ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Secure Cloud Transactions by Performance, Accuracy, and Precision

Patil Vaibhav Nivrutti
M.Tech Student,
Dept of Computer Science & Engineering,
St. Mary's Engineering College, Deshmukhi.

K Suresh Kumar
Assistant Professor,
Dept of Computer Science & Engineering,
St. Mary's Engineering College, Deshmukhi.

ABSTRACT:

In distributed transactional database systems deployed over cloud servers, entities cooperate to form proofs of authorizations that are justified by collections of certified credentials. These proofs and credentials may be evaluated and collected over extended time periods under the risk of having the underlying authorization policies or the user credentials being in inconsistent states. It therefore becomes possible for policy-based authorization systems to make unsafe decisions that might threaten sensitive resources. In this paper, we highlight the criticality of the problem. We then define the notion of trusted transactions when dealing with proofs of authorization. Accordingly, we propose several increasingly stringent levels of policy consistency constraints, and present different enforcement approaches to guarantee the trustworthiness of transactions executing on cloud servers. We propose a Two-Phase Validation Commit protocol as a solution, which is a modified version of the basic Two-Phase Validation Commit protocols. We finally analyze the different approaches presented using both analytical evaluation of the overheads and simulations to guide the decision makers to which approach to use.

Index Terms:
Cloud databases, authorization policies, consistency, distributed transactions, atomic commit protocol.

INTRODUCTION:

From an economic perspective, cloud consumers can save huge IT capital investments and be charged on the basis of a pay-only-for-what-you-use pricing model. One of the most appealing aspects of cloud computing is its elasticity, which provides an illusion of infinite, on-demand resources making it an attractive environment for highly scalable, multi tiered applications.

However, this can create additional challenges for back-end, transactional database systems, which were designed without elasticity in mind. Despite the efforts o f key-value stores like Amazon's SimpleDB, Dynamo, and Google's Bigtable to provide scalable access to huge amounts of data, transactional guarantees remain a bottleneck. To provide scalability and elasticity, cloud services often make heavy use of replication to ensure consistent performance and availability. As a result, many cloud services rely on the notion of eventual consistency when propagating data throughout the system. This consistency model is a variant of weak consistency that allows data to be inconsistent among some replicas during the update process, but ensures that updates will eventually be propagated to all replicas.

This makes it difficult to strictly maintain the ACID guarantees, as the "C" (consistency) part of ACID is sacrificed to provide reasonable availability. In systems that host sensitive resources, accesses are protected via authorization policies that describe the conditions under which users should be permitted access to resources. These policies describe relationships between the system principles, as well as the certified credentials that users must provide to attest to their attributes. In a transactional database system that is deployed in a highly distributed and elastic system such as the cloud, policies would typically be replicated very much like data among multiple sites, often following the same weak or eventual consistency model.

It therefore becomes possible for a policy-based authorization system to make unsafe decisions using stale policies. Interesting consistency problems can arise as transactional database systems are deployed in cloud environments and use policy-based authorization systems to protect sensitive resources. In addition to handling consistency issues among database replicas, we must also handle two types of security inconsistency conditions.

First, the system may suffer from policy inconsistencies during policy updates due to the relaxed consistency model underlying most cloud services. In this paper, we address this confluence of data, policy, and credential inconsistency problems that can emerge as transactional database systems are deployed to the cloud.

EXISTING SYSTEM:

•To protect user access patterns from a cloud data store, Williams et al. introduce a mechanism by which cloud storage users can issue encrypted reads, writes, and inserts. Further, Williams et al. propose a mechanism that enables un trusted service providers to support transaction serialization, backup, and recovery with full data confidentiality and correctness.

•A dynamic consistency rationing mechanism that automatically adapts the level of consistency at runtime. Both of these works focus on data consistency, while our work focuses on attaining both data and policy consistency.

•Proofs of data possession have been proposed as a means for clients to ensure that service providers actually maintain copies of the data that they are contracted to host. In other works, data replications have been combined with proofs of retrieve ability to provide users with integrity and consistency guarantees when using cloud storage.

•CloudTPS is primarily concerned with providing consistency and isolation upon data without regard to considerations of authorization policies.

•This work proactively ensures that data stored at a particular site conforms to the policy stored at that site. If the policy is updated, the server will scan the data items and throw out any that would be denied based on the revised policy.

•The consistency of distributed proofs of authorization has previously been studied, though not in a dynamic cloud environment. This work highlights the inconsistency issues that can arise in the case where authorization policies are static, but the credentials used to satisfy these policies may be revoked or altered.

•The authors develop protocols that enable various consistency guarantees to be enforced during the proof construction process to minimize these types of security issues.

•Disadvantages: This Existing Works only focus on data consistency. It does not focus on policy consistency. This work only concerns itself with local consistency of a single node, not with transactions that span multiple nodes. This work highlights the inconsistency issues that can arise in the case where authorization policies are static, but the credentials used to satisfy these policies may be revoked or altered.

PROPOSED SYSTEM:

•In this paper highlight the criticality of the problem. It defines the notion of trusted transactions when dealing with proofs of authorization. Accordingly, it propose several increasingly stringent levels of policy consistency constraints, and present different enforcement approaches to guarantee the trustworthiness of transactions executing on cloud servers.

•It proposed a Two-Phase Validation Commit protocol as a solution, which is a modified version of the basic Two-Phase Validation Commit protocols.

•It finally analyze the different approaches presented using both analytical evaluation of the overheads and simulations to guide the decision makers to which approach to use.

•In this paper address this confluence of data, policy, and credential inconsistency problems that can emerge as transactional database systems are deployed to the cloud.

•This paper formalized the concept of trusted transactions. Trusted transactions are those transactions that do not violate credential or policy inconsistencies over the lifetime of the transaction.

•It present a more general term, safe transactions, that identifies transactions that are both trusted and conforms to the ACID properties of distributed database systems.

•It defines several different levels of policy consistency constraints and corresponding enforcement approaches that guarantee the trustworthiness of transactions executing on cloud servers.

• It proposed a Two-Phase Validation Commit (2PVC) protocol that ensures that a transaction is safe by checking policy, credential, and data consistency during transaction execution.

• Advantages:

It provides a good balance between accuracy and performance, at the cost of higher code complexity.

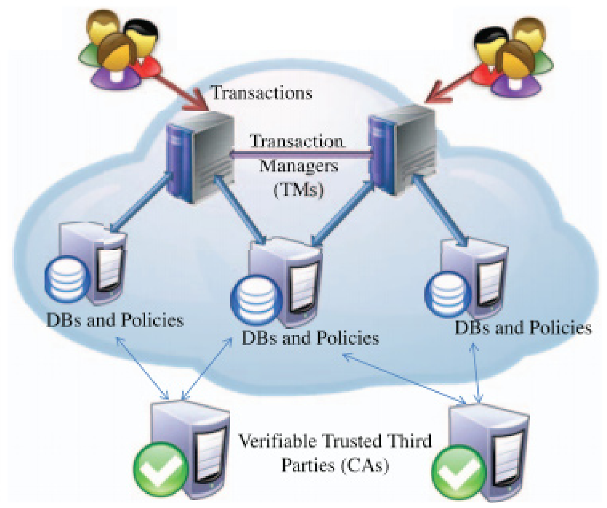
HARDWARE REQUIREMENTS:

• Processor Type	: Pentium IV
• Speed	: 2.4 GHZ
• RAM	: 256 MB
• Hard disk	: 20 GB
• Keyboard	: 101/102 Standard Keys
• Mouse	: Scroll Mouse

SOFTWARE REQUIREMENTS:

• Operating System	: Windows 7
• Programming Package	: Net Beans IDE 7.3.1
• Coding Language	: JDK 1.7
• Database	: SQL Server 2005

SYSTEM ARCHITECTURE:


Transactions
Transaction Managers (TMs)
DBs and Policies
DBs and Policies
DBs and Policies
Verifiable Trusted Third Parties (CAs)

MODULES:

1.Cloud Formation

2.Data Owner Register with Authorization Policies

3.Upload File

4.Safe Transaction

5.Download File

CLOUD FORMATION:

• First create a cloud infrastructure. It consisting of a set of S servers, where each server is responsible for hosting a subset of all data items belonging to a specific application domain.

• Users interact with the system by submitting queries or update requests encapsulated in ACID transactions. A transaction is submitted to a Transaction Manager (TM) that coordinates its execution.

• Multiple TMs could be invoked as the system workload increases for load balancing, but each transaction is handled by only one TM.

• It denote by the set of all credentials, which are issued by the Certificate Authorities (CAs) within the system. Here each CA offers an online method that allows any server to check the current status of credentials.

WITH AUTHORIZATION POLICIES:

• Next Data Owner Registered with authorization policies, valid date from and valid date to in desirable Trusted Third Party or CA.

• This Trusted Third Party or CA allows any server to check the current status of credentials.

• Then the CA creates secret keys for each data owner and end user. Because this Secret Keys are used to Authentication Purpose.

• A Data Owner wants to upload his file and end user wants to download a file, both are used this secret key for encryption and decryption.

UPLOAD FILE:

• Data Owner wants to upload a file. So he encrypted this file using TA's secret Key.

• First he sends a key request to Trusted Third Party.

• Trusted Third Party creates a secret key and provide to Data Owner.

• Then the data owner encrypts his file using this secret key.

SAFE TRANSACTION:

• A safe transaction is a transaction that is both trusted (i.e., satisfies the correctness properties of proofs of authorization) and database correct (i.e., satisfies the data integrity constraints).

• It first describes an algorithm that enforces trusted transactions (2PV), and then expands this algorithm to enforce safe transactions (2PVC).

• 2PV algorithm operates in two phases: collection and validation. During collection, the TM first sends a Prepare-to-Validate message to each participant server.

• In response to this message, each participant 1) evaluates the proofs for each query of the transaction using the latest policies it has available and 2) sends a reply back to the TM containing the truth value (TRUE/FALSE) of those proofs along with the version number and policy identifier for each policy used.

• Further, each participant keeps track of its reply (i.e., the state of each query) which includes the id of the TM, the id of the transaction to which the query belongs, and a set of policy versions used in the query's authorization.

• Once the TM receives the replies from all the participants, it moves on to the validation phase. If all polices are consistent, then the protocol honors the truth value where any FALSE causes an ABORT decision and all TRUE cause a CONTINUE decision.

• In the case of inconsistent policies, the TM identifies the latest policy and sends an Update message to each out -of -date participant with a policy identifier and returns to the collection phase.

• In this case, the participants 1) update their policies, 2) reevaluate the proofs and, 3) send a new reply to the TM.

• 2PVC can be used to ensure the data and policy consistency requirements of safe transactions.

• Specifically, 2PVC will evaluate the policies and authorizations within the first, voting phase. That is, when the TM sends out a Prepare-to-Commit message for a transaction, the participant server has three values to report 1) the YES or NO reply for the satisfaction of integrity constraints as in 2PC, 2) the TRUE or FALSE reply for the satisfaction of the proofs of authorizations as in 2PV, and 3) the version number of the policies used to build the proofs as in 2PV.

• The process for the TM under view consistency. It is similar to that of 2PV with the exception of handling the YES or NO reply for integrity constraint validation and having a decision of COMMIT rather than CONTINUE. The TM enforces the same behavior as 2PV in identifying policies inconsistencies and sending the Update messages. The same changes to 2PV can be made here to provide global consistency by consulting the master policies server for the latest policy version.

DOWNLOAD FILE:

• An end User wants to access this upload file, he give the download request to particular DB's Server.

• This request contains filename, data owner and so on.

• The particular Server match this request to its database then retrieve the result and provide output to the user.

• Finally, the end users decrypt this file with data owner's secret key and access this file.

CONCLUSIONS:

Despite the popularity of cloud services and their wide adoption by enterprises and governments, cloud providersstill lack services that guarantee both data and access control policy consistency across multiple data centers. In this paper, we identified several consistency problems that can arise during cloud-hosted transaction processing using weak consistency models, particularly if policy-based authorization systems are used to enforce access controls. To this end, we developed a variety of lightweight proof enforcement and consistency models—i.e., Deferred, Punctual, Incremental, and Continuous proofs, with view or global consistency—that can enforce increasingly strong protections with minimal runtime overheads.

REFERENCES:

1.M.K. Iskander, D.W. Wilkinson, A.J. Lee, and P.K. Chrysanthis, "Enforcing Policy and Data Consistency of Cloud Transactions," Proc. IEEE Second Int'l Workshop Security and Privacy in Cloud Computing (ICDCS-SPC-CICDCS-SPCC), 2011.

2.S. Das, D. Agrawal, and A.E. Abbadi, "Elastras: An Elastic Transactional Data Store in the Cloud," Proc. Conf. Hot Topics in Cloud Computing (USENIX HotCloud '09), 2009.

3.D.J. Abadi, "Data Management in the Cloud: Limitations and Opportunities," IEEE Data Eng. Bull., vol. 32, no. 1, pp. 3-12, Mar. 2009.

4.J. Li, N. Li, and W.H. Winsborough, "Automated Trust Negotiation Using Cryptographic Credentials," Proc. 12th ACM Conf. Computer and Comm. Security (CCS '05), Nov. 2005.

5.J. Li and N. Li, "OACerts: Oblivious Attribute Based Certificates," IEEE Trans. Dependable and Secure Computing, vol. 3, no. 4, pp. 340-352, Oct.-Dec. 2006.

6.L. Bauer et al., "Distributed Proving in Access-Control Systems," Proc. IEEE Symp. Security and Privacy, May 2005.

7.T. Wobber, T.L. Rodeheffer, and D.B. Terry, "Policy-Based Access Control for Weakly Consistent Replication," Proc. ACM Fifth European Conf. Computer Systems (EuroSys '10), 2010.

8.A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket, "Venus: Verification for Untrusted Cloud Storage," Proc. ACM Workshop Cloud Computing Security (CCSW '10), 2010.

About Author's:

Patil Vaibhav Nivrutti
M.Tech Student,
Dept of Computer Science & Engineering,
St. Mary's Engineering College, Deshmukhi.

K Suresh Kumar
Assistant Professor,
Dept of Computer Science & Engineering,
St. Mary's Engineering College, Deshmukhi.