# Overview of Auditing Cloud Consistency

Hemant T. Aher[1]
BE Student at BVCOE&RI
University of PUNE
Nasik, Maharashtra, India
*aher.hemant@hotmail.com*

Poonam D. Shirode[2]
BE Student at BVCOE&RI
University of PUNE
Nasik, Maharashtra, India
*pshirode25@gmail.com*

Kundan L. Shinde[3]
BE Student at BVCOE&RI
University of PUNE
Nasik, Maharashtra, India
*kundancool007@gmail.com*

Arti A. Jadhav[4]
BE Student at BVCOE&RI
University of PUNE
Nasik, Maharashtra, India
*jadhav.arti655@gmail.com*

***Abstract* —** Cloud storage services have become very popular due to their infinite advantages. To provide always-on access, a cloud service provider (CSP) maintains multiple copies for each piece of data on geographically distributed servers. A major disadvantage of using this technique in clouds is that it is very expensive to achieve strong consistency on a worldwide scale. In this system, a novel consistency as a service (CaaS) model is presented, which involves a large data cloud and many small audit clouds. In the CaaS model we are presented in our system, a data cloud is maintained by a CSP. A group of users that participate an audit cloud can verify whether the data cloud provides the promised level of consistency or not. The system proposes a two level auditing architecture, which need a loosely synchronize clock in the audit cloud. Then design algorithms to measure the severity of violations with two metrics: the commonality of violations, and the oldness value of read. Finally, heuristic auditing strategy (HAS) is devised to find out as many violations as possible. Many experiments were performed using a combination of simulations and a real cloud deployment to validate HAS.

***Keywords-*** *Cloud Service Provider (CSP), Consistency as a Service (CaaS), Heuristic Auditing strategy, Service Level Agreement, User Operation Table, Directed Acyclic Graph, Network Time Protocol*

*****

## I. INTRODUCTION

CLOUD computing is popular commercially because it provides three main factors needed for computing like scalability, elasticity, and high availability at low cost[1], [2]. According to today's trend of everything-as-a-service (XaaS) model, data storages, virtualized infrastructure & platforms, software & applications are being provided and consumed as service in cloud. Typical cloud storage services are like data storage, which are billed according to the size of storage, example Google drive which provide free storage service up to 15GB per user and charge for extra storage. The main advantage of this kind of services is user can access this service any time as needed and from anywhere. These services are device independent, no need to purchase special hardware to access the service.

To fulfill the commitment of 24/7 access, the cloud service provider (CSP) stores multiple copies of same data on different servers located at different locations geographically. The problem in using this technique is that it required high cost to achieve strong consistency on global scale. According to CAP principle[3],[4] various CSPs only provide weak consistency level like eventual one, to provide high performance and availability, where users view old data for a period of time. One of the most popular applications like the DNS (Domain Name System) uses eventual consistency. In eventual consistency model the updates are not visible immediately all users are ensured to see them in some time. However, eventual type of consistency is not suitable for all kind of applications. Interactive type of applications specially needs stronger consistency.

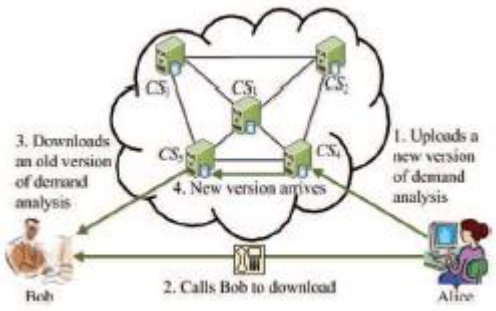


Fig.1 An application that requires causal consistency

Consider the scenario shown in Fig. 1. In this case assume that Alice and Bob are working on a project using cloud storage services, the data related to the project is copied to five cloud storage servers CS1,…..,CS5. Alice upload a new version of the requirement analysis to CS4, then calls Bob to download the updates. The causal relationship [5] must be established between update and read. Therefore, the cloud should provide causal consistency, which guarantees that Alice's update is committed to all copies before Bob's read. If the server provides eventual consistency only, then Bob read an old update which may not satisfy customer requirements.

Generally, the consistency requirement varies according to its applications. For example, social networking services need causal consistency whereas mail servers need monotonic read consistency [6]. The system considering both correctness and cost per transaction. The CaaS (consistency as a service) model presented in this paper. It consists of mainly two types of clouds large data clouds and multiple small audit clouds. The

data cloud is maintained by Cloud Service Provider (CSP) and audit cloud is maintained by a group of users which are cooperating on the same project, job, etc. The Service Level Agreement (SLA) is established between the data and audit cloud, which provide guidelines about the level of consistency should be provided, and how much cost is charged if the SLA is violated.

The implementation detail of the data cloud is hidden to the users because of virtualization technique. So, it is hard for users to find the multiple copies of the data in the cloud is latest or not. The solution provided in [7], users in the audit cloud can verify the cloud consistency by analyzing their interactive operations. The system only need loosely synchronized clock for our solution. Each user has to maintain a logical vector [8] for partial ordering of operations. Here the presented system uses two level auditing scheme, Firstly each user perform local auditing on its own with local trace of operations, Secondly global auditing is perform by elected auditor from the users with an global trace of operations. The two main focus area for auditing, on local level it on monotonic read and read your write consistencies can be perform by a light weight online algorithm, on global level causal consistency auditing perform by constructing directed graph, if the graph is directed acyclic graph (DAG) then conclusion is that causal consistency is preserved. The violations are measure on two factors: commonality and staleness of the value of read.[9] Finally HAS (Heuristic Auditing Strategy) use to add appropriate reads to find out as many as violations possible.

Key Factors are as follows:
1) System Present CaaS model, which consist of data cloud and audit cloud
2) System suggests a two level auditing structure.
3) System design algorithms to measure the occurrences of violations with different metrics.
4) In this system devise HAS to find out as many as violations possible.

## II. Related Work

A cloud is basically a major distributed system where each portion of data is copied on multiple globally distributed servers to attain high accessibility and high performance. Thus, we first check the consistency models in distributed systems. Ref. [10], as anticipated two consistency models: data-centric consistency and client-centric consistency. Data-centric consistency model consider the inner state of a storage system, that how updates stream through the system and what guarantees the system can supply with respect to updates. On the other hand, to a customer, it actually does not matter whether or not a storage system inside contains any old copies. As long as no old data is observed from the client's side, the customer is satisfied. Therefore, client-centric consistency model focuses on what exact customers want, with the aim of is how the customers view data updates. Their work also describes multiple levels of consistency in distributed systems, as of strict consistency to weak consistency. High consistency results in high cost and reduced availability. Firm consistency is never necessary in practice [11], and is even considered detrimental. In reality, by the CAP protocol [3], [4], many distributed systems forgo strict consistency for availability.

Then, the system analyzes the work on attaining different levels of consistency in a cloud. Investigated the consistency properties provided by commercial clouds and made several useful opinions [12]. Existing commercial clouds generally limit strong consistency promises to small datasets (Google's Megastore and Microsoft's SQL Data Services), or provide only eventual consistency (Amazon's simpleDB and Google's BigTable) [13]. The consistency requirements differ over time depending on tangible accessibility of the data, and the authors deliver techniques that make the system dynamically adjust to the consistency level by monitoring the state of the data. The proposed novel consistency model that allows it to automatically modify the consistency levels for altered semantic data [14].

Finally, we analyze the work on authenticating the levels of consistency provided by the CSPs from the user's point of view. Existing solutions can be categorized into trace-based verifications [7], [9] and benchmark-based verifications [15]-[18]. Trace-based verifications focus on three consistency semantics: safety, regularity. A register is safe if a read that is not coexisting with any write returns the value of the most recent write, and a read that is coexisting with a write can return any value.

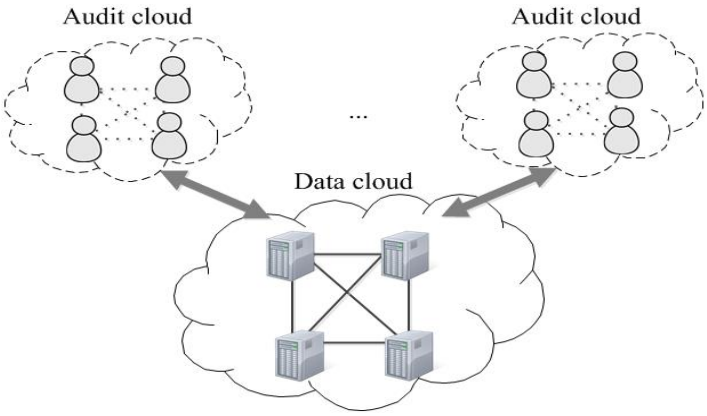


Fig.2 Consistency as a service model.

A register is regular if a read that is not coexisting with any write returns the value of the most contemporary write, and a read that is coexisting with a write returns either the value of the most contemporary write, or the value of the coexisting write. A register is atomic if every read returns the value of the most contemporary write. Misra [19] is the first to present an algorithm for confirming whether the suggestion on a read/write catalog is atomic. Following his work, Ref. [7] proposed offline algorithms for validating whether a key-value storage system has protection, reliability, and atomicity properties by assembling a directed graph. Ref. [9] offered an online verification algorithm by using the GK algorithm [20], and Used diverse metrics to enumerate the brutality of violations.

The main weakness of the existing trace-based authentications is that a global clock is required among all users. Our solution belongs to trace-based authentications. However system emphasis on different consistency semantics in commercial cloud systems, where a loosely synchronized clock is proper for our explanation Benchmark-based authentications emphasis on benchmarking in a storage

system. The results of validate our two-level auditing structure. Refer client centric benchmarking approach for understanding ultimate consistency in scattered key value storage systems. Amazon, Google, and Microsoft's contributions showed that, in Amazon S3, consistency was surrendered and only a weak consistency level known as, eventual consistency was attained.

## III. PRELIMINARIES

In preliminaries section, first system illustrates the consistency as a service (CaaS) model. Then, it illustrates the structure of *the user operation table* (UOT), with which each user records his operations. Lastly, system makes available an overview of the two-level auditing structure and associated definitions.

### A. *Consistency as a Service (CaaS) Model*

The CaaS model consists of a *data cloud* and multiple *audit clouds*. Data cloud is maintained by the cloud service provider (CSP), is a key-value data storage system where each part or piece of data is recognized by a unique key. The CSP replicates all of the data on multiple geographically distributed cloud servers to afford always-on services.

An audit cloud consists of a group of users that assist on a job. Now assume that each user in the audit cloud is identified by a unique ID. The audit cloud and the data cloud will engage in a service level agreement (SLA), before outsourcing the job to the data cloud, Which specifies the promised level of consistency. The audit cloud verify whether the data cloud violates the SLA or not, and to enumerate the severity of violations.

In this system, a two-level auditing model is implemented: each user records his operations in a user operation table (UOT), which is referred to as a local trace of operations. *Local auditing* can be carry out freely by each user with his own UOT; periodically, an auditor is designated from the audit cloud. In this, all other users will send their UOTs to the auditor, which will present *global auditing* with a global trace of operations. The system simply let each user turn into an auditor. The dotted line in the audit cloud shows that users are loosely connected. It implies that users will communicate to exchange messages after executing a set of reads or writes, rather than communicating instantly after executing each operation. Once two users finish communication, a causal relationship on their operations is established.

### B. *User Operation Table (UOT)*

Each record in the UOT has three elements: *operation*, *logical vector*, and *physical vector*. User will record operation, his current logical vector and physical vector, while issuing an operation in his UOT.

### C. *Overview of Two-Level Auditing Structure*

System examined several consistency models provided by profitable cloud systems. Following their work, we provide a two-level auditing structure for the CaaS model. At the first each user independently performs local auditing at his own with UOT. The following consistencies should be verified at this level

**Monotonic-read consistency.** *If a process reads the value of data, any successive reads on data by that process will always return that same value or a more recent value*

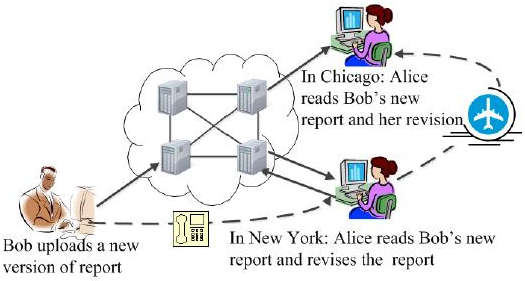


Fig.4. shows an application that has different consistency requirements.

In Fig. 4, after uploading a latest version of the report to the data cloud, Bob ask over Alice to download it. After the call, Bob's update and Alice's read are causally associated. Therefore, causal consistency needs that Alice must read Bob's new report.

**Read-your-write consistency.** *Effect of a write by a process on data K will always be seen by a successive read on data K by the same process.*

**Causal consistency.** *Writes that are causally related must be seen by all processes in the similar order. Simultaneous writes may be seen in a different order on different machines*

## IV. VERIFICATION OF CONSISTENCY PROPERTIES

In this, systems first afford the algorithms for the two level auditing structure for the CaaS model, and then analyze their success. Finally, system demonstrates how to perform a trash collection on UOTs to save space. As the accesses of data with different keys are independent of each other, a user can group operations by key and then verify whether each group satisfies the promised level of consistency. After that, system reduces read operations with $R(a)$ and writes operations with $W(a)$.

### A. *Local Consistency Auditing*

**Algorithm 1** Local consistency auditing

Initial UOT with $\emptyset$
**While** issue an operation *op* *does*
  **If** $op = W(a)$ **then**
    Record $W(a)$ in UOT
  **If** $op = r(a)$ **then**
    $W(b) \in$ UOT is the last write
    **If** $W(a) \rightarrow W(b)$ **then**
      Read-your-write consistency is violated
    $R(c) \in$ UOT is the last read
    **If** $W(a) \rightarrow W(c)$ **then**
      Monotonic-read consistency is violated
    Record $r(a)$ in UOT

Where,
UOT – User Operation Table

R (a) – Users Current Read
W (a) – Current reads dictating write
R (c) – Last Read in UOT
W (c) – Last Read in UOT's dictating write
W (b) – Last write in UOT

This is an online algorithm. In this each user will record all of his operations in his UOT. User will perform local consistency auditing independently.

### B. Global Consistency Auditing

**Algorithm 2** Global consistency auditing
Every operation in the global trace is represented by a vertex
**Let** any two operations *op*1 and *op*2 **do**
**If** *op*1 → *op*2
**Then**
A time edge is added from *op*1 to *op*2

**If** *op*1 = *W* (*a*), *op*2 = *R* (*a*), and two operations come from different users
**Then**
A data edge is constructed from *op*1 to *op*

**If** *op*1 = *W* (*a*), *op*2 = *W* (*b*), two operations come from different users, and *W* (*a*) is on the route from *W* (*b*) to *R* (*b*)
**Then**
A causal edge is added from *op*1 to *op*2 Check whether the graph is a DAG by topological sorting.

This is an offline algorithm (Alg. 2). An auditor will be chosen periodically from the audit cloud to perform global consistency auditing. All other users will submit their UOTs to the auditor for getting a global trace of operations. After performing global auditing, the auditor will send audit results as well as its vectors to all other.

### C. Effectiveness

Effectiveness of the local consistency auditing algorithm is easy to demonstrate. For monotonic-read consistency, a user is needed to read either the same value or a newer value. Hence, if the dictating write of a new read happens before the dictating write of the last read, then system say that monotonic read consistency is violated. In case of read-your-write consistency, the user is needed to read his latest write. Hence, if the dictating write of a new read happens before his last write, system can say that read-your-write consistency is violated.

For causal consistency, system should prove that:
(1) There is an violation if the constructed graph is not a DAG.
(2) There is no violation if the graph is DAG.

### C. Garbage Collection

Each user should keep all operations in his UOT in the process of auditing, exclusive of intercession; the size of the UOT would grow without bound. Also, the communication cost for transferring the UOT to the auditor will be too much. So, system provides a garbage collection system which can delete unnecessary records, which will preserve the efficiency of auditing.

In local consistency auditing, suppose dictating write of a new read does not exist in the UOT and the dictating write is issued by the user, the user can say that he has failed to read his last updates, and asserts that read-your-write consistency is violated.

Suppose the dictating write of this read happens before the dictating write of his last read recorded in the UOT, the user can say that he has read an old value, and asserts that monotonic-read consistency is violated. Let the dictating write of a new read does not present in the user's UOT and the dictating write comes from other users, then a violation will be exposed by the auditor.

In global consistency auditing, if a read that does not have a dictating write, then the auditor say that the value of this read is too stale, and state that causal consistency is violated.

**Summary.** HAS can detect nearly all of the violations when the inception value and interval length are chosen accurately; Random can perceive only about 60% of destructions. Although HAS involves the auditing cloud to dispute more auditing reads, the grossed profit is still higher than Random. Specifically, as the parameters inception value and interval length reduce, HAS works better

## V. QUANTIFYING SEVERITY OF VIOLATIONS

System provides two ways to quantify the severity of violations for the Camas model: commonality and staleness. Commonality quantifies how the violations happen whereas the Staleness quantifies how much older the value of a read is compared to that of the latest write. Staleness can be classified into time-based staleness and operation based staleness. Commonality can be easily quantified by leasing each user set a local counter and increasing the counter by one when a local consistency violation is exposed. Commonality can be quantified by counting the number of cycles in the erected graph for global consistency. This can be transformed into removing the less number of edges to make the graph acyclic.

## VI. HEURISTIC AUDITING STRATEGY

From the auditing procedure in the Camas model, we see that only reads can expose violations by their values. Hence, the basic indication of our heuristic auditing strategy (HAS) is to add appropriate reads for illuminating as many violations as possible. We call these supplementary reads as *auditing reads*. As shown in Fig. 6, Physical time is divided into *L* time slices according to HAS, where *l* time slices established an intermission.

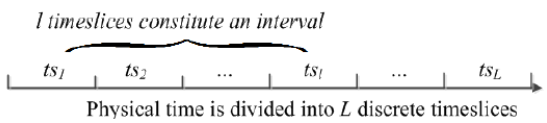

Fig.6 Physical time is divided into time slices.

Each time slice is associated with a state, which can be either *normal* or *abnormal*. A normal state denotes that there is no consistency violation, and an abnormal state denotes that there is one destruction in this time slice. Under the CaaS

model, consistency suits a part of the SLA, the users can acquire proportional benefit from the CSP, by illuminating consistency violations and enumerating the severity of the violations. We believe that the CaaS model will help both the CSP and the users implement consistency as an important fact of cloud services offerings.

## VII. EVALUATION

In this unit, system unite HAS with a random strategy, denoted as Random. To confirm the efficiency of HAS, system conduct tests on synthetic as well as real violation traces.

### *A. Synthetic Violation Traces*

System review the parameters used in the artificial violation races in Table II. In the random strategy, system erratically choose [1, *l*] auditing reads in each recess, where *l* is the length of an recess. To obtain the synthetic violation traces, physical time is divided into 2,000 time slices. We accept that once a data cloud activates to violate the assured consistency, this violation will continue for several time slices, rather than ending instantaneously. In the recreation, the period of each violation *d* is set to 3-10 time slices.

Consider that the audit cloud can earn $5 from the data cloud once a consistency violation is detected; the audit cloud will be charged $0.1 for an auditing read task. Fig. 8 shows the contrast results of the earned profit *P*. From Fig. 8, we know that HAS typically earns a higher profit than Random. Finally, HAS will produce higher earned profit as the parameters $\alpha$ and *l* decrease.

### *B. Real Violation Traces*

To check the productivity of HAS, system collect traces from two real clouds. We use network time protocol (NTP) to coordinate time amongst all cases. We know that the proportion of exposed destructions reduces as *l* rises, in terms of both HAS as well as Random. However, the change of *l*'s value has less impression on HAS than Random. We know that the proportion of exposed destructions drops as $\alpha$ rises or k falls. However, these factors have slight effects on the proportion of exposed destructions. We know the percentage of revealed violations decreases as α *increases*.

## VIII. CONCLUSION

In this paper, The presented system is a consistency as a service (CaaS) model and a two-level auditing scheme to help users validate whether the cloud service provider (CSP) is providing the promised consistency, and to enumerate the occurrences of the violations. The CaaS model used in the system helps the users can assess the superiority of cloud services and decide a right CSP among various services. For example the less costly one that still provides satisfactory consistency for the users' applications.

## REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, 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.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing (draft)," NIST Special Publication 800-145 (Draft), 2011.

[3] E. Brewer, "Towards robust distributed systems," in *Proc. 2000 ACM PODC*.

[4] "Pushing the CAP: strategies for consistency and availability," *Computer*, vol. 45, no. 2, 2012.

[5] M. Ahamad, G. Neiger, J. Burns, P. Kohli, and P. Hutto, "Causal memory: definitions, implementation, and programming," *Distributed Computing*, vol. 9, no. 1, 1995.

[6] 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*.

[7] 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*.

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

[9] W. Golab, X. Li, and M. Shah, "Analyzing consistency properties for fun and profit," in *Proc. 2011 ACM PODC*.

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

[11] W. Vogels, "Data access patterns in the Amazon.com technology platform," in *Proc. 2007 VLDB*.

[12] "Eventually consistent," *Commun. ACM*, vol. 52, no. 1, 2009.

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

[14] 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.

[15] H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu, "Data consistency properties and the trade-offs in commercial cloud storages: the consumers' perspective," in *Proc. 2011 CIDR*.

[16] D. Bermbach and S. Tai, "Eventual consistency: how soon is eventual?" in *Proc. 2011 MW4SOC*.

[17] M. Rahman, W. Golab, A. AuYoung, K. Keeton, and J. Wylie, "Toward a principled framework for benchmarking consistency," in *Proc. 2012Workshop on HotDep*.

[18] D. Kossmann, T. Kraska, and S. Loesing, "An evaluation of alternative architectures for transaction processing in the cloud," in *Proc. 2010 ACM SIGMOD*.

[19] J. Misra, "Axioms for memory access in asynchronous hardware systems," *ACM Trans. Programming Languages and Systems*, vol. 8, no. 1, 1986.

[20] P. Gibbons and E. Korach, "Testing shared memories," *SIAM J. Computing*, vol. 26, no. 4, 1997.

**Hemant T Aher** he is engineering student of Information Technology at NGSPM Brahma Valley College, Nasik under University of Pune. His areas of interest include Cyber Security, Networking.

 ISSN: 2321-8169
46 - 51

**Poonam D Shirode** she is engineering student of Information Technology at NGSPM Brahma Valley College, Nasik under University of Pune. Her areas of interest include Web Designing, Cloud Computing.

**Kundan L Shinde** he is engineering student of Information Technology at NGSPM Brahma Valley College, Nasik under University of Pune. His areas of interest include Programming, Networking.

**Arti A Jadhav** she is engineering student of Information Technology at NGSPM Brahma Valley College, Nasik under University of Pune. Her areas of interest include Web Designing, Computer Network Security.