outside the enterprise. Therefore, it is indispensable for cloud service providers 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. So the proposed system has data fragmentation, data security and storage on multiple cloud services. This system used trusted third party to store the data on multiple cloud and find the data access by un-trusted cloud service providers. In this system the client data divided into 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.

***D.* What consistency does your key-value store actually provide:**

A different approach to measuring consistency of cloud storage platforms is taken by Anderson et al [3], 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.

***E.* 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. To this end, this system formulates 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.

F. 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. The algorithms do not change the communications graph or require a central timestamp issuing authority.

G. 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 includes 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.

4. Conclusion

Here we can reason that our proposed framework information DE duplication of record is done approves way and safely. In this we have additionally proposed new duplication check system which produce the token for the private document. The information client needs to present the benefit alongside the united key as a proof of possession. We have settled more basic piece of the cloud information stockpiling which is just endured by diverse systems. Proposed routines guarantee the information duplication safely.

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. Grith, 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.