Transaction Model for NoSQL Couch DB

Anusha Nadig[1] , Nethravathi N P[2]

[1]*USN: 1BM13IS011, Department of ISE, BMS College of Engineering, Bengaluru, India*
[2]*USN: 1BM13IS043, Department of ISE, BMS College of Engineering, Bengaluru, India*

ABSTRACT

Now a days NoSQL cloud database systems are becoming very popular. They are built across thousands of cloud nodes and are capable of storing and processing Big Data. NoSQL systems have increasingly been used in large scale applications that need high availability and efficiency but with weaker consistency. Consequently, such systems lack support for standard transactions which provide stronger consistency. In this paper we proposes a new multi-key transactional model which provides NoSQL systems with standard transaction support and stronger level of data consistency. The strategy is to supplement current NoSQL architecture with an extra layer that manages transactions. The proposed model is configurable where consistency, availability and efficiency can be adjusted based on application requirements. The proposed model is validated through a prototype system using MongoDB. Preliminary experiments show that it ensures stronger consistency and maintains good performance.

Keywords:*Data mining, Misclassification Error, Gini index.*

1. INTRODUCTION

NoSQL database frameworks are new types of databases that are worked crosswise over a large number of cloud hubs and are equipped for putting away and preparing Big Data. NoSQL frameworks have progressively been utilized as a part of expansive scale applications that require more availability and efficiency however with weaker consistency. Thus, such frameworks need bolster for standard exchanges which give more grounded consistency. This venture proposes another multi-key value-based model which gives NoSQL frameworks standard exchange bolster and more grounded level of information consistency. The technique is to supplement current NoSQL engineering with an additional layer that oversees exchanges. The proposed model is built where consistency, availability and efficiency can be balanced in light of utilization necessities. The proposed model is approved through a model framework utilizing MongoDB. Preparatory tests demonstrate that it guarantees more grounded consistency and keeps up great execution.

The idea of Big Data has prompted a presentation of another arrangement of databases utilized as a part of the distributed computing environment, that go astray from the qualities of standard databases. The design of these databases grasps new components and procedures that bolster parallel preparing and replication of information. Information are conveyed over numerous hubs and every hub is in charge of preparing questions coordinated to its subset of information. Every subset of information overseen by a hub is called shard. This procedure of information storage and handling utilizing different hubs enhance efficiency and availability. The engineering of these new frameworks, otherwise called NoSQL (Not Only SQL) databases, is intended to scale over various frameworks. As opposed to customary social databases which is based on sound numerical model, NoSQL databases are intended to take care of the issue of Big Data which is described by 3Vs (Volume, Variety, Velocity) model. In that capacity, NoSQL frameworks don't take after standard models or plan standards in preparing Big Data. Diverse merchants give exclusive execution of NoSQL frameworks with the end goal that they meet their needs. For example, not at all like conventional social database frameworks which depend vigorously on standardization and referential trustworthiness, NoSQL frameworks join next to zero standardization in the information administration.

The essential target of NoSQL frameworks is to guarantee high efficiency, availability and scalability in putting away and handling Big Data. NoSQL frameworks don't guarantee more grounded consistency and honesty of information. They accordingly don't actualize ACID (Atomicity, Consistency, Isolation, and Durability) exchanges. Be that as it may, it is vital to give more grounded consistency and honesty of information while keeping up fitting levels of efficiency, availability and scalability Ease of Use

In this paper we propose a model that considers transactional principle of database frameworks. The goal is to give Consistency and to keep up the ACID properties while mulling over the accessibility and efficiency of NoSQL databases. The proposed model is based on the idea of Multi-Key transactions and is alluded to as Multi-

Key transaction model. It means to beat the difficulties of actualizing ACID exchanges in NoSQL databases. The proposed M-Key model takes after an loosely coupled model keeping in mind the end goal to separate transaction handling from fundamental information data storage and to guarantee transparency and abstraction.

2. RELATED WORK

Stonebraker inquiries NoSQL's capacity to fulfill the rising requirement for substantial scale online exchange preparing (OLTP) applications, for example, those requested by web based amusements, betting locales and interpersonal organizations [1]. Stonebraker contends that NoSQL is not suited for OLTP because of the requirement for ACID assurances to guarantee information exactness and respectability. The absence of local ACID certifications inside the NoSQL database places more prominent request on the application layer and makes questioning troublesome. As an other option to NoSQL, Stonebraker (2012) proposes that NewSQL frameworks are proper for reacting to the requirements of OLTP due to NewSQL's capacity to safeguard SQL ACID assurances while proceeding to offer elite and versatility. Another review examined roads to enhance NoSQL's execution and address the relative youthfulness of NoSQL versatility highlights (Thomson, et al., 2014). "Calvin" is an "exchange booking and information replication layer" that permits bunches of ease item machines to protect their scalability while likewise keeping up value-based assurances, even at solid consistency levels [7].

A study by Ghosh energizes a multi-worldview way to deal with database plan. Ghosh perceives that there are a vast assortment of NoSQL innovations, with differing qualities and shortcomings, and promoters utilizing a powerful mix of advancements so as to bamboozle their qualities while balancing their shortcomings. Ghosh additionally energizes the utilization of social databases so as to bolster NoSQL frameworks while giving extra capacities, including report era and review bolster.

As a method for binding together the distinctive advances, Ghost suggests an arrangement of offbeat informing, which would accomplish inevitable consistency between the advances [10]. In any case, the creator perceives that there are hindrances to multi-worldview configuration, including the quickly developing nature of NoSQL advancements, which could bring about similarity issues, and troubles that could emerge in adjusting NoSQL and social databases if the social database upholds ACID exchange guidelines with which NoSQL can't consent [8].

Different methodologies were proposed to solve the problem of transaction management in NoSQL databases. Be that as it may, in view of the differing flavors and sorts of NoSQL, there has been no acknowledged standard way of overseeing exchanges in NoSQL. Deuteronomy is a way of exchange handling in NoSQL. Deuteronomy isolates the value-based segment (TC) from the information segment (DC). The TC oversees exchanges and exchanges can traverse numerous DCs. Rather than the approach told in this project, Deuteronomy makes utilization of locking instrument to oversee concurrency and guarantee consistency. Locking is valuable yet it effects affects the execution of exchanges.

G-Store acquaints a key gathering convention with gathering keys for applications that need multi-push exchanges. Bunches inside G-store are dynamic and have a life expectancy. Subsequently gatherings will be erased after their life expectancy. Exchanges are restricted to inside a gathering and G-Store can't give exchanges crosswise over gatherings. Megastore, utilizes element bunches development like G-store. In any case, in Megastore, aggregate development is static and a substance has a place with a solitary gathering for the duration of the life expectancy of that element. Accordingly, ACID exchanges can just happen inside indicated gatherings.

COPS (Cluster of Order Preserving Servers), presents two factors called conditions and forms to save arrange crosswise over keys. It is executed utilizing a disseminated key esteem NoSQL database. CloudTPS, similar to Deuteronomy, make utilization of two layers' engineering which incorporates LTM (Local Transaction Manager) and the distributed storage. Exchanges are duplicated crosswise over LTMs to protect consistency within the sight of disappointments.

Limitation of existing approaches can be listed as: Because of the assorted flavors and sorts of NoSQL databases, there has been no acknowledged standard approach of overseeing exchanges in NoSQL databases. Deuteronomy approach effectively affects the execution of exchanges. Transactions are restricted to inside a gathering and G-Store can't give exchanges crosswise over gatherings. Megastore, amass development is static and an element has a place with a solitary gathering for the duration of the life expectancy of that substance. In that capacity, ACID exchanges can just occur inside indicated gatherings.

3. PROPOSED WORK

The potential commitments of the proposed model are outlined as takes after:

- The outline of another Multi-Key transaction for NoSQL frameworks that keeps up ACID properties of exchanges with a specific end goal to guarantee more grounded consistency.
- Development of a loosely-coupled architectural design that isolates the transactional logic from underlying information thus ensuring transparency and abstraction.

- Development of a model framework utilizing genuine NoSQL framework, MongoDB, which is assessed utilizing the YCSB+T benchmark in view of standard Yahoo! Cloud Services Benchmark (YCSB). The outcomes demonstrate upgraded consistency and performance.

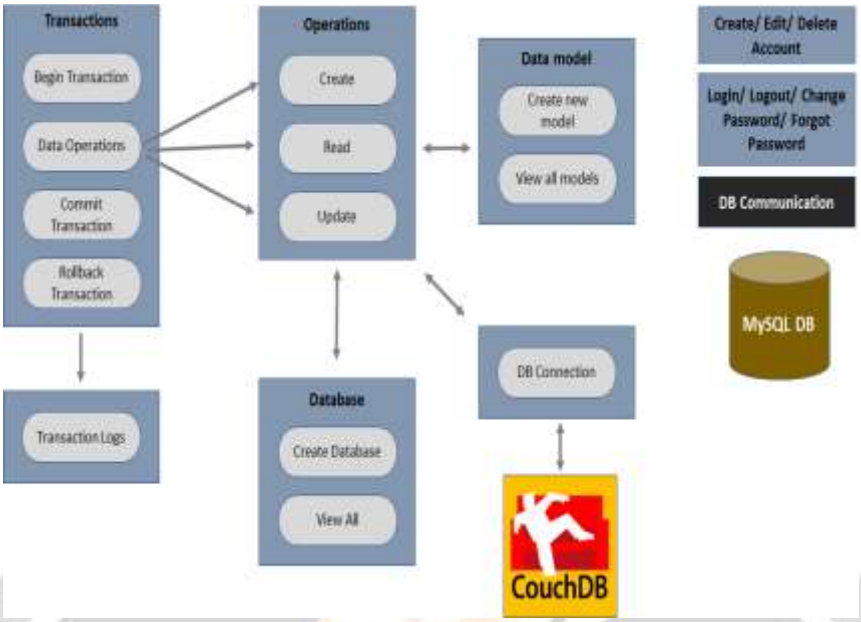

Fig-1: Proposed Architecture

Step 1: A client initiates a request to start a new NOSQL transaction (NST)

Step 2: TPE receives the request and generates an ID for the NST which is to be executed on NOSQL data.

Step 3: The Transaction Logs component will make a note of the start time of the transaction.

Step 4: The Transaction session will be started by the TPE within which the transaction ID will be written.

Step 5: The client then performs various operations on the NoSQL Database. The TPE sends each operation to the NoSQL data store to execute the operation. Also, the Transaction Logs component will be writing the details of each operation executed into the Transaction session.

Step 6: The TPE waits for the Client to receive the commit request or the rollback request. If the client doesn't provide any request, the transaction will be rolled back once the session expires. If the client provides the commit request, the TPE will save the session and invalidates it. If the client provides the rollback request, the TPE will contact the Transaction Logs component to retrieve the list of all the operations that got executed against the NoSQL system in this session. The TPE will then execute the counterpart of each operation against the NoSQL system to roll back the entire transaction.

a. The counter part of insert operation will be delete operation

b. The counter part of delete operation will be insert operation

c. The counter part of update operation will be the update operation

d. The read operation doesn't require any counterpart to roll it back since it doesn't modifies any data from the NoSQL system.

Step 7: The Transaction Logs component updates the transaction commit/ rollback time.

4. EXPERIMENT RESULTS

As a proof of concept the proposed approach is implemented using real Couch DB database system which does not support multi-key transactions. In the proposed model, transactional logic is implemented using Java language. Initially data models are created by specifying the attribute name and data type and can also view the data models already created along with their attributes and types. Then user can perform create, read and update operations. Under create operation user can specify the database and data model name and then start giving values to attributes. Under read operation user can specify the database name and fetch the data.

User can start the transaction by pressing begin transaction button and the perform all the operation and user can commit the transaction by clicking commit button and can roll back the transaction by pressing rollback button at

any point of time. User can view the details about transaction, when it started and when it ended. And what action was performed.

Fig-2: Snapshot showing the Transaction Committ and Rollback options

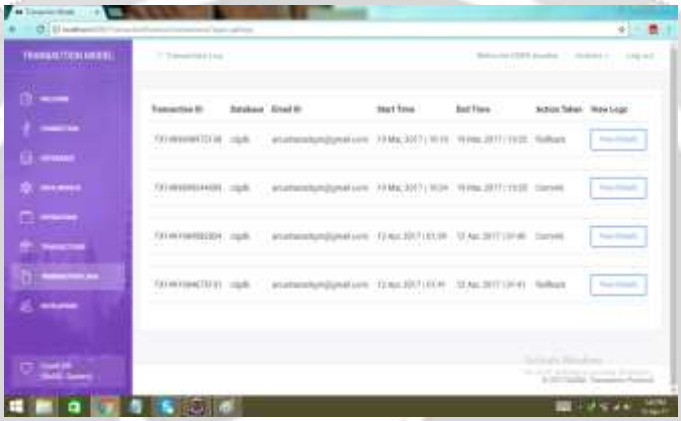

Fig-3 : Snapshot showing the Transaction logs

5. CONCLUSION

In this paper, we have 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. As a future scope we can generalize the transaction layer to operate across multiple nosql systems so that the adoption of this layer is easy. Also we can Integrate the transaction protocol with the NOSQL system provided as a service on the cloud.

6. ACKNOWLEDGEMENT

We thank Prof. Abhijith H V, Assistant Professor, Dept. of ISE, BMSCE for his constant guidance, useful inputs and motivation.

7. REFERENCES

[1] A. Dey, A. Fekete, R. Nambiar, and U. Rohm, "YCSB+T: Benchmarking web-scale transactional databases," Proc. - Int. Conf. Data Eng., 2014.

[2] 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.

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

[4] 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.

[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] 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.

[7] J. J. Levandoski, "Deuteronomy : Transaction Support for Cloud Data," Conf. on Innov. Data Systems Research (CIDR), California, USA.vol. 48, 2011.

[8] 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.

[9] 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.

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