The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 12 || December - 2014 || Pages || 33-38||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Two Level Auditing Architecture to Maintain Consistent In Cloud

Deepak Batta[1] , P.Srinivasan[2]
[1]*ME-CSE (Final year), Muthayammal Engineering College, Rasipuram*
[2]*Assistant professor, Muthayammal Engineering College, Rasipuram*

ABSTRACT
Confidential data in an enterprise may be illegally accessed through a remote interface provided by a multiple-cloud, or relevant data and archives may be lost or tampered with when they are stored into an uncertain storage pool outside the enterprise. Therefore, it is indispensable for cloud service providers (CSPs) to provide security techniques for managing their storage services. To overcome these problems we present a Consistency as a service auditing cloud scheme. We prove the security of our scheme based data fragmentation on multiple clouds. So the proposed system has data fragmentation, data security and storage on multiple cloud services. We used trusted third party to store the data on multiple cloud and find the data access by untrusted cloud service providers. In this system the client data divide into multiple pieces and send to the multiple clouds with help of trusted third party. If any of the untrusted cloud service providers try modify the data the alert will send to trusted third party about illegal access of untrusted cloud service provider.

Date of Submission: 12 December 2014 Date of Accepted: 05 January 2015

I. INTRODUCTION

Cloud computing has become commercially popular ,as it promises to guarantee scalability, elasticity, and high availability at a low cost , the trend of the everything-as-a-service (XaaS) model, data storages ,virtualized infrastructure, virtualized platforms, as well as software and applications are being provided and consumed as services in the cloud. Cloud storage services can be regarded as a typical service in cloud computing, which involves the delivery of data storage as a service, including data base like services and network attached storage, often billed on a utility computing basis, e.g., per gigabyte per month. Examples include Amazon SimpleDB1, Microsoft Azure storage2, and so on. The usage of 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. To meet the promise of ubiquitous 24/7 access, the cloud service provider (CSP) stores data replicas on multiple geographically distributed servers. A key 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. Actually, mandated by the CAP principle3, many CSPs (e.g., Amazon S3) only ensure weak consistency, such as eventual consistency, for performance and high availability, where a user can read stale data for a period of time. The domain name system (DNS) is one of the most popular applications that implement eventual consistency. Updates to name will not be visible immediately, but all clients are ensured to see them eventually.

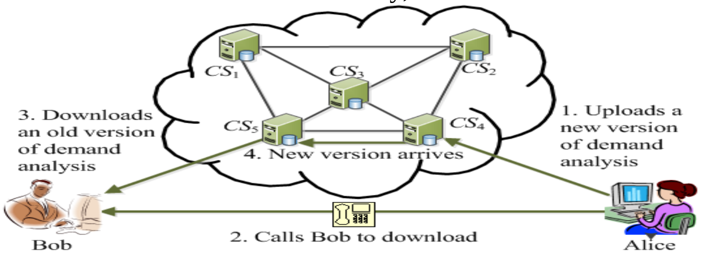

CS_1
CS_3
CS_2
CS_5
CS_4
3. Downloads an old version of demand analysis
4. New version arrives
1. Uploads a new version of demand analysis
2. Calls Bob to download
Bob
Alice

Fig. 1.1: An application requires casual consistency

However, eventual consistency is not a catholicon for all applications. Especially for the interactive applications, stronger consistency assurance is of increasing importance. Consider the following scenario as shown in Fig. 1. 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, *CS*1, . . ., *CS*5. After uploading a new version of there requirement analysis to a *CS*4, Alice calls Bob to download the latest version for integrated design. Here, after Alice calls Bob, the causal relationship [5] 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. In this case, the integrated design that is based on an old version may not satisfy the real requirements of customers. Actually, different applications have different consistency requirements.

II. RELATED WORK

In "A VIEW OF CLOUD COMPUTING**"** paper the author proposed the Private-key storage e outsourcing allows clients with either limited resources or limited expertise to store and distribute large amounts of symmetrically encrypted data at low cost. Since regular private-key encryption prevents one from searching over encrypted data, clients also lose the ability to selectively retrieve segments of their data. To address this, several techniques have been proposed for provisioning symmetric encryption with search capabilities the resulting construct is typically called searchable encryption. The area of searchable encryption has been identified by DARPA as one of the technical advances that can be used to balance the need for both privacy and national security in information aggregation systems. One approach to provisioning symmetric encryption with search capabilities is with a so-called secure index. An index is a data structure that stores document collections while supporting efficient keyword search, i.e., given a keyword, the index returns a pointer to the documents that contain it. Informally, an index is \secure" if the search operation for a keyword w can only be performed by users that possess a \trapdoor" for w and if the trapdoor can only be generated with a secret key. Without knowledge of trapdoors, the index leaks no information about its contents. one can build a symmetric searchable encryption scheme from a secure index as follows: the client indexes and encrypts its document collection and sends the secure index together with the encrypted data to the server. To search for a keyword w, the client generates and sends a trapdoor for w which the server uses to run the search operation and recover pointers to the appropriate (encrypted) documents.

Symmetric searchable encryption can be achieved in its full generality and with optimal security using the work of oblivious RAMs. More precisely, using these techniques any type of search query can be achieved (e.g., conjunctions or disjunctions of keywords) without leaking any information to the server, not even the \access pattern" (i.e., which documents contain the keyword). This strong privacy guarantee, however, comes at the cost of a logarithmic (in the number of documents) number of rounds of interaction for each read and write. In the same paper, the authors show a 2-round solution, but with considerably larger square-root overhead. Therefore, the previously mentioned work on searchable encryption tries to achieve more efficient solutions (typically in one or two rounds) by weakening the privacy guarantees.

The System introduces new adversarial models for SSE. The first, which we refer to as non-adaptive, only considers adversaries that make their search queries without taking into account the trapdoors and search outcomes of previous searches. The second |adaptive| considers adversaries that choose their queries as a function of previously obtained trapdoors and search outcomes. All previous work on SSE (with the exception of oblivious RAMs) falls within the non-adaptive setting. The implication is that, contrary to the natural use of searchable encryption described in these definitions only guarantee security for users that perform all their searches at once. System addresses this by introducing game-based and simulation-based definitions in the adaptive setting.In analyzing consistency properties forfun and profit paper the author proposed the system consider the following distributed file system: a user U pays a server S for storage service, with the goal being that U can retrieve the stored ¯les anytime and anywhere through Internet connections. For example, U may store files containing personal data that U may want to later access using his wireless PDA. A user might be willing to pay for such a service in order to have access to data without carrying devices with large amount of memory, and to have the data well-maintained by professionals. Such distributed file services already exist, such as the \Yahoo! Briefcase".1 The system expect such services will grow with the expansion of mobile and pervasive computing. In many cases U will not want to reveal the contents of his files to S in order to maintain security or privacy. It follows that the files will often be stored in encrypted form. Suppose, however, that later U wants to retrieve files based on a keyword search. That is, U wants to retrieve ¯les containing (or indexed by) some keyword. If the files are encrypted, there is no straightforward way for S to do keyword search unless U is willing to leak the decryption key. A trivial solution that preserves the security of U's files is to have S send all the encrypted files back to him. This may not be a feasible solution if U is using mobile devices with limited bandwidth and storage space. An additional complication is that U may naturally also want to keep secret the keyword that he is interested in as well.

Data-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, the customer is satisfied.

Client-centric consistency : Client-centric consistency model concentrates on what specific customers want, i.e., how the customers observe data updates. Their work also describes different levels of consistency in distributed systems, from strict consistency to weak consistency. High consistency implies high cost and reduced availability. the states that strict consistency is never needed in practice, and is even considered harmful

III. EXISTING METHODOLOGY

There exist various tools and technologies for multiple cloud, such as Platform VM Orchestrator, Vmware vSphere, and Ovirt. These tools help cloud providers construct a distributed cloud storage platform (DCSP) for managing clients' data. However, if such an important platform is vulnerable to security attacks, it would bring irretrievable losses to the clients. For example, the confidential data in an enterprise may be illegally accessed through a remote interface provided by a multiple-cloud, or relevant data and archives may be lost or tampered with when they are stored into an uncertain storage pool outside the enterprise. Therefore, it is indispensable for cloud service providers (CSPs) to provide security techniques for managing their storage services.

DISADVANTAGES

- Untrusted clouds.
- Full data in all cloud.
- Low level security.
- Need more data space.
- Cloud provider can access client's data.

IV. PROPOSED SYSTEM

We present a Consistency as a service auditing cloud scheme. We prove the security of our scheme based data fragmentation on multiple clouds. So the proposed system has data fragmentation, data security and storage on multiple cloud services. We used trusted third party to store the data on multiple cloud and find the data access by untrusted cloud service providers. In this system the client data divide into multiple pieces and common secret key, send to the multiple clouds with help of trusted third party. If any of the untrusted cloud service providers try modify the data the alert will send to trusted third party about illegal access of untrusted cloud service provider.

TTP (TRUSTED THIRD PARTY)

Log In : Here TTP has to log in by using their unique user name and password. TTP is the only authorized person to access TTP module for security purpose. So others don't get rights to access this module.

Transfer : In this module TTP view the client uploaded file and transfer them into multiple-cloud. The file will split into 3 pieces and stored in cloud. TTP is the only authorized person to access TTP module for security purpose. So others don't get rights to access this module.

View : In this module TTP view the client uploaded file from multiple-cloud. TTP is the only authorized person to access TTP module for security purpose. So others don't get rights to access this module.

Alerts : In this module TTP view the alerts of the security issues of client uploaded files in cloud. That is if any of CSP try to access client file the alert will send to TTP. TTP is the only authorized person to access ttp module for security purpose. So others don't get rights to access this module.

CSP (CLOUD SERVICE PROVIDER)

Log In : Here CSP has to log in by using their unique user name and password. CSP is the only authorized person to access CSP module for security purpose. So others don't get rights to access this module.

View : In this module CSP view the client uploaded file in their cloud as encrypted format. If CSP try edit the client file the alert will send to TTP. CSP is the only authorized person to access TTP module for security purpose. So others don't get rights to access this module.

CLIENT
Register : Here client has to register their details to log In. It contains unique user name and password. Client is the only authorized person to access this module for security purpose. So others don't get rights to access this module.

Log In : Here client has to log in by using their unique user name and password after registration. Client is the only authorized person to access this module for security purpose. So others don't get rights to access this module.

Upload : In this module client upload their files means give to the correct secret key. The secret key is wrong means the file is not uploaded. And also enter the expire date. That date only visible the file in cloud. Client is the only authorized person to access this module for security purpose. So others don't get rights to access this module.

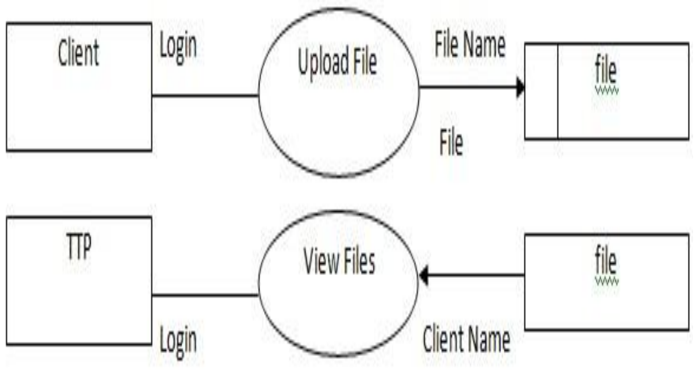
View : In this module client view their uploaded file from multiple-cloud. Client can edit the particular split. The updates will changed on cloud. Client is the only authorized person to access this module for security purpose. So others don't get rights to access this module.

Download : In this module client download their files means give to the correct secret key. The secret key is wrong means the file is not download. Particular date only client download the upload files.. Client is the only authorized person to access this module for security purpose. So others don't get rights to access this module.

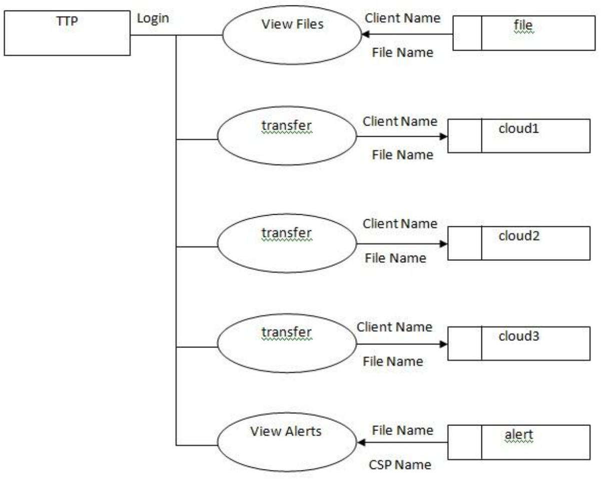
ADVANTAGES
- Trusted third party.
- Incomplete data in all cloud.
- Security alerts, encryption data.
- Need low data space.
- Cloud provider can't accessing client's data.

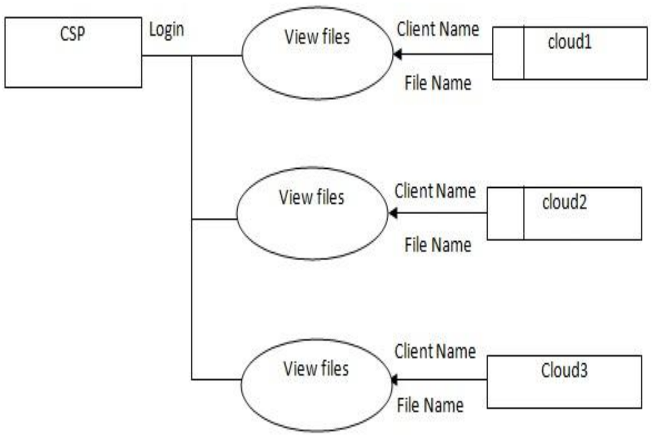
V. DATAFLOW DIAGRAM

LEVEL 0:



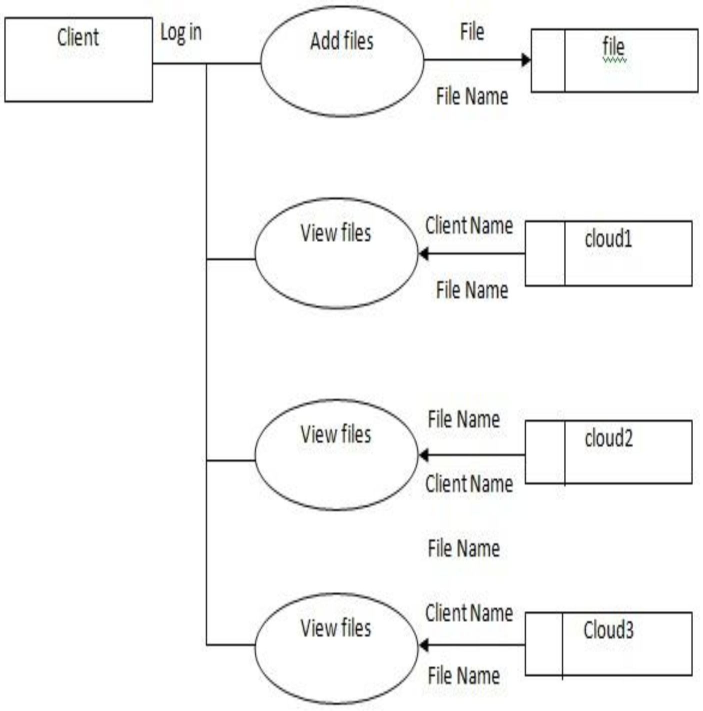
LEVEL 1:





LEVEL2:

LEVEL 3:



VI. ALGORITHM USED

LOCAL CONSISTENCY AUDITING

Local consistency auditing is an online algorithm (Alg. 1). In Alg. 1, 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.

- 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** $W(a) \rightarrow W(b)$ **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

GLOBAL CONSISTENCY AUDITING : Local consistency auditing is an online algorithm (Alg. 1). In Alg. 1, 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.

- Each operation in the global trace is denoted by a vertex
- **for** any two operations *op*1 and *op*2 **do**
- **if** $op1 \rightarrow op2$ **then**
- A time edge is added from *op*1 to *op*2
- **if** $op1 = W(a)$, $op2 = R(a)$, and two operations come
- from different users **then**
- A data edge is added from *op*1 to *op*2
- **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 *op*1 to *op*2
- Check whether the graph is a DAG by topological sorting

VII. CONCLUSION

The system proposes a consistency as a service (CaaS) model and a two-level auditing structure to help users verify whether the cloud service provider (CSP) is providing the 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, e.g., the least expensive one that still provides adequate consistency for the users' applications .For our future work, we will conduct a thorough theoretical study of consistency models in cloud computing.

REFERENCES

[1] Qin Liu, Guojun Wang, , and Jie Wu "Consistency as a Service: Auditing Cloud Consistency "Ieee Transactions On Network And Service Management Vol:11 No:1 Year 2014

[2] 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.

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

[4] E. Brewer, "Towards robust distributed systems," in Proc. 2000 ACMPODC.

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

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

[7] W. Lloyd, M. Freedman, M. Kaminsky, and D. Andersen, "Don't settlefor eventual: scalable causal consistency for wide-area storage witCOPS," in Proc. 201 1 ACM SOSP.

[8] E. Anderson, X. Li, M. Shah, J. Tucek, and J. Wylie, "What consistencydoes your key-value store actually provide," in Proc. 2010 USENIXHotDep.

[9] C. Fidge, "Timestamps in message-passing systems that preserve thepartial ordering," in Proc. 1988 ACSC.

[10] W. Golab, X. Li, and M. Shah, "Analyzing consistency properties forfun and profit," in Proc. 2011 ACM PODC.