International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

Recovering Data Stability Service for Preserving Rational Data in Cloud Environment

Dr. R. Reka[1], Dr. T. Parithimarkalaignan[2]

[1]Professor & Head, Department of Computer Science & Engineering, Annai Mathammal Sheela Engineering College, Namakkal, Tamil Nadu, India

[2]Principal, Annai Mathammal Sheela Engineering College, Namakkal, Tamil Nadu, India

ABSTRACT

Cloud computing usage has increased rapidly in many companies. Cloud computing offers many benefits in terms of low cost and accessibility of data. Cloud computing has recently emerged as a key technology to provide individuals and companies with access to remote computing and storage infrastructures. In order to achieve highly available yet high performing services, cloud data stores rely on data replication. However, the replication technique brings the issue of stability. The data is replicated in multiple geographically distributed data centers, and to meet the increasing requirements of distributed applications, many cloud data stores adapt eventual stability and allows running the data intensive operations under low latency and results in the cost of data staleness. Reliability is often enhanced in cloud computing environments because Service Providers utilize multiple redundant sites for disaster recovery. This is attractive to enterprises for business continuity. Due to these issues we proposed a novel called Data Stability as a Service (DSaaS) model for efficient cloud process and to provide promised level of stability by using crypto analysis algorithm for security using hidden approach mechanism. We proposed Third Party Auditing technique and also role based access control which only requires a loosely synchronized clock in the audit cloud.

Keywords : Cloud Storage, Data Stability as a Service, Data Management, TPA, RBAC.

I. INTRODUCTION

Cloud computing has recently emerged as a technology to allow users to access infrastructure, storage, software and deployment environment based on a pay-for-what-they-use model. Cloud computing is a complete new technology. It is the development of parallel computing, distributed computing grid computing, and is the combination and evolution of Virtualization, Utility computing, Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS). Cloud is a metaphor to describe web as a space where computing has been pre-installed and exist as a service; data, operating systems, applications, storage and processing power exist on the web ready to be shared. To users, cloud computing is a Pay-per-Use-On-Demand mode that can conveniently access shared IT resources through the Internet. Where the IT resources include network, server, storage, application, service and so on and they can be deployed with much quick and easy manner and least management and also interactions with service providers. Cloud computing can much improve the availability of IT resources and owns many advantages over other computing techniques. Users can use the IT infrastructure with Pay-per-Use-On-Demand mode; this would benefit and save the cost to buy the physical resources that may be vacant [1].

II. METHODS AND MATERIAL

1. Service Models

A. **Infrastructure as a Service (IaaS)** means you're buying access to raw computing hardware over the Net, such as servers or storage. Since you buy what you need and pay-as-you-go, this is often referred to as utility computing. Ordinary web hosting is a simple example of IaaS: you pay a monthly subscription or a per-megabyte/gigabyte fee to

have a hosting company serves up files for your website from their servers.

B. **Software as a Service (SaaS)** means you use a complete application running on someone else's system. Web-based email and Google Documents are perhaps the best-known examples. Zoho is another well-known SaaS provider offering a variety of office applications online.

C. **Platform as a Service (PaaS)** means you develop applications using Web-based tools so they run on systems software and hardware provided by another company. So, for example, you might develop your own ecommerce website but have the whole thing, including the shopping cart, checkout, and payment mechanism running on a merchant's server. App Cloud (from salesforce.com) and the Google App Engine are examples of PaaS.

2. Related Work

A cloud is essentially a large scale distributed system [3] where each piece of data is replicated on multiple geographically distributed servers to achieve high availability and high performance. Thus, we first review the stability models in the distributed systems. In a standard textbook, anticipated two classes of stability models: data-centric stability and client-centric stability. Data-centric stability 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. Therefore, client-centric stability model concentrates on what specific customers want i.e., how the customers observe the data updates and their work describes different levels of stability in distributed systems, from strict stability to weak stability. High stability implies high cost and reduced availability and in the states strict stability never needed practice concern and is even considered harmful. In reality, mandated by the CAP protocol [4] many distributed systems sacrifice strict stability for high availability. Then, we have a brief work on achieving different levels of stability in a cloud. By means of encrypting and assigning secured identities for each request and response at each stage, together with the maintenance of machine readable usage/access rights, privacy is preserved. While it is easier to carry out the encryption schemes, there exists a difficulty in providing machine readable access rights. This problem of effective right expressions generation is the future work that has to be carried out. Considering the privacy of the users in the cloud environment and projected a flexible method of access control. Each cloud user is linked with certain attribute, which determines their access rights. The paper propounded a two-tier encryption model in which the base phase and surface phase builds up the two-tier of the model respectively. At the first phase, the data owner performs local attribute-based encryption on the data that has to be outsourced. The surface phase on the other hand is performed by the cloud servers, after the initialization done by the cloud data owner this phase implements T. The causal memory model [5] has attracted the attention of a number of researchers because it is considered to be powerful enough to allow easy programming (strong memory models), but at the same time it also allows inexpensive implementations (weak memory models). As a consequence, a number of algorithms implementing the causal memory increases the concurrency and supports replication of data. With the replication, there are copies (replicas) of the same variables in the local memories of several processes of the system, which allows these processes to use the variables simultaneously. However, in order to guarantee the stability of the shared memory, the system must control the replicas when the variables are updated and that control can be done by either invalidating outdated replicas or by propagating the new variable values to update the replicas.

3. Stability Model

Nowadays cloud computing is the fastest growing technology that is used as a source of providing service through Internet. It is an enhanced model of Utility Computing. It embodies all technologies in Computer Architecture. It delivers the clients the needed applications, processes and Information as a service. It provides software platform as a service and virtualized servers, storage area and networks as a service. In addition to that it also manages and delivers database services. The traditional time consuming way of database management degrades the system performance. At present a weak form of stability is maintained in cloud computing environment. This paper introduced recovered stability model for cloud

computing platform using prioritized read-write mechanism [4].

Analysis of assuring the stability models for semi-active replication protocol is presented in this part of the paper. Both data-centric and client-centric stability gives assured with the two scopes: ordering and staleness. Staleness describes how much a given replica is lagging behind, either expressed in terms of time or versions. Low bounded staleness values can often be tolerated by applications as long as the corresponding real-world events that would have the same or higher staleness values without a database system. In general, apart from the context of semi-active protocol, when replica send request to replica, the system storage of replica will be updated right away. In contrast replica might not be consistent with replica might not be consistent with replica for some time. The small staleness values often will appear but we may not sense it and ordering is more critical than staleness. In a setting with strict stability, all requests must be executed on all replicas in same chronological order. This stability model is hard to implement in distributed databases due to clock synchronization issues and communication delays which cause that replicated servers might disagree on the chronological order of events. The standard database mechanism of locking offers poor performance levels in a distributed setting. Based on this, data centric stability models exist that relax certain ordering requirements while keeping those that are essential to applications.

4. Data Stability Models In Public Cloud Storage

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 stability models. Different cloud service providers employ different data stability models nowadays. In this post, we survey the data stability models provided by the solutions from the three big players: Amazon S3 and DynamoDB, Google Cloud Storage and Windows Azure Storage.

A. Amazon S3

Amazon S3 is a simple key-based object store service for the Internet. Amazon S3 buckets in all Regions provide **read-after-write stability for PUTS of new objects** and **eventual stability for overwrite PUTS and DELETES**. However, there is one exception: if a HEAD or GET request to a key name is made to find if the object exists before the object is create, Amazon S3 provides only eventual stability. Updates to a single key are atomic.

B. Google Cloud Storage

Google Cloud Storage provides **strong global stability** for upload and delete operations and list operations in a region, and **eventual stability** for object list operations across regions. For access controlling, granting is **strongly consistent** while revoking is **eventual consistent**. Additionally, the upload operations to Google Cloud Storage are atomic. Caches, as usually, have a different stability model from the storage itself. Cached objects from Google Cloud Storage that are publicly readable might not exhibit strong stability.

C. Windows Azure Storage

Windows Azure Storage provides three properties that the CAP theorem claims are difficult to achieve at the same time: **strong consistency**, high availability, and partition tolerance. Brad Calder et al. published the design of the Windows Azure Storage in the paper Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency at SOSP'11.

5. Groundwork

Cloud storage services [2] have become commercially popular due to their overwhelming advantages. To provide ubiquitous always-on access, a cloud service provider (CSP) maintains multiple replicas for each piece of data on geographically distributed servers. A key problem of using the replication technique in clouds is that it is very expensive to achieve strong stability on a world wide scale. In this section we first illustrate the stability as a service (SaaS) model, and then we describe the auditing strategies. Finally, we discuss about the proposed model Data Stability as a Service (DSaaS) with the auditing structure to provide the security levels to guarantee the stability levels.

A. STABILITY AS A SERVICE (SAAS)

In this paper, we first present a novel Stability as a Service (SaaS) model [13], which consists of a large data cloud and multiple small audit clouds. In this model, data cloud is maintained by a CSP, and a group

of users that constitute an audit cloud can verify whether the data cloud provides the promised level of consistency or not. In this model 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 commonality of violations, and the staleness of the value of a read. Finally, we devise a Heuristic Auditing Strategy (HAS) to reveal as many violations as possible. Extensive experiments were performed using a combination of simulations and real cloud deployments to validate HAVE. Furthermore, in this model some issues have been occurred they are, discrimination may cause a much more information loss from updating, data downloading time consumption is more for requesting, sensitive attributes does not prevent unethical and less security for data transformation addressed issues in this model.

B. DATA STABILITY AS A SERVICE (DSAAS)

Motivated by the increasing popularity of eventually consistent key-value [9] stores as a commercial service, we address two important problems related to the stability properties in a history of operations on a read/write register i.e., the start time, finish time, argument, and response of every operation). To consider how to detect stability violation as soon as one happens. To this end, we formulate a specification for offline verification algorithms, and to overcome this problem Data Stability as a Service (DSaaS) model is proposed as a new platform service to encapsulate the proposed approach. DSaaS service ensures SaaS services with crypto analysis services for cloud portability and security, as it works as a cloud adapter between Role Based Access Control (RBAC) service instances. Experiments show that proposed approach realized by the DSaaS service with RBAC access provides much better response time when compared with classical locking and blocking techniques. This model is considered as a rising subject, cloud stability is playing an increasingly important role in the decision support activity of every walk of life and to get efficient item set result based on the DSaaS.

6. Stability Types

In data –centric stability models ordered by the strictness of their guarantees in semi-active data replication protocol. For each model, how it can be translated into a client-centric stability model is discussed in this section. As already discussed, there are two stability scopes: staleness and ordering. The following stability models (apart from Linearizability) do not consider staleness. In fact, increasing strictness of ordering guarantees often leads to higher staleness values as updates may not be applied directly but are required to fulfill dependencies at first. The lowest possible ordering guarantee is typically described as weak stability. As the name states, guarantees are very weak in that they do not really exist. Essentially, weak stability translates to a colloquial “replicas might by chance become consistent”. While an implementation may or may not have a protocol to synchronize replicas, a typical use-case can be found in the context of a browser cache: it is updated from time to time but replicas will rarely be consistent. As weak stability does not provide any ordering guarantees at all, there is no relation to client-centric stability models in semi-active data replication protocol. The Eventual stability [6] is a little strict stability. It requires convergence of replicas, i.e., in the absence of updates and failures the system converges towards a consistent state. Updates may be reordered in any way possible and a consistent state is simply defined as all replicas being identical. Eventual stability is very vague in terms of concrete guarantees but is very popular for web-based services. In terms of client-centric stability guarantees, eventual stability often fulfills these guarantees for a majority of requests but does not guarantee to do so. As an example, Amazon S34 currently delivers MRC for about 95% of all requests whereas it still violated MRC in about 12% of all requests. While there are certainly some use-cases where eventual stability cannot be applied, it often suffices as the real world itself is inherently eventually consistent. The difference is that more conflict resolution is necessary at the application layer requiring a higher skill set from application developers. Instead of pessimistically locking data items “guesses and apologies” are used. Data stores that are eventually consistent thus have the property that in the absence of updates, all replicas converge toward identical copies of each other. Eventual stability essentially requires only that updates are guaranteed to propagate to all replicas. Eventual stability is therefore often cheap to implement. The causal stability [7] is the strict level of stability that can be achieved in an always available storage system based on the trade-offs of the CAP theorem [4]. In a causal consistent storage system, all requests that have a causal relationship to another

request must be serialized i.e., executed in the same order on all replicas while unrelated requests may be serialized in arbitrary order.

III. RESULTS AND DISCUSSION

1. DSAAS ARCHITECTURE

A. DSAAS ARCHITECTURAL DESIGN

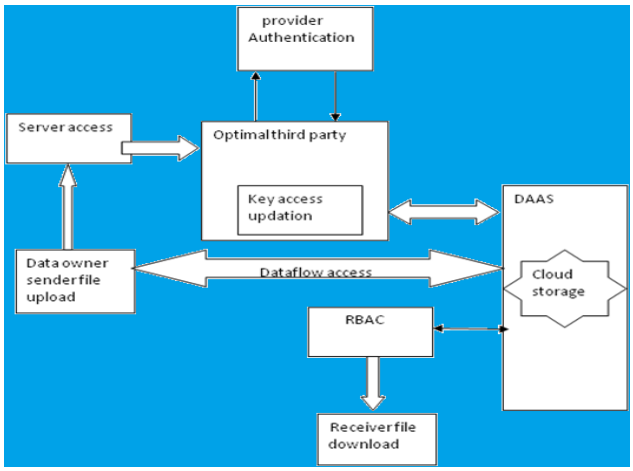


Figure 1. DSAAS Architecture

B. RBAC ARCHITECTURE

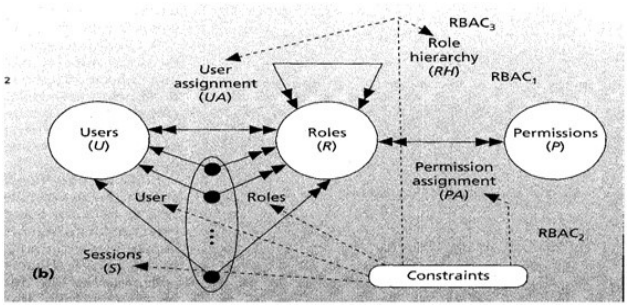


Figure 1. RBAC Access Roles

One kind of access control which makes only the authorized persons are allowed s clearly determined by Role Based Access Control (RBAC). RBAC consists of different roles, constraints according to the sessions the roles are taken place in the accessing resources from the authorized accessibilities. RBAC distinguishes different roles and responsibilities in order to make the access of system or to maintain the database resources from client to the server and server to the client with regard to the roles such as individual user or user of the organization or group of users which is controlled by the group member. According to the role the access is made with the authorization abilities without unauthorization accessibilities.

C. DSAAS Architecture\

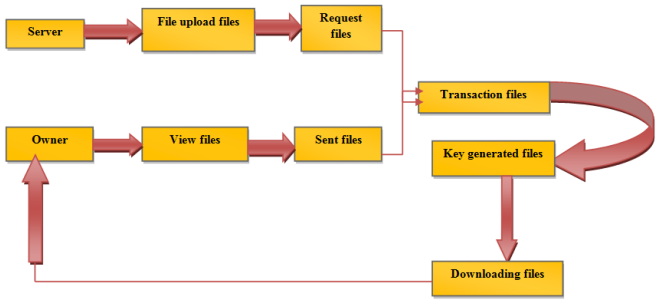
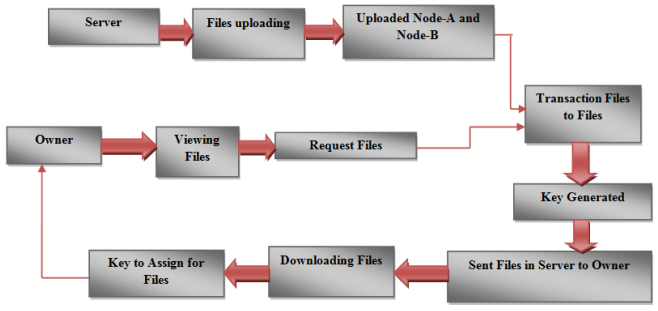


Figure 3. DSaaS Architecture

D. DATA FLOW DIAGRAM



2. MODULES OF DSAAS

A. EPIGRAPHY KEY HYPOTHESIS

Cloud computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. In this article, we focus on cloud data storage security, which have not been an important aspect of quality of service. To ensure the correctness of users' data in the cloud, we propose an effective and flexible distributed scheme with two salient features, opposing to its predecessors. By utilizing the homomorphism token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s). Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including data updates, delete and append operations. Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine

failure, malicious data modification attack, and even server colluding attacks.

B. ELEMENT KEY HYPOTHESIS

Group key distribution schemes has recently received a lot of attention from the researchers, as a method enabling large and dynamic groups of users to establish group keys over unreliable network for secure multicast communication. In such schemes, time is divided into epochs called sessions. At the beginning of each session, a Group Manager transmits some broadcast message, in order to provide a common key to each member of the group. Every user, belonging to the group, computes the group key using the message and some private information. The main property of the scheme is that, if some broadcast message gets lost, then users are still capable of recovering the group key for that session by using the message they received at the beginning of a previous session and the message they will receive at the beginning of a subsequent one, without requesting additional transmission from the Group Manager. This approach decreases the workload on the Group Manager and reduces network traffic as well as the risk of user exposure through traffic analysis.

C. KEY ALLOTMENT

Common group key if frequently updated to ensure secure multicast communication Group lifetime is divided into sessions; single key instance is valid only throughout one session. Group membership can change between consecutive sessions. At the beginning of session j, Group Membership distributes a new session key to nodes. Session duration is determined by the Group Membership. It can vary over time, depending on security policy, group membership changes and nodes behavior. Session key changes have to be performed, with some predefined minimum frequency to protect the system from cryptanalysis attacks. Moreover, to effectively remove the node from multicast group, who is willing to leave, or is forced to leave, a new session must begin and nodes shall start from protecting group communication using a new session, which is not accessible to the attacks. Thus, the choice of session length is a tradeoff between key distribution cost in terms of communication and computational overhead, and the required security level.

IV. CONCLUSION

We concluded that a new service (i.e., Data Stability as a Service) to encapsulate the proposed approach. DSaaS service also ensures SaaS with crypto analysis services for cloud portability and security, as it works as a cloud adapter between RBAC service instances. Experiments show that anticipated approach realized by the DSaaS service with RBAC access provides much better response time when compared with classical locking and blocking techniques.

V. 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]. E. Brewer, "Towards robust distributed systems," in Proc. 2000 ACM PODC.

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

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

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

[7]. 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.

[8]. 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.

[9]. 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.

[10]. C. Fidge, "Timestamps in message-passing systems that preserve the partial ordering," in Proc. 1988 ACSC.

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

[12]. A. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms. Prentice Hall PTR, 2002.
[13]. Qin Liu, Guojun Wang, Member, IEEE, and Jie Wu, Fellow, IEEE, "Consistency as a Service: Auditing Cloud Consistency", vol.11, no.1, March 2014.
[14]. http://www.explainthatstuff.com
[15]. https://www.systutorials.com