Transaction Processing System for NOSQL Cloud Database System

Seokil Song*, Dojin Choi*
*Korea National University of Transportation, Republic of Korea
E-mail : sisong@ut.ac.kr, mycdj91@gmail.com

1. Introduction

Existing cloud database systems have been focused on scalability and availability than consistency. Consequently, the responsibility for data consistency rest upon application developers. However, according to [1], recently emerging applications such as online gaming, collaborative editing, and social networking, require strong consistency, which requires that there be only one copy of each data item, and only serializable access be permitted. In order to relieve this problem, several cloud databases [1-7] have been proposed, and some of them are proposed by major cloud service providers. However, most of them provide only limited transactional services. For example, some of them restrict data access to a partition of a database. Also, most of them employee optimistic concurrency control with lock methods that may reduce the response time of a transaction.

However, [3, 7] have different approaches. In their methods, a DB kernel is decomposed into a transaction component (TC) and a data component (DC). The TC provides concurrency control and recovery functions, and has logical level knowledge about keys and records without knowledge of physical structures such as pages and buffers. The DC provides access methods and cache management functions, but has no knowledge of how they are grouped in user transactions. This approach gives scalability and fast commit of transactions to cloud database systems.

These methods also have some problems. Because a TC has no idea about keys in tables, and it cannot request any lock on any specific key. Therefore, they proposed a partition covering lock. The partition covering lock uses partitioned key ranges of a table as lock resources instead physical resources such as pages and records. If the key ranges have the same size and the range is static, the method is hard to handle skewed keys. Too much lock contentions in the skewed key ranges may reduce the throughput and response time of transactions. Also, they use lock based concurrency control, so the performance may be restricted in distributed environment.

[1] proposed advanced methods to solve the problems of [3, 7]. It applies SI techniques instead of lock based concurrency control methods to [3, 7], and proposed a dynamic partitioning lock method to handle the skewed key problem. It monitors the number of transactions that access each partition and the number of records in each partition. Then, it splits any partition that is accessed more times than a particular threshold and maintains the split information using an index structure.

The methods proposed in [1, 3, 7] is difficult to be applied to existing cloud data stores. However, many cloud service providers and cloud applications already use or are based on existing cloud data stores. For them, a new approach is needed that it does not require the modification of existing applications and cloud database systems or require very small modification. In this paper, we propose a cloud transaction processing system (CTPS) that are loosely coupled with existing cloud data stores. The proposed CTPS enables existing cloud data stores to process transactional queries. In our method, the CTPS is similar to a TC in [1, 3, 7] and an existing cloud data store is similar to a DC in [1, 3, 7]. The CTPS analyzes the CRUD (create, read, update, delete) queries and transforms them to transactional queries. Also, it performs concurrency control and recovery in a similar way to those of [1, 3, 7].

2. The Proposed Cloud Transaction Processing System

In this paper, the proposed cloud transaction processing system (CTPS) enables existing cloud data stores to provide transactional functionalities with the small modification of them. The architecture of our proposed CTPS is shown in Figure 1. The CTPS consists of query analyzer, metadata manager, transaction manager, concurrency control manager, recovery manager and log manager. The queries from clients of cloud data stores are delivered to the query analyzer of the CTPS. The query analyzer analyzes a query with CRUD (create, read, update, delete) operations from a client, and transforms it to a transactional query. The query analyzer uses CRUD ontology that describes the CRUD operations of a certain cloud data store. The transactional query may contain locks, transaction control statement (begin, abort, transaction, rollback), log operations and CRUD operations.

For example, assume that a query from a client is “read A; A=A+1; write A;”. The query analyzer transforms the query to “begin transaction; r-lock (A); read A; A=A+1; w-lock (A); write A+timestamp; write A; return A to CTPS; log (A); If fail, rollback; commit;” by analyzing the query. In this example, we assume that the CTPS employee lock based OCC (optimistic concurrency control) method, so the transactional query includes lock and timestamp operations.

Like Deuteronomy [7], the proposed CTPS is loosely coupled with cloud data stores. Therefore, the CTPS maintain metadata store for the database in cloud data store. The metadata store contains the key range of each table, lock information and the state of each transaction. The metadata store may be accessed very frequently, so it is built

based on in-memory data store. The transformed transactional query is sent to a transaction manager. The transaction manager registers the transactional query as a transaction in the metadata store, and monitors the transaction's life.

The crud operations in the transformed query are delivered to cloud data store sequentially, and the cloud data store performs them. Before sending a CRUD operation, the CTPS checks the needs of lock operations. If lock operation is needed, the transaction manager sends lock operations to concurrency control manager. After the completion of the query, cloud data store returns the results. The log manager of the CTPS create log records by analyzing the returned results, and write them to log managed by the cloud data store. If the cloud data store returns fail messages, the transaction manager aborts the transaction with the created log records. Otherwise, the transaction is committed.

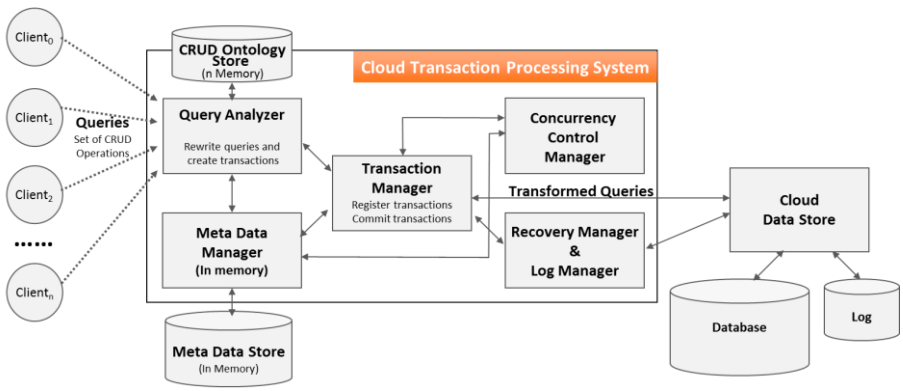

Figure 1. Architecture of the proposed CTPS

3. Conclusion

In this paper, we proposed a cloud transaction processing system that can be applied existing cloud data stores easily. Our proposed CTPS requires no modification to existing applications and cloud database systems or only small modification to them. The proposed CTPS is loosely coupled with existing cloud data stores. It enables existing cloud data stores to process transactional queries. It analyzes CRUD (create, read, update, delete) queries and transforms them to transactional queries, and provides concurrency control and recovery functionalities.

4. Acknowledgement

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the CITRC(Convergence Information Technology Research Center) support program (NIPA-2014-H0401-14-1007) supervised by the NIPA(National IT Industry Promotion Agency). Also, this research was supported by Technology Development Program for ('Agriculture and Forestry' or 'Food' or 'Fisheries'), Ministry for Food, Agriculture, Forestry and Fisheries, Republic of Korea.

5. References

[1] T. Kim, and S. Song, "Dynamic Partition Lock Method to Reduce Transaction Abort Rates in Cloud Data Management Systems", Cluster Computing Journal, accepted.
[2] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska, "Building a database on S3." In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 251-264.
[3] D. Lommet, A. Fekete, G. Weikum, and M. Zwilling, "Unbundling transaction services in the cloud", In Proceedings of CIDR, 2010.
[4] S. Das, D. Agrawal, and A. E. Abbadi, "ElasTraS: An Elastic Transactional Data Store in the Cloud", In Proceedings of USENIX HotCloud Workshop, June 2009.
[5] Z. Wei, G. Pierre, and C.-H. Chi, "Scalable Transactions for Web Applications in the Cloud", In Proceedings of the Euro-Par Conference on Parallel Processing, 2009.
[6] J. Baker, C. Bond, J. Corbett, J. J. Furman, A. Khorlin, J. Larson, J. Léon, Y. Li, A. Lloyd, and V. Yushprakh, "Megastore: Providing Scalable, Highly Available Storage for Interactive Services", In Proceedings of CIDR, 2011, pp. 223-234.
[7] J. J. Levandoski, D. B. Lomet, M. F. Mokbel, and K. Zhao, "Deuteronomy: Transaction support for cloud data", In Proceedings of CIDR, 2011, pp. 123-133.