Enforcing Policy and Credential using Two Phase Validation Commit Protocol

Ms. S. Silpa Krishna,
M.E(IInd Year).
Department of Computer Science and Engineering,
Prathyusha Institute of Technology and Management.

Ms. S. Sowmithra,
B.Tech,M.Tech.
ASSISTANT PROFESSOR I
Department of Computer Science and Engineering,
Prathyusha Institute of Technology and Management.

Ms. K. P. Revathi,
B.E,M.E.
ASSISTANT PROFESSOR
Department of Computer Science and Engineering,
Prathyusha Institute of Technology and Management.

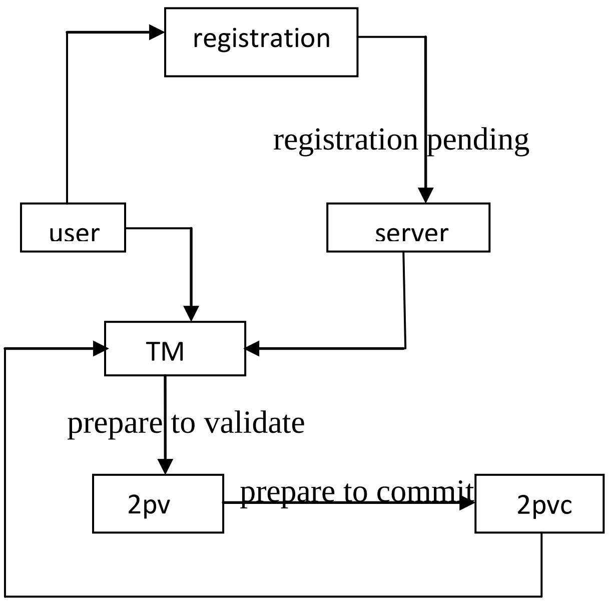
***Abstract*---Two phase commit(2PC) is used to coordinate the commitment of transaction in distributed systems. The standard 2PC optimization is the presumed abort variant, which users fewer messages when transactions are aborted and allows the coordinator to forget about aborted transaction .The proofs and credentials may be evaluated and collected over the extended time periods under the risk of having the underlying authorization policies or the user credentials being in inconsistent states. The several increasingly exact levels of policy consistency constraints, are presented for different enforcement approaches to guarantee the confidence of transaction executing on cloud server. The two phase validation commit protocol(2PVC) as a solution, that is a modified version of the basic two phase commit protocol. In this the time stamped voting was used both to optimize 2PC and to provide each committed transaction serialization.**

***Keywords*-atomic commitment protocol, two phase validate commit protocol, safe transaction.**

I INTRODUCTION:

Cloud computing has recently emerged as a computing paradigm in which storage and computation can be outsourced from organizations to next generation datacenters hosted by companies such as Amazon,Google,Yahoo, and Microsoft[1].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 requirement because even the slightest outage has significant financial consequences and impacts to the customer trust[2].Increasingly,data management is outsourced to third parties these trend is driven by growth and advances is cheap,high-speed communication infrastructures as well as by the fact that the total cost of data management is 5-10 times higher than the initial acquisition costs[3].

Moreover, today the remote servers are increasingly maintained by third party storage vendors. The third party storage vendors do not provide strong assurances of data confidentiality and integrity. For example personal emails and confidential files are being stored on third party servers such as Gmail and Xdrive. Privacy guarantees of such services are the best declarative and often subject customers to unreasonable fine-print clauses. To protect the data stored in such as untrusted server model security systems should offer users assurances of data confidentiality and access privacy. As a first line of defense to ensure confidentiality all data and associated meta-data can be encrypted at the client side using non-malleable encryption before being stored on the server. Encryption provides the important privacy guarantees at low cost.
The simplest way to store the structured data in the cloud is to deploy a relational database such as MySQL or Oracle.The relational data model,typically implemented via the SQL language and it provides the great flexibility in accessing data.It support sophisticated data access operations such as aggregation,range queries,join queries,etc.The RDBMS supports transactions and guarantees the strong data consistency.One can easily deploy a classical RDBMS in the cloud and thus get support for transactional consistency[5].



Interaction among the system components

A transaction manager is a set of queries to be executed automatically on a single consistent view of a database.The main challenge to support transactional guarantees in a cloud computing environment is to provide the ACID properties of Atomicity,Consistency,Isolation,Durability without compromising the scalability properties of the cloud[5].Obviously,any centralized transaction manager would face two scalability problems:1.A single transaction manager must execute all incoming transactions and eventually become the performance and availability bottleneck;2.A single transaction manager must maintain a copy of all data accessed by transactions and would eventually run out of storage space.To support the scalable transactions,to split the transaction manager into any number of Local Transaction Manager(LTMs) and to partition the application data and the load of transaction processing across LTMs.The hash function can be viewed as generating a universal name space for data blocks.Without co-operating or co-ordinating,multiple clients can share this name space and share a venti server[6].

II COORDINATING DISTRIBUTED COMMIT:

Distributed systems rely on the two phase commit(2PC) protocol to coordinate the commitment of transactions . 2PC guarantees the atomicity of distributed transactions, that is, that all cohorts of a transaction either commit or abort the transaction.The cost of 2PC is an important factor in the performance of distributed transactions. It requires multiple messages in multiple phases.

It requires that the information about transactions be recorded stably to ensure that the transaction remain atomic even if there is a failure during the commit protocol. This is usually done by writing information to a log. When information must be stable at some point in the protocol, the log must be "forced", that is, the write must be completed before proceeding to the next step.Forced writes cost more than simple writes because they require actual I/O, whether a block of the log is filled or not[8].

III THE TWO PHASE COMMIT PROTOCOL:

Two phase commit (2PC) [16] is the simplest atomic commitment protocol, and is the most widely used in practice.The basic 2PC and three of its main variants: the presumed commit, the presumed abort, and the early prepare protocols.

A BASIC TWO PHASE COMMIT:

In its basic form, 2PC proceeds in two phases. In the first phase, known as the voting phase, the transaction coordinator exchanges one round of messages with the participants to solicit a "vote" on whether to commit or abort the transaction. During the voting phase, a participant validates that the transactions still holds all its locks as well as may perform integrity checks on affected data. Commit must be by unanimous votes. In the second phase, the decision phase the coordinator exchanges another round of messages with all participants to commit or abort the transaction as voted. No change is made in the state of the data at a participant site until the second phase, i.e., when the participant receives a commit or abort message from the coordinator[9].In the case of view consistency , there will be at most two rounds of the collection phase. A participant may only be asked to reevaluate a query using a newer policy by an Update message from the TM after one collection phase.For the global consistency case, the TM retrieves the latest policy version from a master policies server and uses it to compare against the version numbers of each participant[1].

IV TWO-PHASE VALIDATE COMMIT ALGORITHM:

The 2PV protocol enforces trusted transactions, but does not enforces the safe transactions because it does not validate any integrity constraints.Since the Two-Phase Commit atomic protocol commonly used to enforce integrity constraints has similar structure as 2PV, we propose integrating these protocols into a Two-Phase Validation Commit protocol. 2PVC can be used to ensure the data and policy consistency requirements of safe transactions. 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.

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 transaction manager enforces the same behaviour as 2PV in identifying policies inconsistencies and sending the updated 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.The server may be involved in concurrent interactions with many clients[12]. These credential schemes and associated protocols all address Every transactional object is represented by a header containing five fields: a pointer to an "owner" transaction, the owner's serial number, pointers to valid (old) and speculative(new) versions of the object, and a bitmap listing overflow transactions currently reading the object (threads that need to run in overflow mode compete for one of 32 global overflow IDs)[10].Private Information Retrieval has been proposed as a primitive for accessing outsourced data over a network, while preventing its storer from learning anything about client access patterns [11].For a scalable solution, the amount of computation and block accesses at the server should be minimized, because some limitations in ATN(Automated Trust Negotiation), they can be used only as fragments of an ATN process[13].
The validity of a proof of authorization is asserted in two cases:
1) (Credentials are syntactically and semantically valid):According to the definitions in [4], a credential *ck* is syntactically valid if the following conditions hold: (i) it is

formatted properly, (ii) it has a valid digital signature, (iii) the time _(*ck*) has passed, and (iv) the time !(*ck*) has not yet passed. A credential *ck* issued at time *ti* is semantically valid at time *t* if an online method of verifying *ck* 's status indicates that *ck* was not revoked at time *t'* and $ti \le t' \le t$.
2) (The inference rules are satisfiable):A policy is a set of inference rules that are encoded by policy makers to capture systems access control regulations. Given policy *P*, and user credentials *C*, if the inference rules of the policy can be satisfied using the user credentials, then the proof of authorization is said to be valid and the access is granted accordingly[14].
During normal operation, clients of cloud storage should not have to communicate with each other. If clients did communicate, they could simply exchange the root value of a hash tree on the stored objects to obtain consistency[16] In order to permit a requested operation, a reference monitor must verify evidence that the request should be granted. In classical approaches to access control, this evidence may be the presence of an authenticated identity on an access-control list, or the verification of a capability presented with the request. Several more recent proposals encode access-control policy and supporting credentials in a formal logic[17].

In order to avoid the subtle problems of weak atomicity in our Software transactional memory , we impose the programming restriction that each memory location must be accessed always within transactions or always outside transactions; we refer to these as transactional and non-transactional data. This restriction is only required within each interval between global synchronization points (e.g., global barriers); a memory location can be transactional in one such interval and non-transactional in another[18].

A system is strongly accountable if it provides each participant to determine for itself if others are behaving correctly, without trusting assertions of misbehavior by another participant who may itself be compromised.To illustrate strong accountability properties and the means to provide them for shared storage, a certified accountable tamper-evident storage service.The certified accountable tamper-evident storage service is a basic principle for network storage service,it enables the clients to read and write a shared directory of objects maintained by a certified accountable tamper-evident server[19].

The collection of credentials is used to satisfy a given authentication policy act as a partial snapshot of the system which the policy is evaluated.The correctness of an authorization decision depends on the stability and validity of the view during the policy evaluation[20].The obvious solution is to standardize the APIs so that a SaaS developer could deploy services and data across multiple cloud computing providers so that the failure of a single company would not take all copies of customer data with it. The obvious fear is that this would lead to a “race-to-the-bottom” of cloud pricing and flatten the profits of cloud computing providers[15].

V TRANSACTION TIMESTAMPING:

Timestamped voting was used both to optimize 2PC and to provide each committed transaction with a timestamp that agrees with the transaction serialization. This guarantees serializability even when transaction termination is not guaranteed, while permitting the read-only and other optimizations. Given the performance of the read-only optimization, and the fact that commercial commit protocols usually do not require transaction termination, this is important.
There are two cases that we need to consider.

A TIMESTAMPS FOR VERSIONED DATA:

To support transaction-time databases in which versions of data are timestamped with the commit time of the transaction , it is no longer sufficient to know only that a transaction has committed. We must know its commit timestamp as well. This means that we cannot presume commit since we cannot presume the timestamps. Obviously, we want the coordinator to garbage collect these entries once they are no longer needed. Hence presumed abort (PrA), which remembers the committed transactions, is better in this case because it can simply keep the timestamps with its committed transaction entries.

B TIMESTAMPS ONLY FOR COMMIT PROTOCOL:

So long as databases are using transaction timestamps not to timestamp data but solely as part of the commit protocol [5], it is not necessary to remember the timestamp of a committed transaction.The coordinator will have sent its COMMIT message with a timestamp that is within the bounds set by the timestamp ranges of all cohorts. If asked, the coordinator responds that the transaction was committed, and the cohort then knows that the commit time was within the timestamp range of its COMMIT-VOTE message.The cohort uses the knowledge of whether the transaction committed or aborted to permit it to install the appropriate state, before state in the case of abort, after state in the case of commit.

VI CONCLUSION:

In this paper,the prospective consistency problems will be identified that can arise as an transactional database systems are deployed on the cloud servers.The policy-based authorization system is used to protect the sensitive resources and it defines the notions of trusted and safe transaction.It introduces the different levels of policy consistency constraints,the different proofs of authorizations approaches is used to achieve the trusted transaction.The Two-Phase Validation Commit(2PVC)protocol is the enhanced version of the basic Two-Phase Commit(2PC)protocol is used to ensure the safe transaction.The use of H-MAC authentication is a digest and computed using a composite of the URI request timestamp.A timestamp is a sequence of characters or encoded information it identifies when a certain event is occurred it usually giving a date and time of day.

REFERENCES:

[1]. MarianK.Iskander et al.,"Balancing Performance, Accuracy and Precision for secure the cloud transaction,"IEEE Transaction on parallel and distributed system,feb 2014.
[2]. G. DeCandia *et al.*, "Dynamo: amazons highly available key-value store," in *ACM SOSP*, 2007.
[3]. P. Williams, R. Sion, and D. Shasha, "The Blind Stone Tablet:Outsourcing Durability to Untrusted Parties," Proc. 16th Annual Network and Distributed System Security Symp. (NDSS '09), 2009.
[4]. P. Williams, R. Sion, and B. Carbunar, "Building Castles Out of Mud: Practical Access Pattern Privacy and Correctness on Untrusted Storage," Proc. 15th ACM Conf. Computer and Comm.Security (CCS '08), 2008.
[5]. Z. Wei, G. Pierre, and C.-H. Chi, "Scalable Transactions for Web Applications in the Cloud," Proc. 15th Int'l Euro-Par Conf. Parallel Processing (Euro-Par '09), Aug. 2009.
[6]. Venti:A new approach to archival storage ,Sean Quinlan and Sean Dorward et al *Bell Labs, Lucent Technologies.*
[7]. D.J. Abadi, "Data Management in the Cloud: Limitations and Opportunities," IEEE Data Eng. Bull., vol. 32, no. 1, pp. 3-12,Mar. 2009.
[8]. 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, 2011.
[9]. The Performance of Two Phase Commit Protocols in the Presence of Site Failures.M. L. LIU, D. AGRAWAL AND A.EL ABBADI , *Cal Poly, San Luis Obispo, October 8, 1997.*
[10]. An Integrated Hardware-Software Approach to Flexible Transactional Memory.Arrvindh Shriraman, Michael F. Spear, Hemayet Hossain, Virendra J. Marathe,Sandhya Dwarkadas, and Michael L. Scott Department of Computer Science, University of Rochester.
[11]. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan.Private information retrieval. In IEEE Symposium onFoundations of Computer Science, 1995.
[12]. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), 2007.
[13]. 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.
[14]. 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 CloudComputing (ICDCS-SPCCICDCS-SPCC), 2011.
[15]. Above the Clouds:A View of Cloud ComputingMichael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia.
[16]. 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.
[17]. L. Bauer et al., "Distributed Proving in Access-Control Systems,"Proc. IEEE Symp. Security and Privacy, May 2005.
[18]. Robert L.Bocchino Jr,Vikram S.Adve,Bradford L.Chamberlain,"Software Transactional Memory for Large Scale Clusters,"University of Illinois at Urbana-Champaign,2008.
[19]. Adyan R.Yumerefendi and Jeffrey S.Chase.,"Strong Accountability for Network Storage,"Duke University.
[20]. A.J. Lee and M. Winslett, "Safety and Consistency in Policy-Based Authorization Systems," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), 2006.