ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 3, March 2015

Maintaining Secrecy of Users Data from World Scrutinizing In Cloud Storage

Gurupandi.P[1], Vetrithangam.D[2]
M.E, CSE, RVS College of Engineering and Technology, Dindigul, India [1]
Associate Professor, CSE, RVS College of Engineering and Technology, Dindigul, India [2]

Abstract: Cloud storage services have become commercially popular due to their overwhelming advantages to provide ubiquitous always-on access; a cloud service provider maintains multiple replicas for each piece of data on multiple distributed servers. A key problem of using the replication technique, which is nothing but master slave combinations of databases in clouds is that it is very expensive to achieve strong consistency on a worldwide scale. So, this system advise a heuristic auditing strategy (HAS) to reveal as many violations as possible. Cloud storage is a common place for data to be not only stored but also shared across multiple users. Unfortunately, the integrity of cloud data is subject to uncertainty due to the existence of hardware/software failures and human errors. User operation table have been generated to allow both data owners and public verifiers to efficiently audit cloud data integrity without retrieving the entire data from the cloud server. This System proposes a novel privacy-preserving mechanism that supports public auditing on shared data stored in the cloud.

Index Terms—Cloud storage, heuristic auditing strategy (HAS).

I. Introduction

Cloud storage services can be responsible as a typical service in cloud computing, which involves the delivery of data storage as a service, including database-like services and network attached storage, often billed on a utility computing basis. Examples include Amazon SimpleDB1, Microsoft Azure storage2 and so on. By using the cloud storage services, the customers can access data stored in a cloud anytime and anywhere, using any device, without caring about a large amount of capital investment when deploying the underlying hardware infrastructures.

In cloud computing paradigm it is not only used to store the user's data and also allows the users to share the data among them. Sometimes the integrity of cloud data is loss due to the existence of hardware/software failures and human errors. To prevent this problem several mechanisms have been designed to allow both data owners and public verifiers to efficiently audit cloud data integrity without retrieving the entire data from the cloud server. However, public auditing on the integrity of shared data with these existing mechanisms will inevitably reveal confidential information identity privacy to public verifiers. This system proposes a novel privacy-preserving mechanism that supports public auditing on shared data stored in the cloud. identity of the signer on each block in shared data is kept private from public verifiers, who are able to efficiently verify shared data integrity without retrieving the entire file. This system technique is able to perform multiple auditing tasks simultaneously instead of verifying them one by one.

The main scope of this project to solve the above privacy issue on shared data, this systems propose a novel privacy preserving public auditing mechanism. more specifically, this system utilize ring signatures to construct homomorphism authenticators in Orate, so that a public verifier is able to verify the integrity of shared data without retrieving the entire data while the identity of the signer on each block in shared data is kept private from the public verifier. Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service, Platform-as-a-Service and Software-as-a-Service. The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.

Infrastructure as a service is a provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining it, Platform as a Service is a way to rent hardware, operating systems and storage and network capacity over the Internet. The service delivery model allows the customer to rent virtualized servers and associated services for running existing applications or developing and testing new ones, Software as a Service is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet.

II. Related Work

***Badrul Philip YimKwong Cheng** [1]* proposed the system first explain how the heuristics of availability, representativeness and anchoring-and-adjustment could have unfavorable impact on auditors' judgment and decisions, followed by discussions of regency and dilution effects, this system then introduce the properties of the two systems of reasoning put forward by cognitive scientists and referred to System one and System two hereon in the article. Against this background, this system proposes and illustrates with examples an easily implemented dual systems cognitive model. author conclude project with a few directions for future research in a new frontier of behavioural audit research, heuristics are "mental" shortcuts people commonly used to help making decisions or forming judgments, particularly when facing incomplete information or complex problems. People use availability heuristic when they "assess the frequency of a class or the probability of an event by the ease with which instances or occurrence can be brought to mind" had a similar opinion: judgment can be based on both the content of accessible judgment-relevant information and the subjective ease with which this information comes to mind. When an auditor is asked what the minimum sample size is, how often the magic number of 30 comes up as the answer? The number 30 is the minimal sample size and consequently many auditors simply use 30 as the sample size, without going through the process of determining the confidence level, tolerable and expected errors and so forth in working out the minimum sample size. As more and more auditors are using 30 as the minimum sample size, it becomes so popular that it becomes a generally accepted "doctrine" among auditors.

BadrulSarwar, George Karypis, Joseph Konstan and John Riedl [2] proposed that eventually-consistent key-value storage systems sacrifice the ACID semantics of conventional databases to achieve superior latency and availability. However, this means that client applications and hence end-users can be exposed to stale data. The degree of staleness observed depends on various tuning knobs set by application developers (customers of key-value stores) and system administrators (providers of key-value stores). Both parties must be cognizant of how these tuning knobs affect the consistency observed by client applications in the interest of both providing the best end-user experience and maximizing revenues for storage providers. Quantifying consistency in a meaningful way does a critical step toward both understand what clients actually observe and supporting consistency-aware service level agreements (SLAs) in next generation storage systems? Many cloud products and services such as Web search, e-commerce and social networking, have to deal with big data. In order to scale with growing amounts of data and numbers of users, the design of these systems has moved away from using conventional ACID databases, toward a new generation of scalable storage systems called key-value stores (often categorized more broadly as NoSQL storage systems). Many of these key-value storage systems offer a weak notion of consistency called eventual consistency. These items from Brewer's CAP principle, which dictates that a storage system chooses between either consistency or availability during failures that partition the network connecting the storage nodes.

G.DeCandia, D.Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels [5] proposed that reliability at massive scale is one of the biggest challenges this system face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems, Amazon runs a world-wide e-commerce platform that serves tens of millions customers at peak times using tens of thousands of servers located in many data centers around the world. There are strict operational requirements on Amazon's platform in terms of performance, reliability and efficiency and to support continuous growth the platform needs to be highly scalable. Reliability is one of the most important requirements because even the slightest outage has significant financial consequences and impacts

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 3, March 2015

customer trust. In addition, to support continuous growth, the platform needs to be highly scalable.

III. Methodology

This Proposed methodology used HAS technique for auditing and cloud is essentially a large scale distributed system where each piece of data is replicated on multiple distributed servers to achieve high availability and high performance, this system first review the consistency models in distributed systems and it classify two classes of consistency models first is data-centric consistency and second is 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 can provide with respect to updates. However, to a customer, it really does not matter whether or not a storage system internally contains any stale copies. As long as no stale data is observed from the client's point of view then customer is satisfied. A key problem of using the replication technique, replication is nothing but master slave combinations of databases in clouds is that it is very expensive to achieve strong consistency on a worldwide scale. Finally, this system advise a heuristic auditing strategy (HAS) to reveal as many violations as possible. Cloud storage is a common place for data to be not only stored but also shared across multiple users. Unfortunately, the integrity of cloud data is subject to uncertainty due to the existence of hardware/software failures and human errors. User operation table have been generated to allow both data owners and public verifiers to efficiently audit cloud data integrity without retrieving the entire data from the cloud server, Fig.1 illustrates the proposed methodology.

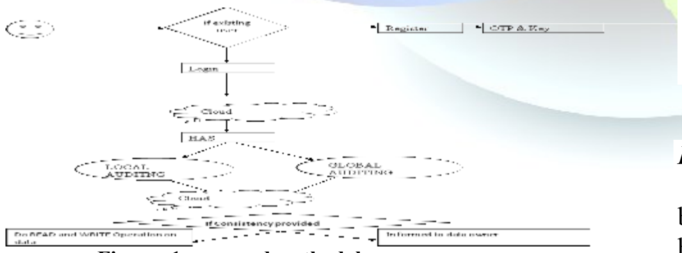


Figure: 1 proposed methodology

CAP THEOREM

CAP is an abbreviation for consistency, availability, and partition tolerance. The basic idea is that in a distributed system, you can have only two of these properties, but not all three at once. Let's look at what each property means.

Consistency

Data access in a distributed database is considered to be consistent when an update written on one node is immediately available on another node. Traditional ways to achieve this in relational database systems are distributed transactions. A write operation is only successful when it's written to a master and at least one slave, or even all nodes in the system. Every subsequent read on any node will always return the data written by the update on all nodes.

Availability

The system guarantees availability for requests even though one or more nodes are down. For any database with just one node, this is impossible to achieve. Even when you add slaves to one master database, there's still the risk of unavailability when the master goes down. The system can still return data for reads, but can't accept writes until the master comes back up. To achieve availability data in a cluster must be replicated to a number of nodes, and every node must be ready to claim master status at any time, with the cluster automatically rebalancing the data set.

Partition Tolerance

Nodes can be physically separated from each other at any given point and for any length of time. The time they're not able to reach each other, due to routing problems, network interface troubles, or firewall issues, is called a network partition. During the partition, all nodes should still be able to serve both read and write requests. Ideally the system automatically reconciles updates as soon as every node can reach every other node again

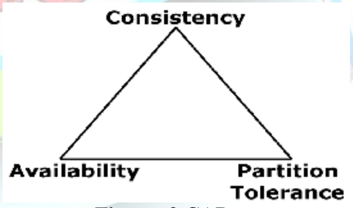


Figure: 2 CAP

Read-your-writes Consistency

A value written by a process on a data item X will be always available to a successive read operation performed by the same process on data item. Each operation op is either a write W (K, a) or a read R(K, a), where W(K, a) means writing the value a to data that is identified by key K, and R(K, a) means reading data that is identified by key K and whose value is a. As system call W(K, a) as R(K, a)'s dictating write, and R(K, a) as W(K, a)'s dictated read. In this paper assume that the value of each write is unique. This is achieved by letting a user attach his ID, and current vectors to the value of write. Therefore, system have the following properties: (1) a read must have a unique dictating write. A write may have zero or more dictated reads. (2) From the value of a read, system can know the logical and physical vectors of its dictating write.

Writes-follows-reads Consistency

In Writes-follow-reads consistency, updates are propagated after performing the previous read operations. For example "A write operation by a process on a data item x following a previous read operation on x by the same process is guaranteed to take place on the same or a more recent value of x that was read.

1) Strong Consistency

Strict consistency is the strongest consistency model. It requires that if a process reads any memory location, the value returned by the read operation is the value written by the most recent write operation to that location

Online Algorithm

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

```
Initial UOT with Ø
While issue an operation op does
If op = W (a) then
Record W (a) in UOT
If op = r (a) then
W (b) ∈ UOT is the last write
If W (a) → W (b) then
Read-your-write consistency is violated
R(c) ∈ UOT is the last read
If W (a) → W(c) then
Monotonic-read consistency is violated
Record r (a) in UOT
```

Offline Algorithm

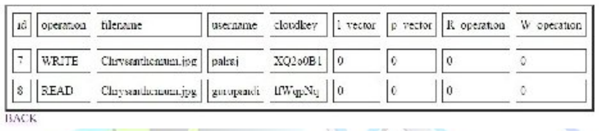
Global consistency auditing is an offline algorithm periodically; an auditor will be elected from the audit cloud to perform global consistency auditing. In this case, all other users will send their UOTs to the auditor for obtaining a global trace of operation.

```
Each operation in the global trace is denoted by a vertex
For any two operations op1 and op2 do
If op1 → op2 then
A time edge is added from op1 to op2
If op1 = W (a), op2 = R (a), and two operations come
From different users then
A data edge is added from op1 to op2
If op1 = W (a), op2 = 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 op1 to op2
Check whether the graph is a DAG by topological sorting
```

IV. Experimental Results

UOT Table

Each user maintains a UOT for recording local operations. Each record in the UOT is described by three elements: *operation*, *logical vector*, and *physical vector*. While issuing an operation, a user will record this operation, as well as his Current logical vector and physical vector, in his UOT. Each operation *op* is either a write *W* (*K, a*) or a read *R* (*K, a*), where *W*(*K, a*) means writing the value *a* to data That is identified by key *K*, and *R*(*K, a*) means reading data that is identified by key *K* and whose value is *a*. As, system call *W* (*K, a*) as *R*(*K, a*)'s *dictating write*, and *R*(*K, a*) as *W*(*K, a*)'s *dictated read*. In this paper assume that the value of each write is unique. This is achieved by letting a user attach his ID, and current vectors to the value of write. Therefore, system have the following properties: (1) A read must have a unique dictating write. A write may have zero or more dictated reads. (2) From the value of a read, user can know the logical and physical vectors of its dictating write.



id	operation	filename	username	cloudkey	l vector	p vector	R operation	W operation
7	WRITE	Chrysanthemum.jpg	palraj	XQ2o0B1	0	0	0	0
8	READ	Chrysanthemum.jpg	guruspandi	lfWgpNg	0	0	0	0

BACK

Figure: 3 UOT Table

Performance Evoluation

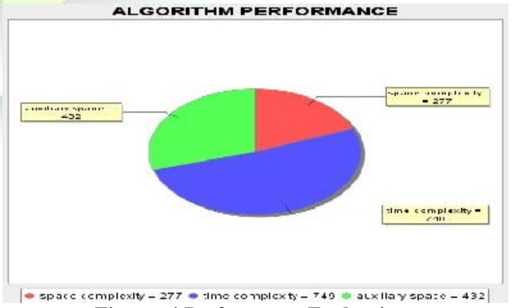


Figure: 4 Performance Evaluations

Performance summarizes the parameters used in the synthetic violation traces in Table II. In the random strategy, users randomly choose [1, l] auditing reads in each interval,

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 3, March 2015

where l is the Length of an interval. To obtain the synthetic violation traces, physical time is divided into 2,000 time slices.user assume that once a data cloud begins to violate the promised consistency, this violation will continue for several time slices, rather than ending immediately. In the simulation, the duration of each violation d is set to 3-10 time slices.

DAG For Consistency

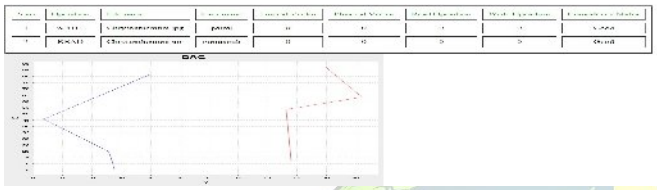


Figure: 5 DAG

V. Conclusion And Future Enhancement

In this paper discussed about a consistency as a service model and a two-level auditing structure to verify the cloud service provider for providing promised consistency and to quantify the severity of the violations if any. With the CaaS model, the users can assess the quality of cloud services and choose a right CSP among various candidates and least expensive one that still provides adequate consistency for the users' applications.

This system will conduct a thorough theoretical study of consistency models in cloud computing and achieving strong consistency in distributed server and will generate individual report system to the user to identify consistency status of their own files.

Reference

[1]. Badrul Philip YimKwong Cheng, "Improving Audit Judgment and Decision Making With Dual Systems Cognitive," in Proc. 2010 USENIX HotDep.

[2]. BadrulSarwar, George Karypis, Joseph Konstan and John Riedl, "Client-centric Benchmarking of Eventual Consistency for Cloud Storage Systems," in Proc. 2010 ACM SOSP.

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

[4]. Davide Gerhard, DavideGirardi, "consistency properties in Amazon SimpleDB and Amazon S3.2011-12.

[5]. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, "Dynamo: Amazon's highly available key-value store," in Proc. 2007 ACM SOSP.

[6]. M Douglas Thain, Todd Tannenbaum and MironLivny," Distributed Computing in Practice: The Condor Experience, vol. 9, no. 1, 2012.

[7]. 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.

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

[9]. Jeremy Lau, Matthew Arnold, Michael Hind, "A Loop Correlation Technique to Improve Performance Auditing," in Proc. 2010 ACSC.

[10]. 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.