International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST229396

Competitive Analysis for the Auditing Cloud Consistency

Rashmi Ashtagi[1], Jyoti R Maranur[2], Afsha Akkalkot[3], Shweta M Madiwal[4], Sridevi Hosmani[5], Arunadevi Khaple[6]
[1,3,6]Zeal College of Engineering and Research, SPPU, Pune, India,
[2,4,5]Faculty of Engineering and Technology (Exclusively for Women), Sharanbasava University, Kalburgi, India

Article Info
Volume 9, Issue 3
Page Number : 408-414

Publication Issue
May-June-2022

Article History
Accepted : 01 June 2022
Published : 06 June 2022

ABSTRACT

Cloud storage is one of the service of cloud comput- ing. Cloud storage services are commercially popular because to their advantages. It allows data owners to move data from their local computing systems to the cloud. It offers high quality and on-demand data storage services to users. A cloud is essentially a large-scale distributed system; each piece of data is replicated on multiple geographically distributed servers to achieve high availability and high performance. A cloud service provider (CSP) keeps multiple replicas for user's data on geographically distributed servers. A main problem of using the replication technique in clouds is that it is very expensive to achieve strong consistency on a worldwide scale. In this paper, we are reviewing consistency as a service (CaaS) model, a two-level auditing architecture and a heuristic auditing strategy (HAS) that reveals as many violations as possible.

Keywords : Cloud storage services, cloud service provider, con- sistency, consistency as a service, heuristic auditing strategy.

I. INTRODUCTION

An increasing number of organizations prefer to outsource the data to cloud servers. This relieves the burden of both data management and maintenance. A cloud is a large-scale distributed system where each piece of data is replicated on multiple geographically distributed servers to achieve high availability and high performance. A wide variety of services are categorized as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [3].

A. Infrastructure as a Service (IaaS)

IaaS is the hardware and also software that powers servers, storage, networks, operating systems. Infrastructure as a Service is a model in which an organization outsources the resources that are used to maintain operations, including hardware, storage, servers and various networking components. So storage is an Infrastructure as a Service. The service provider keeps all these equipment and is responsible for running and maintaining it. Clients pay the cost as they use the resources of cloud.

Characteristics of IaaS are as follows:
• Utility computing service
• Administrative tasks automation
• Dynamic scaling
• Desktop virtualization

Policy-based services Amazon EC2, Windows Azure, Rackspace, Google Compute Engine
• Internet connectivity
IaaS Amazon EC2, Rackspace,Windows Azure, Google Com- pute Engine are exa.

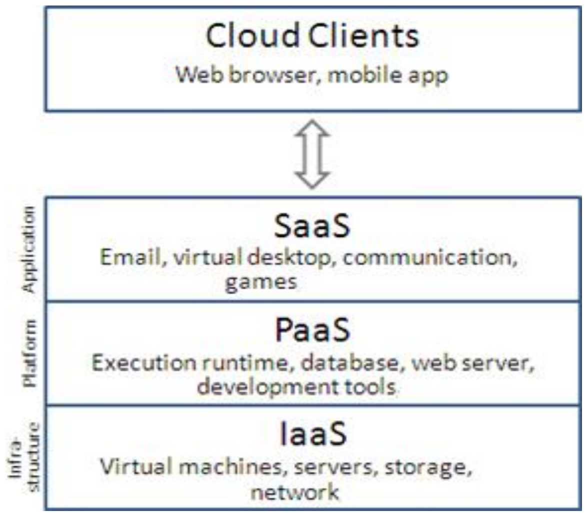
B. Platform as a Service (PaaS)
PaaS is the set of tools and services designed to make coding and deploying those applications quick and efficient. In the PaaS models, cloud providers deliver a computing platform, such as operating system, programming language execution environment, database, and web server. Application developers develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers.

Characteristics of PaaS are as follows:

Services to develop, deploy, test, host and also main- tain applications in the same integrated development environment. All these different services need to fulfill the application development process.
Web based user interface creation tools can create, deploy, modify and test different UI scenarios.
Multiple users can utilize the same development ap- plication concurrently.
Common standards are maintained for integration with web services and databases.
Support for development team collaboration some PaaS solutions includes project planning and commu- nication tools.

PaaS Example: AWS Elastic Beanstalk, Windows Azure, Force.com, Google App Engine.

C. Software as a Service (SaaS)
SaaS is also referred to as ”on-demand software” means SaaS applications can be accessed by users whenever they need. SaaS applications are mainly designed for end-users and they are delivered over the web.
Characteristics of SaaS are as follows:
• All software are managed from a central location.
Delivery of software is based on a “one to many” model.
• Commercial software access through Web.
• Users are not required to handle software upgrades.
APIs allow for integration between different software pieces.
Example of SaaS: Salesforce (CRM SaaS application), Mi- crosoft Office 365, Google Apps. Cloud allows data owners to


Cloud Clients
Web browser, mobile app
Application
SaaS
Email, virtual desktop, communication, games
Platform
PaaS
Execution runtime, database, web server, development tools
Infra- structure
IaaS
Virtual machines, servers, storage, network
Fig.1: Service Models

move data from their local computing systems to the cloud. It offers high quality and on-demand data storage services to users. This frees them from the burden of maintenance [5]. Cloud storage services promise high scalability and availability at low cost. Cloud storage services involve the delivery of data storage as a service, both database-like services and network attached storage. Cloud storage services are billed on a utility computing basis [1], [6]. A cloud storage service allows the customers to access data stored in a cloud anytime and anywhere using internet.

II. CONSISTENCY

A Consistency Model is a contract between the software and the memory if it states that the memory will work correctly but only if the software obeys certain rules.

A. Types of Consistency Models [7]

1) Strict Consistency: Strict consistency is the strictest model in which a read returns the most recently written value. The changes are updated instantaneous.

2) Causal Consistency: Writes which are causally re- lated must be seen by all processors in the same order.

Causally related writes means the write comes after a read that returned the value of the other write.

3) Sequential Consistency: The results of Sequential Consistency are the same as if operations from dif- ferent processors are interleaved, but operations of a single processor appear in the order specified by the program.

4) Release Consistency: Release consistency is same as weak consistency. In this there are two operations “lock” and “unlock” for synchronization.

a) “lock” means writes on other processors to protected variables will be known.

b) “unlock” means that writes to protected vari- ables are exported.

5) Weak Consistency: Weak consistency means that the programmer should manage synchronization. Weak consistency uses synchronization variables that prop- agate writes to a machine and from a machine at appropriate points:

a) accesses to synchronization variables are se- quentially consistent.

b) no access to a synchronization variable is allowed until all previous writes have com- pleted in all processors.

c) no data access is allowed until all previous accesses to synchronization variables (by the same processor) have been performed.

6) Monotonic-read consistency: If a process reads the value of data X, any successive reads on data X by that process will always return that same value or a more recent value [9].

7) Read-your-write consistency: The effect of a write by a process on data X will always be seen by a successive read on data X by the same process [9].

TABLE I. SUMMARY OF CONSISTENCY MODELS.

Consistency	Description
Strict	Absolute time ordering of all shared accesses matters.
Sequential	All processes see all shared accesses in the same order. Accesses are not ordered in time.
Causal	All processes see causally-related shared accesses in the same order.
Weak	Shared data can be counted on to be consistent only after synchronization is done.
Release	Shared data are made consistent when a critical region is exited.
Monotonic-read	If a process has seen a particular value for the object, any subsequent accesses will never return any previous values
Read-your-write	Data item always accesses the updated value and never will see an older value

Consistency Description

Strict Absolute time ordering of all shared accesses matters. Sequential All processes see all shared accesses in the same order. Accesses are not ordered in time. Causal All processes see causally-related shared accesses in the same order. Weak Shared data can be counted on to be consistent only after synchronization is done. Release Shared data are made consistent when a critical region is exited. Monotonic- read If a process has seen a particular value for the object, any subsequent accesses will never return any previous values Read-your- write Data item always accesses the updated value and never will see an older value The cloud service provider (CSP) stores data replicas on multiple geographically distributed servers so that the client can access data ubiquitously 24/7. A main problem of using the replication technique in clouds is that it is very expensive to achieve strong consistency on a worldwide scale, where a user is ensured to see the latest updates [1].

III. RELATED WORK

A cloud is a large-scale distributed system where each piece of data is replicated on multiple geographically distributed servers to achieve high availability and high performance. There are different levels of consistency in distributed systems, from strict consistency to weak consistency. High consistency requires high cost and reduced availability. Strict consistency is never required in practice as it causes reduced availability [12]. The CAP theorem [13] (consistency, availability and Partition tolerance) states that though its desirable to have Consistency, High-Availability and Partition-tolerance in every system, unfortunately no system can achieve all three at the same time. Hence, many distributed systems sacrifice strict consistency for high availability. Many observations are made based the consistency properties provided by commercial clouds in [9]. Most of existing commercial clouds restrict strong consistency guarantees to small datasets, or provide only eventual consistency. In [14], it demonstrates the opportunities and limitations of using S3 as a storage system for general-purpose database applications which involve small objects and frequent updates. Read, write, and commit protocols are described in this. This paper reviewed the cost, performance, and consistency properties of such a storage system. In [2], a transaction paradigm is presented, that allows designers to define the consistency guarantees on the data instead at the transaction level, and also allows to automatically switch consistency guarantees at runtime. It presents a number of techniques that let the system dynamically adapt the consistency level by monitoring the data. The consistency requirements differ depending on availability of the data and also it differ over time. A consistency model and framework is proposed which capable of enforcing different degrees of consistency, accordingly to the data semantics, for data geo-replication in cloud tabular data stores [15].

The solutions for verifying the levels of consistency pro- vided by the CSPs are classified into:

1) Trace-based verifications [10]: An algorithm is pre- sented that help users check whether a key-value store provides safe, regular, or atomic consistency seman- tics. The disadvantage of this trace-based verification is that a global clock is required among all users.

2) Benchmark-based verifications[16], [17]: A novel ap- proach is presented to benchmark staleness in dis- tributed datastores and use the approach to evaluate Amazon's Simple Storage Service (S3) [16]. Paper [17] assessed Amazon, Google, and Microsofts of- ferings, and showed that, in Amazon S3, consistency was sacrificed and only a weak consistency level known as, eventual consistency, was achieved.

It is possible to use cloud-based data clients to migrate data from their systems to cloud-based servers as discussed in [19,20]. As a result, the customer is relieved of the cost of maintenance while still receiving highquality data storage facilities. Cloud storage raises many security concerns. Cloud service providers and data servers are not without flaws. The consumer is worried with whether or not the information stored on the cloud is in order. Furthermore, for data dynamics, this facilitates dynamic operations such as insert, update, remove, and alter at the block stage. The defragmentation technique can be used to determine whether or not the file that the user wishes to store in cloud storage already exists on the cloud server. This system is powerful and safe against malicious server-launched replace attacks.

IV. SYSTEM OVERVIEW

A. Consistency as a Service (CaaS) Model

The CaaS model consists of a large data cloud and many small audit clouds. The data cloud is maintained by a Cloud Service Provider, and an audit cloud consists of a group of users. Data cloud and the audit cloud will be having service level agreement (SLA).This SLA will specify level of consistency the data cloud should provide, and what will be charged if the data cloud fails to comply with the SLA. The CaaS model is shown in Figure. 2, which consists of a data cloud and many audit clouds. The cloud service provider (CSP)

maintains the data cloud and a key-value data storage system

[8] are used. In this, each piece of data is identified by a unique key. In an audit cloud each user in the audit cloud is identified by a unique ID.

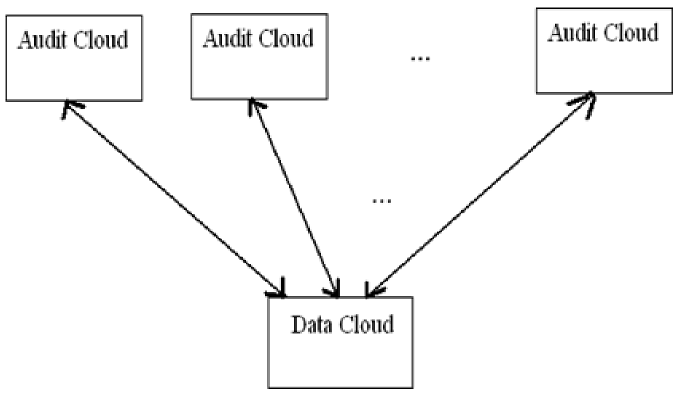


Fig.2: Consistency as a service model.

B. Two-Level Auditing Structure

In a two-level auditing model each user records his opera- tions in a user operation table (UOT) [1]. In this model each user records his operations in a user operation table (UOT). It is referred also as a local trace of operations. A two-level auditing structure for the CaaS model is provided.

In first level, each user independently performs local auditing with his UOT. Two consistencies should be verified at first level, they are:

1) Monotonic-read consistency.
2) Read-your-write consistency.

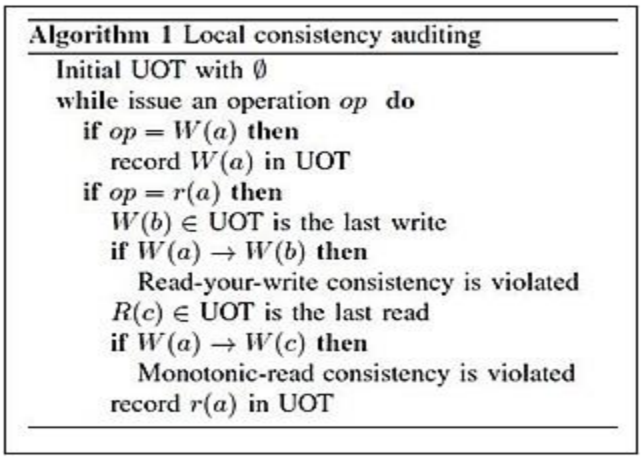
In monotonic-read consistency, a user must read either a newer value or the same value. In read-your-write consistency, a user always reads his latest updates.

At the second level, an auditor gets global trace of all users' operations. Then he performs auditing. Casual consistency should be verified at this level.

Local auditing can be performed independently by each user by using his UOT. An auditor is elected from the audit cloud. All other users will send their UOTs to the auditor, and the auditor will perform global auditing with a global trace of operations. Each user becomes an auditor periodically. Users communicate to exchange messages after executing a set of reads or writes, rather than communicating immediately after executing every operation. When two users finish off their communicating, a causal relationship on their operations is established.A global auditing is performed by constructing a directed graph. If the constructed graph is a directed acyclic graph (DAG), then causal consistency is preserved.

C. Verification Of Consistency Properties

1) Local Consistency Auditing: Local consistency auditing is also known as an online algorithm (Algorithm 1 shown below) [1]. Each user will record all his operations in his User Operation Table. During issue of a read operation, the user will perform local consistency auditing independently.



Algorithm 1 Local consistency auditing

```
Initial UOT with $\emptyset$
while issue an operation $op$ do
    if $op = W(a)$ then
        record $W(a)$ in UOT
    if $op = r(a)$ then
        $W(b) \in$ UOT is the last write
        if $W(a) \to W(b)$ then
            Read-your-write consistency is violated
        $R(c) \in$ UOT is the last read
        if $W(a) \to W(c)$ then
            Monotonic-read consistency is violated
        record $r(a)$ in UOT
```

2) Global Consistency Auditing: Global consistency auditing is also known an offline algorithm (Algorithm 2 shown below) [1]. An auditor is elected from audit cloud. All other users will send their UOTs to the auditor, and the auditor will perform global auditing with a global trace of operations. Once the auditor executes global auditing, the auditor will send auditing results to all other users.

Consistency is verified by constructing a directed graph based on the global trace [10]. Causal consistency is preserved only if the constructed graph is a directed acyclic graph (DAG).

Effectiveness of the local consistency auditing algorithm: In monotonic-read consistency, a user reads 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 monotonic-read consistency is violated. In read-your-write consistency, a user is required to read his latest write. Hence, if the dictating write of a new read happens before his last write, then read-your-write consistency is violated.

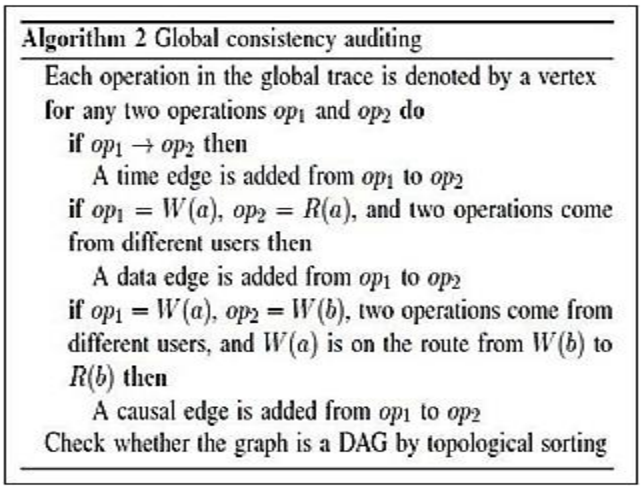
D. Quantifying Severity of Violations

There are two metrics to quantify the severity of violations for the CaaS model [11]. They are:

Commonality: Commonality computes how often the violations happen.

Staleness: Staleness computes how older the value of a read is compared to that of the latest write. Staleness is classified into:

1) Time-based staleness: counts the passage of time between the reads dictating write and the latest write.



Algorithm 2 Global consistency auditing

```
Each operation in the global trace is denoted by a vertex
for any two operations $op_1$ and $op_2$ do
    if $op_1 \to op_2$ 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 added from $op_1$ to $op_2$
    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
```

2) Operation- based staleness: counts the number of intervening operations between the reads dictating write and the latest write.

V. CONCLUSION

This paper reviews on importance of consistency in cloud storage and different types of consistency models. This paper tries to verify whether the cloud service provider (CSP) is providing the promised consistency or not, which helps the users to choose a right cloud service provider among many cloud service providers. A consistency as a service (CaaS) model and a two-level auditing structure verify whether the CSP provides the promised consistency, and also it quantifies the severity of the violations. By this the users can assess the quality of cloud services.

VI. REFERENCES

[1]. Qin Liu, Guojun Wang, Member, IEEE, and Jie Wu, "Consistency as a Service: Auditing Cloud Consistency," in IEEE transactions on network and service management, vol. 11, no. 1, march 2014

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

[3]. H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu, “Data consistency properties and the trade-offs in commercial cloud storages: the con-sumers’perspective,” in Proc. 2011 CIDR.

[4]. P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” as a technical report, Nat’l Inst. of Standards and Technology, 2009.

[5]. Yang, K., Jia, X. (2013). An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Transactions on Parallel and Distributed Systems, 24(9), 17171726.

[6]. M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski,

[7]. G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., “A view of cloud computing,” Commun. ACM, vol. 53, no. 4, 2010.

[8]. Andrew S. Tanenbaum , Maarten Van Steen, Distributed Systems: Principles and paradigms.

[9]. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazons highly available key-value store,” in Proc. 2007 ACM SOSP.

[10]. Werner vogels, “Eventually consistent,” Commun. ACM, vol. 52, no. 1, 2009.

[11]. 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.

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

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

[14]. E. Brewer, “Towards robust distributed systems,” in Proc. 2000 ACM PODC.

[15]. M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska, “Building a database on S3,” in Proc. 2008 ACM SIGMOD.

[16]. 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.

[17]. D. Bermbach and S. Tai, “Eventual consistency: how soon is eventual?” in Proc. 2011 MW4SOC.

[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]. D. B. Terry et al. Session guarantees for weakly consistent replicated data. In Proc of International Conference on Parallel and Distributed Information Systems (PDIS’94)., pages 140149. IEEE Computer Society, 1994.

[20]. Rashmi, R. Patil, Yatin Gandhi, Vinaya Sarmalkar, Prajakta Pund, and Vinit Khetani. "RDPC: Secure Cloud Storage with Deduplication Technique." In 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 1280-1283. IEEE, 2020.

[21]. Rashmi, R. Patil, and S. M. Sangve. "Public auditing system: Improved remote data possession checking protocol for secure cloud storage." In 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 75-80. IEEE, 2015.

Cite this article as :

Rashmi Ashtagi, Jyoti R. Maranur, Afsha Akkalkot, Shweta M Madiwal, Sridevi Hosmani, Arunadevi Khaple, "Competitive Analysis for the Auditing Cloud Consistency", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 3, pp. 408-414, May-June 2022. Available at doi : https://doi.org/10.32628/IJSRST229396

Journal URL : https://ijsrst.com/IJSRST229396