International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 3, Issue 01, January 2016
Available at http://internationaljournalofresearch.org

Proposal of a Two-Phase Validation Commit Protocol as a Solution to Guarantee Trustworthiness of Transactions on Cloud Servers

Maddirala Narasimha Rao[1]; G.Sreenivasa Reddy[2]; Y. Dasradh Ram Reddy[3]& Prof.Dr.G.Manoj Someswar[4]

1. M.Tech.(CSE) from Buchepalli Venkayamma Subbareddy Engineering College, Affiliated to JNTUH, Telangana, India.

2. Associate Professor & HOD, Dept. of CSE, Buchepalli Venkayamma Subbareddy Engineering College, Affiliated to JNTUH, Telangana, India.

3. Associate Professor, Dept. of CSE, Buchepalli Venkayamma Subbareddy Engineering College, Affiliated to JNTUH, Telangana, India.

4. Principal & Professor, Dept. of CSE, Anwar-ul-uloom College of Engineering & Technology, Vikarabad, RR District, Affiliated to JNTUH, Telangana, India.

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 research 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.
KEYWORDS: Two-Phase Validation Commit; Byzantine failure; Transaction Manager(TM); Certificate Authorities (CAs)

INTRODUCTION

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.[1] 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.

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 3, Issue 01, January 2016
Available at http://internationaljournalofresearch.org

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.

- We formalize the concept of trusted transactions.
- We define several different levels of policy consistency constraints and corresponding enforcement approaches that guarantee the trustworthiness of transactions executing on cloud servers.
- We propose a Two-Phase Validation Commit (2PVC) protocol that ensures that a transaction is safe by checking policy, credential, and data consistency during transaction execution.
- We carry out an experimental evaluation of our proposed approaches.[2]

LITERATURE SURVEY

In many distributed-memory parallel computers and high-speed communication networks, the exact structure of the underlying communication network may be ignored. These systems assume that the network creates a complete communication graph between the processors, in which passing messages is associated with communication latencies.[3] In this paper, we explore the impact of communication latencies on the design of broadcasting algorithms for fully-connected message-passing systems. For this purpose, we introduce the postal model that incorporates a communication latency parameter 1. This parameter measures the inverse of the ratio between the time it takes an originator of a message to send the message and the time that passes until the recipient of the message receives it.[4] We present an optimal algorithm for broadcasting one message in systems with n processors and communication latency , the running time of which is (log n log(+1)). For broadcasting m 1 messages, we examine several generalizations of the algorithm for broadcasting one message and then analyze a family of broadcasting algorithms based on degree-d trees. All the algorithms described in this paper are practical event-driven algorithms that preserve the order of messages.[5]

Cloud computing has been envisioned as the next-generation architecture of IT enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, 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.[6] In this article, we focus on cloud data storage security, which has always 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 homo morphic 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).[7] Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including: data update, delete and append. 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.

Potential users of cloud services often fear that cloud providers' governance is not yet mature enough to consistently and reliably protect their data. As the trend toward cloud-based services continues to grow, it has become clear that one of the key barriers to rapid adoption of enterprise cloud services is customer concern over data security (confidentiality, integrity, and availability). This paper introduces the concept of transparent security and makes the case that the intelligent disclosure of security design, practices, and procedures can help improve customer confidence while protecting critical security features and data, thereby improving overall governance. Readers will learn how transparent security can help prospective cloud computing customers make informed decisions based on clear facts.[8]

Although virtualization and cloud computing can help companies accomplish more by breaking the physical bonds between an IT infrastructure and its users, heightened security threats must be overcome in order to benefit fully from this new computing paradigm. This is particularly true for the SaaS provider. Some security concerns are worth

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 3, Issue 01, January 2016
Available at http://internationaljournalofresearch.org

more discussion. For example, in the cloud, you lose control over assets in some respects, so your security model must be reassessed. Enterprise security is only as good as the least reliable partner, department, or vendor. Can you trust your data to your service provider? This excerpt discusses some issues you should consider before answering that question. [9]

With the cloud model, you lose control over physical security. In a public cloud, you are sharing computing resources with other companies. In a shared pool outside the enterprise, you don't have any knowledge or control of where the resources run. Exposing your data in an environment shared with other companies could give the government "reasonable cause" to seize your assets because another company has violated the law. Simply because you share the environment in the cloud, may put your data at risk of seizure. Storage services provided by one cloud vendor may be incompatible with another vendor's services should you decide to move from one to the other. Vendors are known for creating what the hosting world calls "sticky services;" services that an end user may have difficulty transporting from one cloud vendor to another (e.g., Amazon's "Simple Storage Service" [S3] is incompatible with IBM's Blue Cloud, or Google, or Dell).[10]

A growing number of online service providers offer to store customers ' photos, email, file system backups, and other digital assets. Currently, customers cannot make informed decisions about the risk of losing data stored with any particular service provider, reducing their incentive to rely on these services. We argue that third party auditing is important in creating an online service oriented economy, because it allows customers to evaluate risks, and it increases the efficiency of insurance based risk mitigation. We describe approaches and system hooks that support both internal and external auditing of online storage services, describe motivations for service providers and auditors to adopt these approaches, and list challenges that need to be resolved for such auditing to become a reality.

A growing number of online services, such as Google, Yahoo!, and Amazon, are starting to charge users for their storage. Customers often use these services to store valuable data such as email, family photos and videos, and disk backups. Today, a customer must entirely trust such external services to maintain the integrity of hosted data and return it intact. Unfortunately, no service is infallible.[11] To make storage services accountable for data loss, we present protocols that allow a third-party auditor to periodically verify the data stored by a service and assist in returning the data intact to the customer. Most importantly, our protocols are privacy-preserving, in that they never reveal the data contents to the auditor. Our solution removes the burden of verification from the customer, alleviates both the customer's and storage services' fear of data leakage, and provides a method for independent arbitration of data retention contracts.

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.[12]

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.[13]

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources.[14] This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 3, Issue 01, January 2016
Available at http://internationaljournalofresearch.org

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.[15]

SYSTEM DESIGN

SYSTEM ARCHITECTURE:

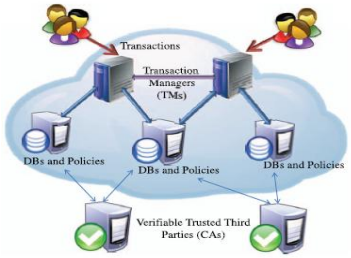
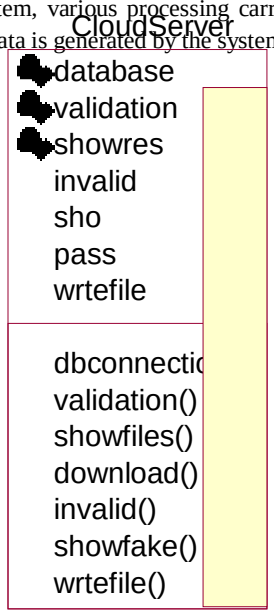


Figure 1: System Architecture

SYSTEM DESIGN

Data Flow Diagram / Use Case Diagram / Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.



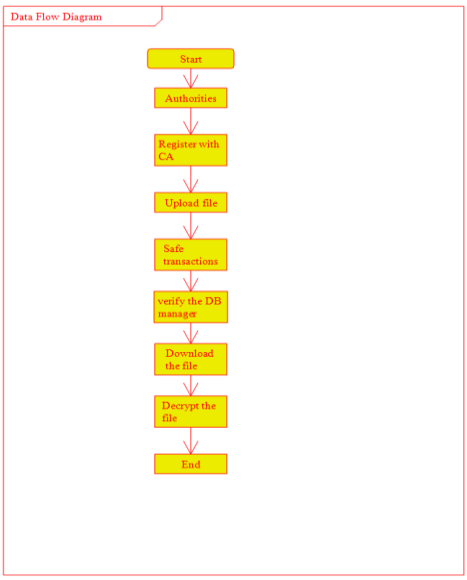


Figure 2: Data Flow Diagram

ACTIVITY DIAGRAM

An activity diagram is characterized by states that denote various operations. Transition from one state to the other is triggered by completion of the operation. The purpose of an activity is symbolized by round box, comprising the name of the operation. An operation symbol indicates the execution of that operation. This activity diagram depicts the internal state of an object.

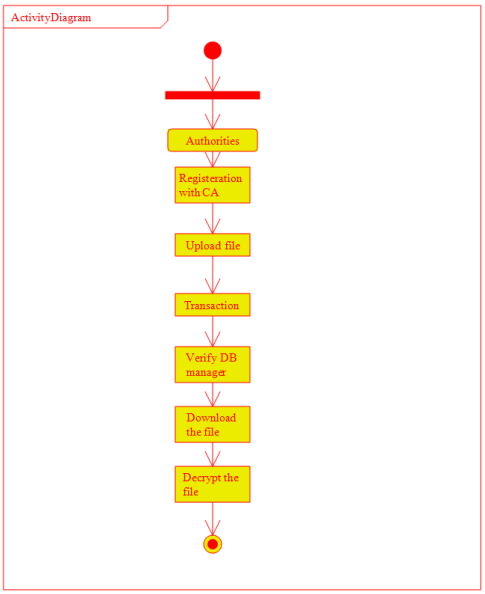


Figure 3: Activity Diagram

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 3, Issue 01, January 2016
Available at http://internationaljournalofresearch.org

UML SEQUENCE DIAGRAM

The sequence diagrams are an easy and intuitive way of describing the system's behavior, which focuses on the interaction between the system and the environment. This notational diagram shows the interaction arranged in a time sequence. The sequence diagram has two dimensions: the vertical dimension represents the time, the horizontal dimension represents different objects. The vertical line also called the object's *lifeline* represents the object's existence during the interaction.

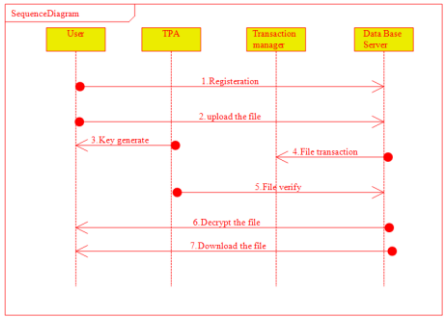


Figure 4: Sequence Diagram

CLASS DIAGRAM

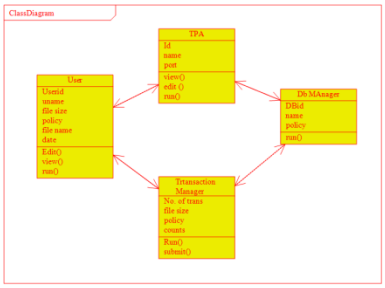


Figure 5: Class Diagrams

USE CASE DIAGRAM

A use-case diagram is a graph of actors, a set of use cases enclosed by a system boundary, participation associations between the actors and the use-cases, and generalization among the use cases.

In general, the *use-case* defines the outside (actors) and inside(use-case) of the system's typical behavior. A use-case is shown as an ellipse containing the name of the use-case and is initiated by actors.

An *Actor* is anything that interacts with a use-case. This is symbolized by a stick figure with the name of the actor below the figure.

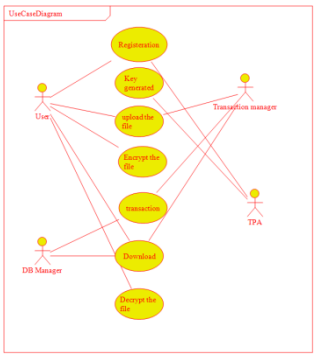


Figure 6: Use case Diagram

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free

from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

SYSTEM ANALYSIS

EXISTING SYSTEM:

To provide scalability and elasticity, cloud services oftenmake 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.

DISADVANTAGES OF EXISTING SYSTEM:

- Consistency problems can arise as transactional database systems are deployed in cloud environments and use policy-based authorization systems to protect sensitive resources.
- The system may suffer from policy inconsistencies during policy updates.
- It is possible for external factors to cause user credential inconsistencies over the lifetime of a transaction.

PROPOSED SYSTEM:

- We formalize the concept of trusted transactions.
- We define several different levels of policy consistency constraints and corresponding enforcement approaches that guarantee the trustworthiness of transactions executing on cloud servers.
- We propose a Two-Phase Validation Commit (2PVC) protocol that ensures that a transaction is safe by checking policy, credential, and data consistency during transaction execution.
- We carry out an experimental evaluation of our proposed approaches.

ADVANTAGES OF PROPOSED SYSTEM:

- Identifies transactions that are both trusted and conform to the ACID properties of distributed database systems.
- Guarantee the trustworthiness of transactions executing on cloud servers.

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 3, Issue 01, January 2016
Available at http://internationaljournalofresearch.org

- A transaction is safe by checking policy, credential, and data consistency during transaction execution.
- Most suitable in various situations.

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document,

such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

IMPLEMENTATION

MODULES:

1. Server Module.
2. Cloud User Module.
3. Transaction Manager.
4. Certificate Authorities.

MODULES DESCRIPTION:

Server Model

In this Module, We design a cloud infrastructure consisting of a set of servers, where each server is responsible for hosting a subset of all data items belonging to a specific application domain.

Cloud User Module

* In this Module, Users interact with the system by submitting queries or update requests encapsulated in ACID transactions.
* Since transactions are executed over time, the state information of the credentials and the policies enforced by different servers are subject to changes at any time instance, therefore it becomes important to introduce precise definitions for the different consistency levels that could be achieved within a transaction's lifetime. These consistency models strengthen the trusted transaction definition by defining the environment in which policy versions are consistent relative to the rest of the system. Before we do that, we define a transaction's view in terms of the different proofs of authorization evaluated during the lifetime of a particular transaction.

Transaction Manager

* 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.
* A common characteristic of most of our proposed approaches to achieve trusted transactions is the need for policy consistency validation at the end of a transaction. That is, in order for a trusted transaction to commit, its TM has to enforce either

view or global consistency among the servers participating in the transaction.

Certificate Authorities

* We use the set of all credentials, which are issued by the Certificate Authorities (CAs) within the system. We assume that each CA offers an online method that allows any server to check the current status of credentials that it has issued.
* In this module, we provide a 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).
* In this module, also develop Two Phase Validation system. As the name implies, 2PV 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.

RESULTS & CONCLUSION

Despite the popularity of cloud services and their wide adoption by enterprises and governments, cloud providers still 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. We used simulated workloads to experimentally evaluate implementations of our proposed consistency models relative to three core metrics: transaction processing performance, accuracy (i.e., global versus view consistency and recency of policies used), and precision (level of agreement among transaction participants). We found that high performance comes at a cost: Deferred and Punctual proofs had minimal overheads, but failed to detect certain types of consistency problems. On the other hand, high-accuracy models (i.e., Incremental and Continuous) required higher code complexity to implement correctly, and had only moderate performance when compared to the lower accuracy schemes. To better explore the differences between these approaches, we also carried out a tradeoff analysis of our schemes to illustrate how application-centric requirements influence the applicability of the eight protocol variants explored in this paper.

REFERENCES

[1] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), pp. 1-9, July 2009.

[2] Amazon.com, "Amazon Web Services (AWS)," http://aws. amazon.com, 2009.

[3] Sun Microsystems, Inc., "Building Customer Trust in Cloud Computing with Transparent Security," https://www.sun.com/ offers/details/sun_transparency.xml, Nov. 2009.

[4] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.

[5] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," http://www.techcrunch.com/2006/12/28/gmail-disasterreportsof- mass-email-deletions, Dec. 2006.

[6] J. Kincaid, "MediaMax/TheLinkup Closes Its Doors," http:// www.techcrunch.com/2008/07/10/mediamaxthelinkup-closesits- doors, July 2008.

[7] Amazon.com, "Amazon S3 Availability Event: July 20, 2008," http://status.aws.amazon.com/s3-20080720.html, July 2008.

[8] S. Wilson, "Appengine Outage," http://www.cio-weblog.com/ 50226711/appengine_outage.php, June 2008.

[9] B. Krebs, "Payment Processor Breach May Be Larges Ever,"

http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html, Jan. 2009.

[10] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[11] 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), pp. 598-609, Oct. 2007.

[12] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.

[13] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, http://eprint.iacr.org, 2008.

[14] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Netowrks (SecureComm '08), pp. 1-10, 2008.

[15] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.