International Journal on Recent and Innovation Trends in Computing and Communication
Volume: 5 Issue: 7

ISSN: 2321-8169
141 – 144

Secure Transaction Model for NoSQL Database Systems: Review

Mrs. Chaitra M
Asst. Prof, Dept. of CSE
SJBIT, Bengaluru, India.

Md Aquib Raza Usmani[1], Priyanshu[2], Ranjan Mishra[3], Rashid Ahmed[4]
SJBIT, Bengaluru, Karnataka

Abstract**—** NoSQL cloud database frameworks would consist new sorts of databases that would construct over many cloud hubs and would be skilled about storing and transforming enormous information. NoSQL frameworks need to be progressively utilized within substantial scale provisions that require helter skelter accessibility. What's more effectiveness for weaker consistency? Consequently, such frameworks need help for standard transactions which give acceptable and stronger consistency. This task proposes another multi-key transactional model which gives NoSQL frameworks standard for transaction backing and stronger level from claiming information consistency. Those methodology is to supplement present NoSQL structural engineering with an additional layer that manages transactions. The recommended model may be configurable the place consistency, accessibility Furthermore effectiveness might make balanced In view of requisition prerequisites. The recommended model may be approved through a model framework utilizing MongoDB. Preliminary examinations show that it ensures stronger consistency Furthermore supports great execution.

Keywords**—***NoSQL databases, cloud, multi-key, transactions, consistency, availability, efficiency.*

*****

I. Introduction

The concept of Big Data has led to an introduction of a new set of databases used in the cloud computing environment, that deviate from the characteristics of standard databases. The design of these new databases embraces new features and techniques that support parallel processing and replication of data. Data are distributed across multiple nodes and each node is responsible for processing queries directed to its subset of data. Each subset of data managed by a node is called shard. This technique of data storage and processing using multiple nodes improve performance and availability.

The architecture of these new systems, also known as NoSQL (Not Only SQL) databases, is designed to scale across multiple systems. In contrast to traditional relational databases which is built on sound mathematical model, NoSQL databases are designed to solve the problem of Big Data which is characterized by 3Vs (Volume, Variety, Velocity) or 4Vs (Volume, Variety, Velocity, and Value) model. As such, NoSQL systems do not follow standard models or design principles in processing Big Data. Different vendors provide proprietary implementation of NoSQL systems such that they meet their (business) needs. For instance, unlike traditional relational database systems which rely heavily on normalization and referential integrity, NoSQL systems incorporate little or no normalization in the data management. The primary objective of NoSQL systems is to ensure high efficiency, availability and scalability in storing and processing Big Data. NoSQL systems do not ensure stronger consistency and integrity of data.

They therefore do not implement ACID (Atomicity, Consistency, Isolation and Durability) transactions. However, it is important to provide stronger consistency and integrity of data while maintaining appropriate levels of efficiency, availability and scalability.

II. RELATED WORK

A. While the use of MapReduce systems for large scale data analysis has been widely recognized and studied, we have recently seen an explosion in the number of systems developed for cloud data serving. These newer systems address "cloud OLTP" applications, though they typically do not support ACID transactions. Examples of systems proposed for cloud serving use include BigTable, PNUTS, Cassandra, HBase, Azure, CouchDB, SimpleDB, Voldemort, and many others.

B. NoSQL Cloud data stores provide scalability and high availability properties for web applications, but at the same time they sacrifice data consistency. However, many applications cannot afford any data inconsistency. CloudTPS is a scalable transaction manager which guarantees full ACID properties for multi-item transactions issued by Web applications, even in the presence of server failures and network partitions.

C. Megastore is a storage system developed to meet the requirements of today's interactive online services. Megas- tore blends the scalability of a NoSQL datastore with the convenience of a traditional RDBMS in a novel way, and provides both strong consistency guarantees and high availability. We provide fully serializable ACID semantics within re-grained partitions of data. This partitioning allows us to synchronously replicate each write across a wide area network with reasonable latency and support seamless failover between datacenters [4] Deuteronomy: Transaction Support for Cloud Data.

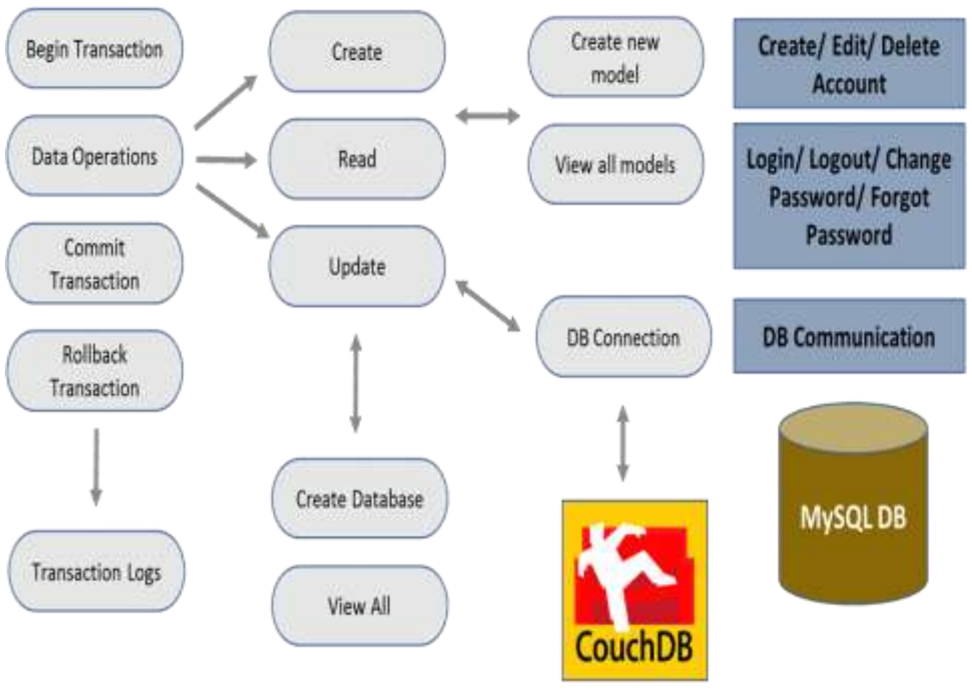
D. The Deuteronomy system supports efficient and scalable ACID transactions in the cloud by decomposing functions of a database storage engine kernel into: (a) a transactional component (TC) that manages transactions and their "logical" concurrency control and undo/redo recovery, but knows nothing about physical data location and (b) a data component (DC) that maintains a data cache and uses access methods to support a record-oriented interface with atomic operations, but knows nothing about transactions. The Deuteronomy TC can be applied to data anywhere (in the cloud, local, etc.) with a variety of deployments for both the TC and DC. In this paper, we describe the architecture of our TC, and the considerations that led to it. Preliminary experiments using an adapted TPC-W workload show good performance supporting ACID transactions for a wide range of DC latencies.

E. ANSI SQL-92 [MS, ANSI] defines Isolation Levels in terms of phenomena: Dirty Reads, Non Repeatable Reads, and Phantoms. This paper shows that these phenomena and the ANSI SQL definitions fail to characterize several popular isolation levels, including the standard locking implementations of the levels. Investigating the ambiguities of the phenomena leads to clearer definitions; in addition new phenomena that better characterize isolation types are introduced. An important multiversion isolation type, Snapshot Isolation, is defined.

F. We present the design and implementation of COPS, a key-value store that delivers this consistency model across the wide-area. A key contribution of COPS is its scalability, which can enforce causal dependencies between keys stored across an entire cluster, rather than a single server like previous systems. The central approach in COPS is tracking and explicitly checking whether causal dependencies between keys are satisfied in the local cluster before exposing writes.

III. SYSTEM ARCHITECTURE

The below figure shows a general block diagram describing the activities performed by this project. The entire architecture has been implemented in nine modules which we will see in high level design and low level design in later chapters.



Three major divisions in this project are

Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs.

Account Operations

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

Connection to Couch DB and Databases

Here, the end user can create a connection to the couch DB by specifying the host name and the port number of the installed instance. The end user can also connect to a remote couch db that is present in a different geographic location by just entering its host's IP address and the port number. The default port number will be 5984. However, the user can modify this during the couch DB installation. The user can create a new data base or view the list of all existing databases he/she have created using this module. The user can also grant the permission on the database to the other users of the transaction layer. By granting the permission, the user is allowing the other participant to perform any of the transaction operations on the database he/she have created. In addition to that, the user can delete the database or the user can also remove the permission he/she had granted to the other users.

Data Models

Here, the user can define the data model for his dataset. We are introducing the concept of data models to ensure that the user can perform various data operations on any type of the data. We are strictly refraining our self from hardcoding the data structure thus giving a freedom to the user to define his own data models. The user just have to define all the keys along with the type of the data that can be a probable value for that key, while defining the data model. The user can also view the list of all the data models he/she have created along with the possibility of deleting any of them if they don't wanted to use going forward.

Operations

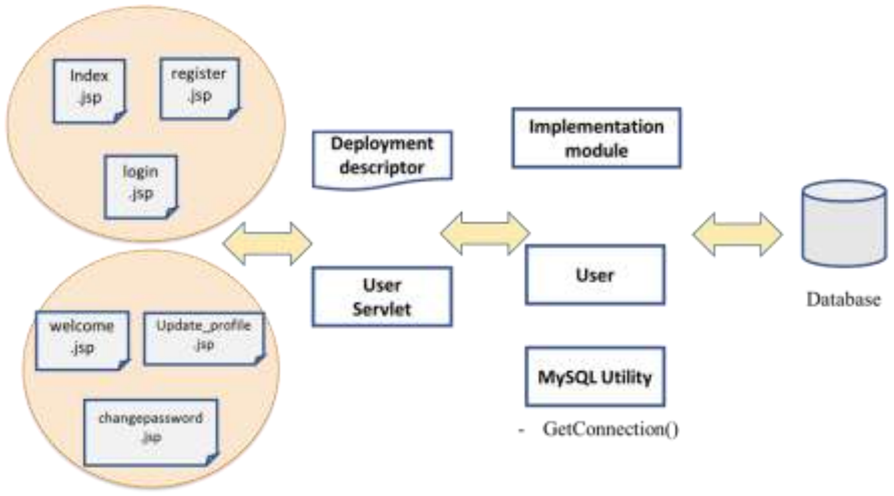
Here, the end user can perform various data operations. The possible data operations include the write access, read access, update, or the delete access. Before the user can perform any of these mentioned data operations, they will have to select the database against which the data operations must be performed. The data operations performed on this module will not be logging anything unlike the Transaction module.

Transactions

Here, the end user can initiate a new transaction and perform various data operations on as discussed in the previous division. Before the end user can perform the transaction, he/she will have to select the database against which the transaction has to be executed. The database the user is going to select will be either the one, he/she created him/herself or the one which the other users have granted the access to it. Each and every single operation in the transaction session will be logged in the local mysql table and will be available to view in the GUI. The end user can either rollback or commit the transaction after all the data operations he/she have performed.

Data flow diagram

A data flow diagram is the graphical representation of the flow of data through an information system. DFD is very useful in understanding a system and can be efficiently used during analysis. A DFD shows the flow of data through a system. It view a system as a function that transforms the inputs into desired outputs. Any complex systems will not perform this transformation in a single step and a data will typically undergo a series of transformations before it becomes the output.



IV. CONCLUSION

This project proposed a new model, called M-Key transaction model, for NoSQL database systems. It provides NoSQL databases with standard ACID transactions support that ensures consistency of data. The project described the design of the proposed model and the architecture within which it is implemented. As a proof of concept the proposed approach is implemented using real Couch DB database system.

Future work

Generalize the transaction layer to operate across multiple NoSQL systems so that the adoption of this layer is easy. Integrate the transaction protocol with the NOSQL system provided as a service on the cloud.

REFERENCES

[1] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears "Benchmarking Cloud Serving Systems with YCSB".

[2] Z. Wei, G. Pierre, and C. H. Chi, "CloudTPS: Scalable transactions for web applications in the cloud," IEEE Trans. Serv. Comput., vol. 5, 2012.

[3] J. Baker, C. Bond, J. Corbett, and J. Furman, "Megastore: Providing Scalable, Highly Available Storage for Interactive Services.," Proc. of the Conference on Innovative Data system Research (CIDR 2011), 2011.

[4] J. J. Levandoski, Lomet Mohamed F. Mokbel Kevin Keliang Zhao "Deuteronomy: Transaction Support for Cloud Data," Conf. on Innov. Data Systems Research (CIDR), California, USA.vol. 48, 2011.

[5] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A Critique of ANSI SQL Isolation Levels", 2007.

[6] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen, "Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS. In: Proc. of the 23rd ACM Symposium on Operating Systems Principles. Cascais, Portugal. 2011.

[7] D. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Systems," Commun. ACM, vol. 35(6), Jun. 1992.

[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. a. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," 7th Symp. Oper. Syst. Des. Implement. (OSDI '06), Nov. 6-8, Seattle, USA, 2006.

[9] S. Das and A. El Abbadi, "G-Store: A Scalable Data Store for Transactional Multi key Access in the Cloud," In: Proc. of the 1st ACM symposium on Cloud computing. Indianapolis, USA, ACM, 2010.

[10] A. Silberstein, A. Silberstein, B. F. Cooper, B. F. Cooper, U. Srivastava, U. Srivastava, E. Vee, E. Vee, R. Yerneni, R. Yerneni, R. Ramakrishnan, and R. Ramakrishnan, "PNUTS: Yahoo!'s Hosted Data Serving PLatform," Proc. 2008 ACM SIGMOD Int. Conf. Manag. Data - SIGMOD '08, 2008.