An Efficient Auditing System for the Cloud

Ab Waheed Lone[1], Sheikh Riyaz-ul-Haq[2] , Syed Sarver Hussain[3],
[12]*Department of Information Technology, Central University of Kashmir, J&K, India*
[3]*Jammu and Kashmir Bank, Jammu and Kashmir, India*

Abstract- A number of organizations are nowadays migrating their critical information technology services, from healthcare to business intelligence, into public cloud computing environments. Cloud computing has been emerged as a solution to the elevating storage costs of IT industry. However, even if cloud technologies are continuously evolving, they still have not reached a level that allows them to provide users with high data integrity, consistency, and security of their data beyond existent service level agreements. Cloud storage moves the user's data to immensely colossal remotely located data centers, on which user does not have any control. Hence to overcome this issue, we are going to propose approach of service that is Consistency as a Service known as CaaS. The Consistency as a Service (CaaS) model concentrates on; in this we have large data cloud and small multiple audit clouds. In the CaaS model, a CSP maintains the data cloud, audit cloud can verify whether the data cloud provides the promised level of consistency or not. We propose a two-level auditing architecture, which only requires a loosely synchronized clock in the audit cloud. Then, we design algorithms to quantify the severity of violations with two metrics.The data owner can also audit the data integrity in the corresponding cloud for verifying whether the data is safe or not. At last we devise a heuristic auditing strategy (HAS) to reveal as many violations as possible.

Keywords- CSP, CaaS, Audit

I. INTRODUCTION

The popularity of cloud services has attained immense heights with the world moving towards a data centric paradigm. Availability is the biggest factor contributing towards the growth of cloud computing. The cloud service provider (CSP) has to ensure round the clock availability of data. Cloud computing is driving the momentum towards making the database available as a service on the cloud.Database services take care of scalability and high availability of the database.

The CSP stores the different copies of data to immensely colossal remote centers in a distributed manner Data needs to be updated in several locations and a problem thus arises when one or more of these locations are temporarily not accessible. Sharing data and computations over a scalable network of nodes, which will be end users, data centers and web services is the main objective to be achieved.

The public cloud storage services like Amazon S3, Google Cloud Storage and Windows Azure Storage replicate the data to ensure high availability. On the other hand, with data being replicated, the storage services exhibits certain data consistency models. Different cloud service providers employ different data consistency models nowadays. The physical database administration tasks, such as backup, recovery, managing the logs, etc., are managed by the cloud provider. The responsibility for logical administration of the database, including table tuning and query optimisation, rests on the developer. These cloud offerings of database services still use traditional SQL-based database technology, as underlying platform not specifically reinvented for the cloud..

A consistency of type called as eventual consistency is provided by many cloud service providers. Here a user can read the data for particular time. Now-a-days stronger consistency assurance is getting importance. Consider the following figure. [1]

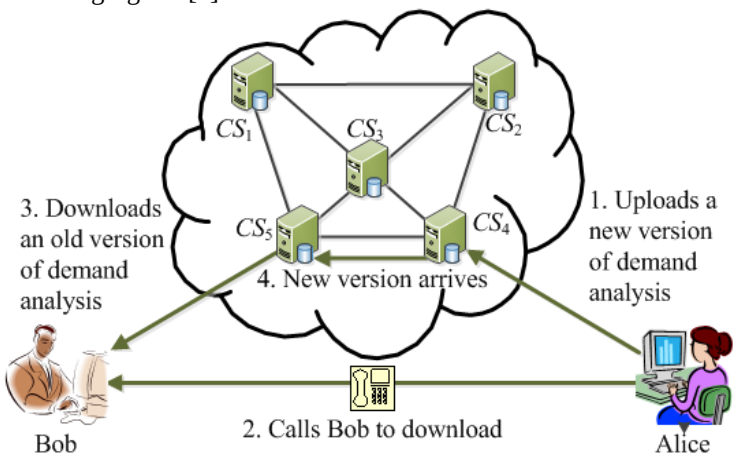

Fig.1: Example to show casual consistency

Here data is stored in multiple copies on five cloud servers (CS1, CS2…, CS5), users specified in the figure share data through a cloud storage service. The cloud should be a provider of casual consistency service. Alice uploads a data on the cloud server CS4. This update should be reflected in all servers. If only eventual consistency is maintained then the receiver is going to receive only the old version of data. Such integrated design based on traditional version may not satisfy customer requirements thus strong consistency is needed

II. DATA CONSISTENCY AND INTEGRITY ON THE CLOUD

Data consistency and integrity are two important aspects that need to be ensured in all types of databases including the cloud database.

Data consistency implies that all instances of an application are presented with the same set of data values all of the time. This is sometimes referred to as strong data consistency.Cloud applications typically use data that is dispersed across different data stores. Managing and maintaining data consistency in such an environment is a critical aspect. Concurrency and availability are the issues faced during this. Strong consistency needs to be traded for availability. This leads to the need of designing solutions around the notion of eventual consistency and accept that the data might not be completely consistent all of the time.

Maintaining data consistency across distributed data stores which may be geographically at different locations is a difficult task. Strategies such as serialization and locking only work well if all application instances share the same data store, and the application is designed to ensure that the locks are very short-lived. If the data is partitioned or replicated across different data stores, locking and serializing data access to maintain consistency can become an expensive overhead that impacts the throughput, response time, and scalability of a system. Therefore, most modern distributed applications do not lock the data that they modify, and they take a rather more relaxed approach to consistency, known as eventual consistency.

A. Strong Consistency

All the changes are atomic. If a transaction updates multiple data items, the transaction is not allowed to complete until either all of the changes have been made successfully or have all been undone. The aim of the strong consistency model is to minimize the chance that an application instance might be presented with an inconsistent view of the data. In a distributed environment, if the data stores holding the data affected by a transaction are geographically remote from each other, network latency could adversely impact the performance of such transactions and result in concurrent access to data being blocked for an extended period. If a network failure renders one or more of the data stores inaccessible during a transaction, an application updating data in a system that implements strong consistency may be blocked until every data store becomes accessible again.

In a distributed environment such as the cloud, implementing strong consistency is not tolerant of the types of failure that may occur. For example, it may not be possible to roll back a transaction and release the resources that it holds if a component participating in the transaction has stopped responding due to a long-lasting network outage. In this case, it will be necessary to resolve the situation through other means, such as manually reconciling the data.

B. Eventual Consistency

In many cases, strong consistency is not actually required as long all the work performed by a transaction is completed or rolled back at some point, and no updates are lost. In the eventual consistency model, data update operations that span multiple sites can ripple through the various data stores in their own time, without blocking concurrent application instances that access the same data. One of the drives for eventual consistency is that distributed data stores are subject to the CAP Theorem. This theorem states that a distributed system can implement only two of the three features (Consistency, Availability, and Partition Tolerance) at any one time. integrity ensures data is recorded exactly as intended.data is the same as it was when it was originally recorded. In short, data integrity aims to prevent unintentional changes to information.

(I) Data consistency models of providers

A. Amazon S3 Data Consistency Model

Amazon S3 provides read-after-write consistency for PUTS of new objects in your S3 bucket in all regions with one caveat. The caveat is that if you make a HEAD or GET request to the key name (to find if the object exists) before creating the object, Amazon S3 provides eventual consistency for read-after-write.

Amazon S3 offers eventual consistency for overwrite PUTS and DELETES in all regions.

Updates to a single key are atomic. For example, if you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never write corrupted or partial data.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However, information about the changes must replicate across Amazon S3, which can take some time, and so you might observe the following behaviors:

1. A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
2. A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
3. A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.
4. A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.

B. Google cloud consistency

Strongly consistent operations Cloud Storage provides strong global consistency for the following operations, including both data and metadata:

Read-after-write
Read-after-metadata-update
Read-after-delete
Bucket listing
Object listing
Granting access to resources

When you upload an object to Cloud Storage, and you receive a success response, the object is immediately available for download and metadata operations from any location where Google offers service. This is true whether you create a new object or overwrite an existing object. Because uploads are strongly consistent, you will never receive a 404 Not Found response or stale data for a read-after-write or read-after-metadata-update operation.In addition, when an upload request succeeds, it means your data is replicated in multiple data centers. The latency for writing to Cloud Storage's globally consistent, replicated store may be slightly higher than for a non-replicated or non-committed store. This is because a success response is returned only when multiple writes complete, not just one. Strong global consistency also extends to deletion operations on objects. If a deletion request succeeds, an immediate attempt to download the object or its metadata will result in a 404 Not Found status code. You get the 404 error because the object no longer exists after the delete operation succeeds.Bucket listing is strongly consistent. For example, if you create a bucket, then immediately perform a list buckets operation, the new bucket appears in the returned list of buckets. Object listing is also strongly consistent. For example, if you upload an object to a bucket and then immediately perform a list objects operation, the new object appears in the returned list of objects.

C. Windows Azure consistency model

In WAS, data is stored durably using both local and geographic replication to facilitate disaster recovery. Currently, WAS storage comes in the form of Blobs (files), Tables (structured storage), and Queues (message delivery). In this paper, we describe the WAS architecture, global namespace, and data model, as well as its resource provisioning,load balancing, and replication systems

III. LITERATURE REVIEW

Integrity of data is quit important the main aim of the existing system is to provide verification for integrity of different data storage systems, the problem of supporting both public audit ability and data dynamics has still not been solved or addressed. The existing system follows eventual consistency that is there is no dynamic updating of data, the customers can access data stored in a cloud anytime and anywhere, without actually caring about a substantial amount of capital investment when deploying the underlying hardware infrastructures. The updates done to a name will not be visible immediately in the system. The user won't be able to see them, but the system where the clients are working with the system, have to make sure they are going to see them eventually.[2]

Major disadvantages of the existing system [3] :-

(a).The infrastructure under the cloud are still facing the broad range of internal and external threats for security and data integrity although being more powerful and reliable than computing devices.

(b). It is not a practical solution to simply download all the data for testing integrity of data the reason being the Expensiveness in transmission (I/o).

(c). User cannot see the latest updates. [4]

IV. PROPOSED SYSTEM

We have proposed a standard model called as a consistency technique defined as CaaSmodel. Here the proposed model follows a two-level efficient auditing structure. This aids the users in checking whether the cloud service provider (CSP) guarantees consistency service. It also enables one to express the severity of the violations, present if any. Further with the CaaS model, the system users can take the decision to choose a right CSP in the various candidates present and will be able to assess the quality work of cloud services. Example the cloud service which provides less expensive one operation but able to provide strong consistency for the applications of users.

Our key contributions are as follows:

1) A novel consistency as a service (CaaS) model has been presented

2) A two-level auditing structure proposes

3) Algorithms to quantify the severity of violations with different metrics have been designed

4) A heuristic auditing strategy (HAS) to reveal as many violations as possible.

The advantages offered include, cloud consistency and an efficient auditing item set result based on the CaaS obtained. Cloud consistency has become an inevitable part as it is playing an increasingly important role in the decision support activity of every walk of life.

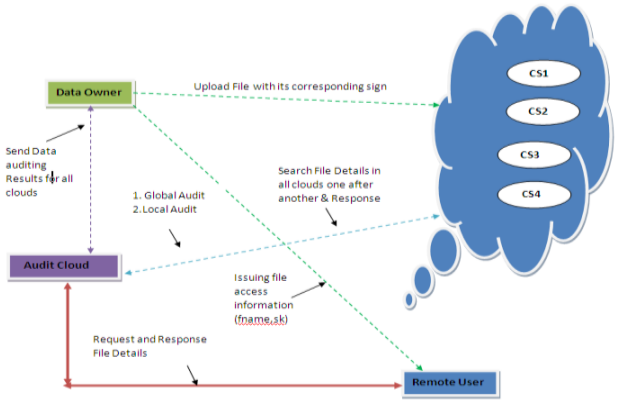

Fig.2: System architecture

A. Data owner

The owner has privileges to upload the data on cloud server. For the security purpose the data is encrypted the data file is then stored in the cloud. The Data owner thus can manipulate the encrypted data file which is being uploaded to the cloud. As shown in level 0 The data is send to audit cloud the audit cloud can check data integrity and can create end users and set permissions (read and write) to user.

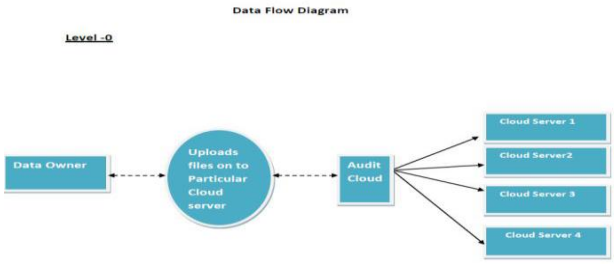

Fig.3: Level 0 diagram

B. Data Consumer

The end user sends a request and gets file contents response from the corresponding cloud servers. The audit cloud checks the file name and secret key, access permission if its correct then the end is getting the file response from the cloud if there is no match he will be considered as an attacker and also can be blocked in corresponding cloud. The end user request for the required file by using file name and secret key to the audit cloud. The audit cloud verifies the user details such as file name and secret key. If the given filename and secret key is correct it would allow the user to access & authorize the file .If there is no match of file name and secret key then the user cannot access the file.

The audit cloud performs auditing i.e local auditing and global auditing. UOT or the user operation table has operations of each user records. This is also referred to as a local trace of operations in this paper. Each user can perform local auditing independently with his own UOT; periodically, an auditor is elected from the audit cloud. In this case, once UOTs are updated all other users will send their UOTs to the auditor, which will perform *global auditing* .We, simply let each user become an auditor in turn, that will provide a more comprehensive solution.

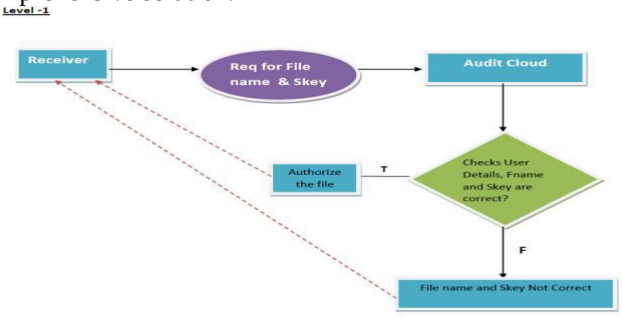

Fig.4: Level 1diagram

C. User Operation Table (UOT)

For recording local operations each user maintains UOT. there are mainly three elements: *operation whether read or write*, *logical vector (event timestamp)*, and physical *vector(physical clock)*.[5] When an operation is issued, a user will record this operation, as well as his current logical vector and physical vector, in his UOT.

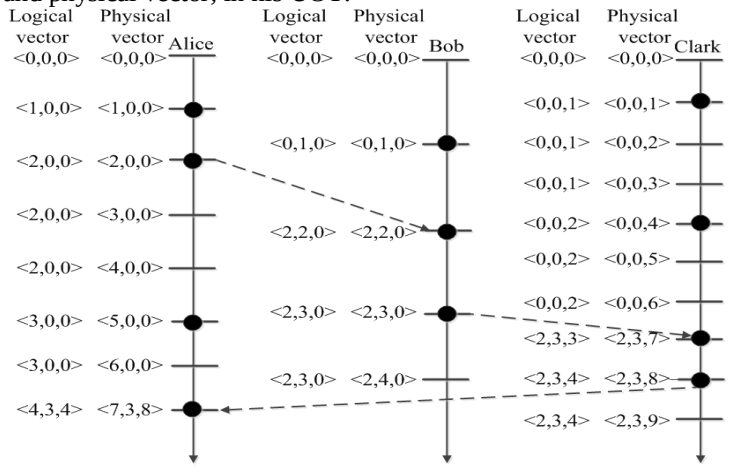

Fig.5: user records this operation, as well as his current logical vector and physical vector, in his UOT

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

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 $op_1 \rightarrow op_2$ **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

The UOT after performing the local auditing and global auditing for various operations looks like

Alice's Operation Log			Bob's Operation Log			Clark's Operation Log		
Operation	logical vector	physical vector	Operation	logical vector	physical vector	Operation	logical vector	physical vector
$W(a)$	$\langle 1,0,0 \rangle$	$\langle 1,0,0 \rangle$	$W(c)$	$\langle 0,1,0 \rangle$	$\langle 0,1,0 \rangle$	$R(c)$	$\langle 0,0,1 \rangle$	$\langle 0,0,1 \rangle$
$W(b)$	$\langle 3,0,0 \rangle$	$\langle 5,0,0 \rangle$	$R(c)$	$\langle 2,4,0 \rangle$	$\langle 2,5,0 \rangle$	$R(d)$	$\langle 0,0,2 \rangle$	$\langle 0,0,4 \rangle$
$R(b)$	$\langle 5,3,5 \rangle$	$\langle 8,3,7 \rangle$	$W(d)$	$\langle 2,5,0 \rangle$	$\langle 2,6,0 \rangle$	$R(a)$	$\langle 2,3,5 \rangle$	$\langle 2,3,10 \rangle$

Global auditing algorithms will contain the strategy described in the figure

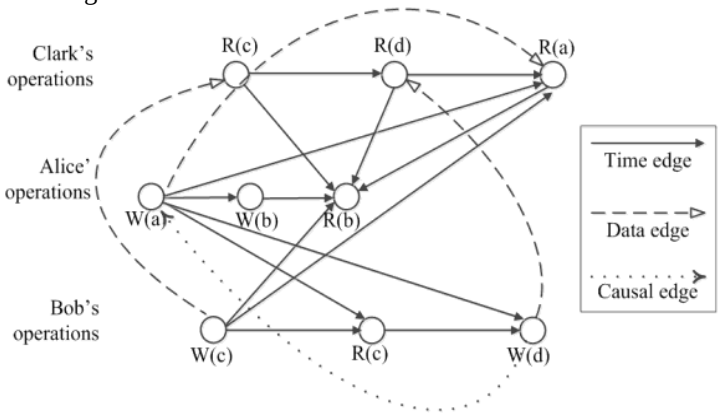

Fig.6: strategy of algorithms

A. Heuristic auditing strategy

Observing from the auditing process we conclude only reads can reveal violations by their values. [7] Therefore, to reveal maximum violations as possible our heuristic auditing strategy (HAS) issued is to add these additional reads are called as auditing reads.[8] The idea behind heuristic auditing strategy (HAS) aims to add appropriate reads for revealing as many violations as possible. [6]

V. RESULTS AND CONCLUSION

The Cloud platform not only enables users to store their data but also provides services for hosting and running their applications, infrastructure and other services. It is an economical platform for users following pay-per-use model. In the corporate and real use world, large number of clients accessing and modify data on the cloud on a daily basis. The real world use leads to the need of maintaining data integrity and consistency. Thus there is a need of a third party auditor (TPA) to achieve this. This paper provided consistency as a service model. The technique proposed uses local auditing and global auditing to check wheather the CSP is going to provide a valid consistency and integrity of data or not. The heuristic auditing strategy is performed to know whether the files are safe or not the main aim is to provide the security of data that is stored on cloud through encrption. The work done is better than the existing systems as it improves security by enabling encryption and leads to improved consistency services. This work can be further expanded in future by conducting an exhaustive theoretical study of consistency models in cloud computing.

VI. REFERENCES

[1]. Blokland, K., Mengerink, J. and Pol, M., 2013. *Testing cloud services: how to test SaaS, PaaS & IaaS*. Rocky Nook, Inc..

[2]. Liu, Q., Wang, G. and Wu, J., 2014. Consistency as a service: Auditing cloud consistency. *IEEE Transactions on Network and Service Management*, *11*(1), pp.25-35.

[3]. Shrinivas, D., 2011. Privacy-preserving public auditing in cloud storage security. *International Journal of computer science nad Information Technologies*, *2*(6), pp.2691-2693.

[4]. Halpert, B., 2011. *Auditing cloud computing: A security and privacy guide* (Vol. 21). John Wiley & Sons.

[5]. Ko, R.K., Jagadpramana, P., Mowbray, M., Pearson, S., Kirchberg, M., Liang, Q. and Lee, B.S., 2011, July. TrustCloud: A framework for accountability and trust in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on* (pp. 584-588). IEEE.

[6]. Wang, C., Ren, K., Lou, W. and Li, J., 2010. Toward publicly auditable secure cloud data storage services. *IEEE network*, *24*(4).

[7]. Li, J., Tan, X., Chen, X. and Wong, D.S., 2013, September. An efficient proof of retrievability with public auditing in cloud computing. In *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on* (pp. 93-98). IEEE.

[8]. Sookhak, M., Talebian, H., Ahmed, E., Gani, A. and Khan, M.K., 2014. A review on remote data auditing in single cloud server: Taxonomy and open issues. *Journal of Network and Computer Applications*, *43*, pp.121-141.

[9]. Bessani, A., Correia, M., Quaresma, B., André, F. and Sousa, P., 2013. DepSky: dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, *9*(4), p.12.

[10]. Mohta, A., Sahu, R.K. and Awasthi, L.K., 2012. Robust data security for cloud while using third party auditor. *International journal of advanced research in computer science and software engineering*, *2*(2).

[11]. Mohta, A. and Awasti, L.K., 2012. Cloud data security while using third party auditor. *International Journal of Scientific & Engineering Research*, *3*(6), p.1.

[12]. Houlihan, R. and Du, X., 2012, December. An effective auditing scheme for cloud computing. In *Global Communications Conference (GLOBECOM), 2012 IEEE* (pp. 1599-1604). IEEE.

[13]. Yang, G., Yu, J., Shen, W., Su, Q., Fu, Z. and Hao, R., 2016. Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability. *Journal of Systems and Software*, *113*, pp.130-139.

[14]. Bhardwaj, R. and Maral, V., 2013. Dynamic Data Storage Auditing Services in Cloud Computing. *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, *2249*, p.8958.

[15]. Duan, Y., Fu, G., Zhou, N., Sun, X., Narendra, N.C. and Hu, B., 2015, June. Everything as a service (XaaS) on the cloud: origins, current and future trends. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on* (pp. 621-628). IEEE.