ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785
(ISRA), Journal Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

EVALUATION OF CLOUD CONSISTENCY USING AUDITING ALGORITHMS

Darakshan Ahmer*, Sameena Banu
[*] Department Of Computer Science & Enigneering , Khaja Banda Nawaz College of Engineering, India

ABSTRACT

Now-a-days, due to advantages cloud storage services become popular. Multiple replicas are stored over a wide range of geographically distributed cloud servers in order to provide always-on access service. The main drawback of such a replication technique is that it requires a high cost to achieve a strong consistency and most of the cloud server's memory is wasted. We are proposing consistency as a service model (CaaS) consisting of data cloud maintained by cloud service provider (CSP) and audit cloud, also implementing a two-level auditing structure. We design algorithms to quantify the severity of consistency violations using two metrics commonality and staleness. Finally we propose a heuristic auditing strategy to know how many violations are possible.

KEYWORDS: Cloud storage, consistency as a service (CaaS), two-level auditing, heuristic auditing strategy(HAS).

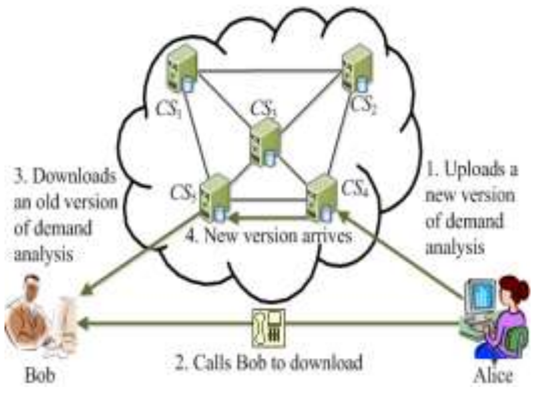
INTRODUCTION

Cloud computing is internet based computing in which large groups of remote servers are networked to allow sharing of data processing tasks, data storage centralization and 24/7 online access to computer services or resources. Cloud computing is becoming commercially popular as it guarantees to provide scalability, elasticity and high availability at a low cost[1],[2]. Now-a-days in order to meet the promise of ubiquitous 24/7 operations, internet applications often rely on globally distributed highly available storage systems. In cloud computing the typical service is regarded as providing cloud storage services, involving the delivery of data storage as a service. A cloud service provider (CSP) maintains multiple replicas for each piece of data on geographically distributed servers to provide ubiquitous always-on access. As such it is very expensive to achieve strong consistency on a world wide scale which is the key problem of using the replication technique in clouds. In this we are proposing consistency as a service model (CaaS) consisting of data cloud maintained by cloud service provider (CSP) and audit cloud ,also implementing a two-level auditing structure. We also design algorithms to quantify the severity of consistency violations using two metrics commonality and staleness. Finally we propose a heuristic auditing strategy to know how many violations are possible.

EXISTING SYSTEM

In cloud computing , cloud storage services are regarded as a typical service which involves the delivery of data storage as service, including database-like services and network attached storage, often billed on a utility computing basis, how much we use that much to pay. Examples include Amazon, Microsoft Azure, and so on. By using the cloud storage services in cloud computing, the customers can access data stored in a cloud anytime and anywhere using any device. To meet the promise of ubiquitous 24/7 access, the cloud service provider (CSP) stores data replicas on multiple geographically distributed servers. The problem of using the replicas is that it is very cost effective to obtain strong consistency over a worldwide scale, where a user is expecting to view the latest updates. Stronger consistency is of more importance for some of the interactive applications. Especially for the interactive applications, stronger consistency assurance is of increasing importance. Consider the following scenario as shown in Fig.1

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785
(ISRA), Journal Impact Factor: 2.114


CS_1
CS_3
CS_2
CS_5
CS_4
3. Downloads an old version of demand analysis
1. Uploads a new version of demand analysis
4. New version arrives
2. Calls Bob to download
Bob
Alice
Fig. 1. An application that require causal consistency.

Suppose that Alice and Bob are cooperating on a project using a cloud storage service, where all of the related data is replicated to five cloud servers, CS1,CS2,CS3,CS4,CS5. After uploading a new version of the requirement analysis to a CS4, Alice calls Bob to download the latest version for integrated design. Here, after Alice calls Bob, the causal relationship is established between Alice's update and Bob's read. Therefore, the cloud should provide causal consistency, which ensures that Alice's update is committed to all of the replicas before Bob's read. If the cloud provides only eventual consistency, then Bob is allowed to access an old version of the requirement analysis from CS5. Actually, different applications have different consistency requirements. For example, mail services need monotonic read consistency and read-your-write consistency, but social network services need causal consistency [3]. Overcomes of this system are ,it is very expensive, the users are enable to see the last update. The replication technique in clouds is that it is very expensive to achieve strong consistency. Hard to verify replica in the data cloud is the latest one or not.

PROPOSED SYSTEM

In reality, different applications require different requirements. For example, mailing services require monotonic-read consistency and read-your-write consistency whereas social networking services require causal consistency. Apart from correctness, in cloud storage consistency also determines the actual cost per transaction. In proposed system we are introducing a novel consistency as a service (CaaS) model for these different applications. The CaaS model constitutes of a large data cloud and multiple small audit clouds. The cloud service provider(CSP) maintains the data cloud whereas an audit cloud consists of group of users that work on a document or a project. The data cloud and audit cloud signs an agreement known as service level agreement (SLA) before doing any job. The SLA ensures what level of consistency must be provided by the data cloud and what will be the cost if any violations by data cloud. A trace of interactive operations are used which allows the users in verifying the consistency in a cloud storage [4]. For this a loosely synchronized clock is used. A two-level auditing structure is developed where each user can perform local auditing and global auditing. Local auditing mainly focus on monotonic-read consistency and read-your-write consistency whereas global auditing focus on causal consistency. . We quantify the severity of violations by two metrics for the CaaS model: commonality of violations and staleness of the value of a read, as in [5]. Finally, we propose a heuristic auditing strategy (HAS) which adds appropriate reads to reveal as many violations as possible.
Our key offerings are as follows:

1) We introduce a consistency as a service (CaaS) model, where a group of users that constitute an audit cloud can verify whether the data cloud provides the promised level of consistency or not.
2) Next a two-level auditing structure is proposed, which only requires a loosely synchronized clock for ordering operations in an audit cloud.
3) Also we design algorithms to quantify the severity of violations with different metrics.
4) We devise a heuristic auditing strategy (HAS) to know how many violations are possible. The benefits of this systems are, due to unique ID's the audit cloud can be easily identified, any user can read and update the data from anywhere using any data, it is less expensive as the replication technique is used only at the user not at the cloud servers. Do not require a global clock among all users for total ordering of operations. The users can assess the quality of cloud services. choose a right CSP Among various candidates, e.g, the least expensive one that still provides adequate consistency for the users' applications.

RELATED WORK

(A. Tanenbaum and M.V. Steen, 2002) Basically A Tanenbaum and M. Van Steen differentiates between two classes of consistency models are: data-centric and client-centric consistency. Data-centric consistency model considers the internal state of a storage system, i.e., how updates flow through the system and what guarantees the system will provide with respect to latest updates. While a, client-centric consistency model usually focus on what cloud storage customers specifically want, i.e., how the customers will observe data updates. It also describes different levels of consistencies such as strict consistency & weak consistency. Weak consistency models are used in consistency checking such as distributed shared memory, distributed transactions etc. High consistency implies high cost and reduced availability.

(E.Anderson,X.Li,M.Shah,J.Tucek and J.Wylie,2010) Anderson, Li,Shah,and Wylie considers how to measure and assess the consistency observed by the customers when using key value storage systems. For a given a trace of operations, our approach is to check whether a trace satisfies safe, regular, or atomic semantics. We first construct a directed graph called the precedence graph, here a vertex represents an operation and an edge indicates that the source operation that should happen before the target operation. We add three kinds of edges: time, data, and hybrid. If the resulting graph is acyclic directed graph then we conclude consistency is achieved or consistency is violated. To conclude whether the graph is acyclic or no we perform a depth-first-search (DFS) algorithm.

(W.Golab,X.Li and M.Shah,2011,ACM PODC.) Golab, Li & Shah has proposed an online verification algorithm to quantify the severity of violations. The main drawback of the existing trace-based verifications is that it requires a global clock among all of its users. Our solution belongs to a trace-based verifications. However, our focus is on different consistency semantics in commercially available cloud systems, where a loosely synchronized clock is suitable for our solution.

(P. Mell and T. Grance, (Draft), 2011.) Cloud computing is a model for enabling ubiquitous always-on, convenient, provides a network access based on a shared pool of computing resources such as, servers, storage, applications, and services only on demand of a user that can be rapidly provisioned and released with minimal management effort or service provider interaction.

(W. Lloyd, M. Freedman, M. Kaminsky, and D. Andersen,. 2011 ACM SOSP.) Brewer's CAP principle states that any shared data system can provide only two of the following three properties: consistency, availability, and partition tolerance to provide an "always on" user experience, scalability to adapt to increasing load and storage demands, refereeing the systems with these four properties as ALPS systems. Traditional systems such as databases and file-systems select to sacrifice availability and only offer strict consistency. Most of the Internet services such as social networks or media-sharing sites favor availability over consistency.

SYSTEM ARCHITECTURE

The system architecture can be clearly explained as follows: First consistency as a service (CaaS) model is illustrated; next we describe the structure of the User operation table (UOT) in which each user maintains a record for the operation performed and finally an overview of the two-level auditing structure.

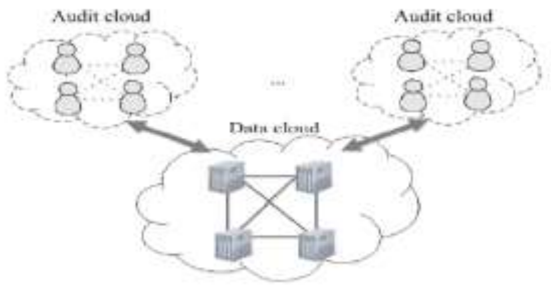


Fig.2. Consistency as a service model.
1 Consistency as a service (CaaS) model.

The CaaS model consists of a large data cloud and small multiple audit clouds as shown in fig 2. The cloud service provider (CSP) maintains the data cloud which is a key-value data storage system where the users upload their files. In a data cloud each piece of data is recognized by a unique identifier. The CSP maintains replicas for each piece of data on a multiple distributed cloud servers in order to provide always on services. In this model an audit cloud consists of group of users which work on a project or a document. In an audit cloud each user is identified by an unique identifier. Here the data cloud and the audit cloud sign a service-level agreement (SLA) before performing any task. The SLA indicates what level of consistency should a data cloud provides and does it violates any consistency. The audit cloud checks whether the data cloud violates the SLA or not and to quantify the severity of violations if any. In this system a two-level auditing model is used which includes Local auditing and Global auditing. In this model each user maintains a user operation table where they record every operation carried referred to as local trace of operations. Each user can perform local auditing independently with its own UOT, an auditor is selected from the audit cloud periodically. In global auditing, global trace-of operations are performed where all other users send their UOTs to the auditor. The dotted line indicates users are loosely connected meaning users will communicate with each other to exchange the messages only after the execution of a set of reads and writes rather than communicating immediately.

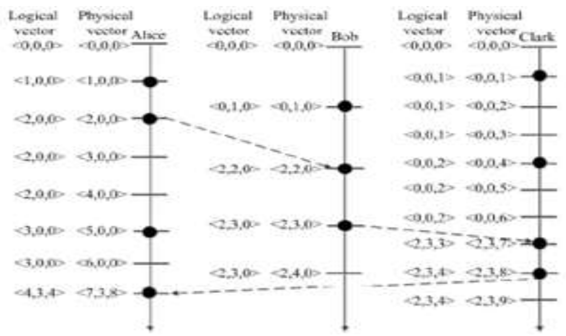


Fig.3. The update process of Logical and Physical Vector.A black solid circle denotes an event (read/write/send message/receive message),and the arrows from top to bottom denote the increase of physical time.

User Operation Table (UOT):
User operation table is maintained by every user for recording their local operations. In the UOT each record is described by three elements: operation, logical vector and physical vector where, each op operation performed by a user is either a write W(K,a) or a read R(K,a), where W(K,a) specifies writing a value a to data that is identified by key K and R(K,a) specifies reading data that is identified by key K whose value is a. As in [4], we read W(K,a) as R(K,a)'s dictating write, and R(K,a) as W(K,a)'s dictated read.
Therefore the following properties have to be considered.

1) Every read operation must have a unique write. Whereas a write can have zero or more reads.
2) The logical and physical vectors can be identified by knowing the value of a read.

TABLE I Alice's Operation table

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>

TABLE II Bob's Operation table

Operation	Logical Vector	Physical Vector
W(a)	<,0,1,0>	<0,1,0>
W(b)	<2,4,0>	<2,5,0>
R(b)	<2,5,0>	<2,6,0>

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785
(ISRA), Journal Impact Factor: 2.114

TABLE III Clark's Operation table

Operation	*Logical Vector*	*Physical Vector*
W(a)	*<0,0,1>*	*<0,0,1>*
W(b)	*<0,0,2>*	*<0,0,4>*
R(b)	*<2,3,5>*	*<2,3,10>*

Overview of two-level auditing structure:
Following the work of [7], a two-level auditing structure is adopted for a CaaS model. In the Ist level, every user performs a local auditing with its own UOT independently. At this level the following consistencies need to be verified.

Monotonic-read consistency: Means is a process has seen a particular value for the object any subsequent access will never return any previous values.

Read-Your- write consistency: It requires that a user always reads his latest updates.

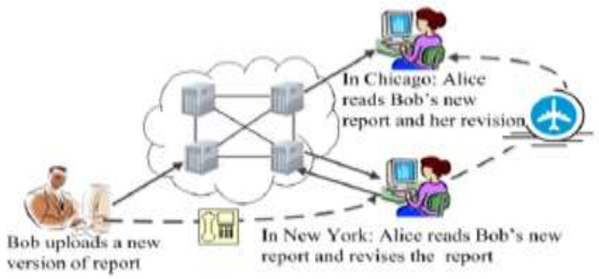

In Chicago: Alice reads Bob's new report and her revision
Bob uploads a new version of report
In New York: Alice reads Bob's new report and revises the report

Fig.4. An application that has different consistency requirements.

Let us consider the example shown in fig 4. from the above figure it is clear that Alice travels from New York to Chicago to work. Hence the CSP maintains replicas on cloud servers in New York and Chicago to provide high availability. In fig.4. Alice reads the new version of report uploaded by Bob and updates the same and after this Alice travels to Chicago. In Chicago Monotonic-read consistency requires that Alice must see the new version of Bob. Read-your-write consistency requires that Alice must read her revision for the new report. In the second level, global auditing is performed by an auditor where the following consistency is checked.

Causal Consistency: When all processes see the records in the same order causal consistency is achieved.

ALGORITHMS

Local Consistency Auditing

Local consistency auditing is an online algorithm. In this module, each user will record all of his operations in his UOT. While issuing a read operation, the user will perform local consistency auditing independently.

Let $R(a)$ denote a user's current read whose dictating write is $W(a)$, $W(b)$ denote the last write in the UOT, and $R(c)$ denote the last read in the UOT whose dictating write is $W(c)$. Read-your-write consistency is violated if $W(a)$ happens before $W(b)$, and monotonic-read consistency is violated if $W(a)$ happens before $W(c)$. Note that, from the value of a read, we can know the logical vector and physical vector of its dictating write. Therefore, we can order the dictating writes by their logical vectors.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785
(ISRA), Journal Impact Factor: 2.114

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
```

Fig. 2. If the first event for Alice is W(K, a), the first record in Alice's UOT is [W(K, a),< 1, 0, 0 >,< 1, 0, 0 >].This means that Alice writes value a to data identified by key K when both her physical and logical clocks are 1.
To illustrate, let us suppose that there are three users in the audit cloud, Alice, Bob, and Clark, where *IDAlice* < *IDBob* < *IDClark*. Each user may update the vectors as shown in

Global Consistency Auditing
Global consistency auditing is an offline algorithm (Alg. 2). Periodically, an auditor will be elected from the audit cloud to perform global consistency auditing. In this case, all other users will send their UOTs to the auditor for obtaining a global trace of operations. After executing global auditing, the auditor will send auditing results as well as its vectors to all other users. Given the auditor's vectors, each user will know other users' latest clocks up to global auditing. From [4], we verify consistency by constructing a directed graph based on the global trace. We claim that causal consistency is preserved if and only if the constructed graph is a directed acyclic graph (DAG).

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
```

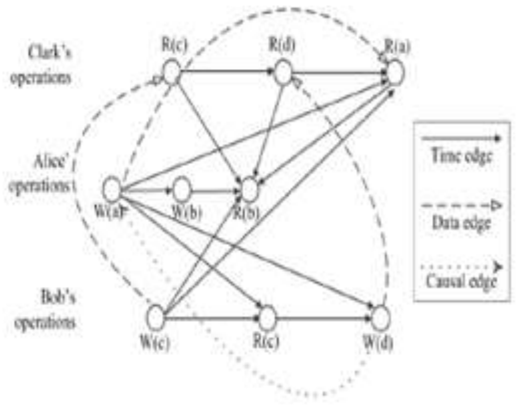


Fig.5. Sample graph constructed with Alg.2.

In Alg.2, each operation is denoted by a vertex. Then, three kinds of directed edges are added by the following rules:

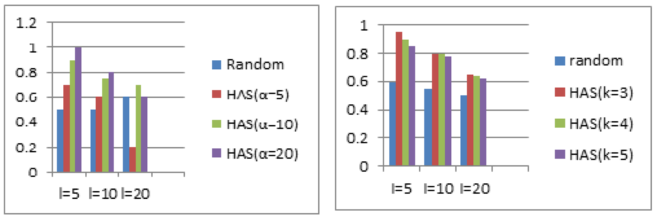
1) Time edge. For operation *op*1 and *op*2, if *op*1 → *op*2, then a directed edge is added from *op*1 to *op*2.
2) Data edge. For operations *R*(*a*) and *W*(*a*) that come from different users, a directed edge is added from *W*(*a*) to *R*(*a*).
3) Causal edge. For operations *W*(*a*) and *W*(*b*) that come from different users, if *W*(*a*) is on the route from *W*(*b*) to *R*(*b*), then a directed edge is added from *W*(*a*) to *W*(*b*).

GARBAGE COLLECTION

In the auditing process, each user has to keep all operations in his UOT. With this, the size of the UOT would grow indefinitely. Additionally, the for transferring cost of the UOT to the auditor will be excessive.
Therefore, a garbage collection mechanism is provided which can delete unwanted records. In this mechanism, each user can clear the UOT, keeping only his last read and last write, after each global consistency verification. This makes sure that a user's last write and last read will always exist in his UOT.

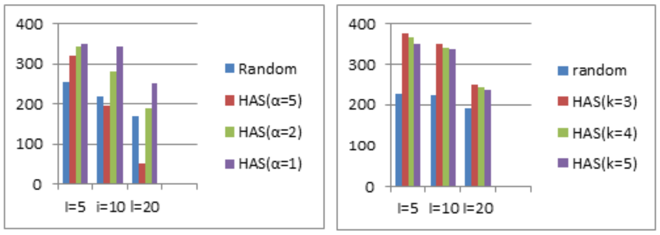
EVALUATION

Fig.6. Comparison of percentage of revealed violations.



(a) ***(b)***

Fig.7. Comparison of earned profit.



(a) ***(b)***

CONCLUSION

Here we have presented a model, consistency as a service (CaaS) model and to help the users to verify whether the cloud service provider (CSP) provides the promised level of consistency or not , and to quantify the severity of violations if any by using a two-level auditing structure. The CaaS model provides the users to assess the quality of cloud services and also helps the users in selecting the right CSP among various cloud service providers. In other words selecting the least expensive but still providing the adequate consistency for the users applications. A thorough theoretical study on consistency models in cloud computing will be conducted and achieving strong consistency in distributed server and will generate individual report system to the user to identify consistency status of their own files.

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] 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.
[4] 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.
[5] W. Golab, X. Li, and M. Shah, “Analyzing consistency properties for fun and profit,” in Proc. 2011 ACM PODC.
[6] A. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms. Prentice Hall PTR, 2002
[7] ——, “Eventually consistent,” Commun. ACM, vol. 52, no. 1, 2009.
[8] T. Gormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms. MIT Press, 1990.
[9] M. Rahman, W. Golab, A. AuYoung, K. Keeton, and J. Wylie, “Toward a principled framework for benchmarking consistency,” in Proc. 2012 Workshop on HotDep.
[10] D. Kossmann, T. Kraska, and S. Loesing, “An evaluation of alternative architectures for transaction processing in the cloud,” in Proc. 2010 ACM SIGMOD.
[11] J. Misra, “Axioms for memory access in asynchronous hardware systems,” ACM Trans. Programming Languages and Systems, vol. 8, no. 1, 1986.
[12] P. Gibbons and E. Korach, “Testing shared memories,” SIAM J. Computing, vol. 26, no. 4, 1997.