Transaction processing in Cloud Environment

CLASSROOM RESERVATION SYSTEM (CLARE) - Final report

- Group 3(Anbang Xu, JIMAHN OK, Taewoo Kim)

1. Description of our Project

The CLARE(CLAssroom REservation) system is a transaction application which supports ACID properties on cloud environment. CLARE is designed for professors and students so that they can reserve classrooms, labs, and tools for their classroom related activities. CLARE runs on a simple database located on a cloud. Simple database only supports read, write, delete, and update and does not support any of ACID properties. We have developed features supporting transaction processing feature and ensures ACID properties on cloud environment.

2. Scope

Transaction processing on cloud environment

A. Implement Transaction Component(Session Manager, Record Manager, Log Manager, Lock Manager), Simple Data Component(Resource Manager)
B. Recovery Databases based on logs

3. Design and Operation Flow

Transaction processing is devised into two parts – TC, DC to utilize cloud environment. Client request operation to TC, and TC ensures ACID properties on operation and request atomic operation to DC. After DC executes that operation on the cloud database, DC will return results to TC and TC will finally return the result to client.

A. Client submits their requests directly to the TC using RMI invocation.
B. These requests are handled by TC Threads and the thread invokes Record Manager.
C. The Record Manager supports operations that include modifying or reading records from DCs.
D. The Record Manager calls Lock Manager to acquire relevant locks to the operation.
E. Lock manager checks that requested lock can be assigned and if it can generate lock without any conflict, it will return new LSN (Log Sequence Number) to the Record Manager. If lock cannot be generated, it will return error and Record Manager will abort the transaction.
F. After receiving new LSN, Record Manager calls relevant DC using RMI invocation and passes operation data along with new LSN. This operation is possible because TC does know about which DC can access to the requested table. However, TC does not know physical information about database.
G. When DC receives new LSN along with operation data, it connects to cloud database to retrieve BEFORE-IMAGE of record and applies (writes) AFTER-IMAGE to cloud database. If this operation is successful, DC will return operation results along with operation's LOG record to TC. At this time, if TC cannot receive the operation result, the operation will be aborted (UNDO) immediately on the cloud database.

H. When TC receives the result and log data(LSN, BEFORE_IMAGE, AFTER_IMAGE), it writes log record to disk and calls Lock Manager to unlock All locks that belongs to current transaction. Then it returns the result to client.

I. In addition, to ensure completeness of log record, TC periodically sends its current LOG LSN (CLSN) to DC to retrieve log record to make sure all log record is written to disk. After receiving CLSN from TC, DC will return all committed log record up to CLSN and checks previous CLSN (pCLSN) with currently received CLSN. If pCLSN is smaller than CLSN, then DC assumes that all log record up to pCLSN is written to disk and deletes all log record up to pCLSN.

4. Database Structure

Cloud Database has simple structure. It resides on Amazon S3 environment.

A. Physical Structure
   a. Cloud database is located at Amazon S3
   b. In S3, a bucket named CLARE is created to store our database.
   c. In CLARE bucket, five folders are created to store corresponding table. For example, CLASSROM folder store CLASSROOM table.
   d. In each folder, each file corresponds to one record. For example, “CS364,ICS Building” file is one logical record. Name of the file contains record information. File contains no content. We adjusted this concept to process fast in Amazon S3 environment.

B. Logical Structure
   a. There are five tables – CLASSROOM, LAB, TOOL, PEOPLE, RESERVATION.
   b. Structure of each table
      i. CLASSROOM – 2 column : name[PK] – String, location - String
      ii. LAB – 2 column : name[PK] – String, location – String
      iii. TOOL – 2 column : name[PK] – String, location – String
      iv. PEOPLE – 2 column : name[PK] – String, location – String
      v. RESERVATION – 5 column : people_name – String, reserve_type – INT, reserved_facility_name – String, starttime – String, endtime – String

5. Related Work

J. Levandoski, D. Lomet, M. Mokbel and K. Zhao. Deuteronomy: Transaction Support for Cloud Data, 2011.

6. Differences between Our implementation and related work

The basic structure (TC – DC) is same as Deuteronomy. However, in our implementation, due to time constraint, we simplified some aspects of related work. First, we implemented our lock mechanism which only locks at the table level granularity. In related work, they implemented multilevel granularity locking based on logical partition. Secondly, we regard each client request as one transaction. Therefore, dealing with transaction is simplified compared with related work. Third, no cache management is applied on DC. Therefore, DC needs to make sure it returns operation result to TC or UNDO operation immediately. In related work, they applied cache concept to DC so TC sends periodic LSN to DC to make cache to stable (database).

7. Test Cases

We created three test cases to check ACID properties.

A. Transaction Abort
   a. (Normal) A client requests an operation. TC passes this operation to DC. DC successfully writes the requested operation to cloud database. After that, DC returns the result of operation to TC. TC writes log to disk, release its acquired lock, and returns the result to client.
   b. (Failure) A client requests an operation. TC passes this operation to DC. DC successfully writes the requested operation to cloud database. However, when DC tries to return the result to TC, TC is not available to receive the result. Therefore, DC needs to UNDO the operation on cloud database and aborts the transaction.

B. Log synchronization
   a. (Normal) After TC receives proper operation results from DC, it writes log record to disk. Besides that, to ensure completeness of log records, TC periodically sends its current LOG LSN (CLSN) to DC to retrieve log record to make sure all log record is written to disk. After receiving CLSN from TC, DC will return all committed log record up to CLSN and checks previous CLSN (pCLSN) with currently received CLSN. If pCLSN is smaller than CLSN, then DC assumes that all log record up to pCLSN is written to disk and deletes all log record up to pCLSN.
   b. (Fail) After TC receives proper operation results from DC, it writes log record to disk; however, the writing operation might fail. Then TC misses log so that TC needs to retrieve missing log from DC. This is done by periodical log synchronization.

C. Lock Manager
   a. (Normal) Lock Manager is based on strict 2PL. A Shared lock or eXclusive lock can be assigned to a transaction if there is no conflict between current locks on lock table and requested lock. Once a lock is assigned, that transaction only releases lock when it is committing. If Lock Manager can grant lock request, it will generate new LSN and return that value to a transaction. In addition, lock conversion can be possible iff a transaction is requesting X lock on a specific table and only that transaction has S lock on the table.
   b. (Fail) When conflicting lock request is received to Lock Manager, it will return zero(error code) to a requesting transaction. Then the transaction should abort to ensure 2PL.

D. Recover DB from Scratch
   a. (Normal) Every operation is processed through WAL. Regarding the nature of cloud environment, we assume no database corruption or failure.
   b. (Fail) Somehow, if database is corrupted or missing, the database can be recovered by applying TC's transaction log. This operation is done by sending TC's log to DC and DC applies that log to corresponding cloud database.

E. Batch Insert
    a. Tests whether CLARE can handle multiple insert at a time to check Robustness of our implementation.

F. 2 DC scenarios
    a. The relation between TC and DC is extendible. For example, TC can communicate with multiple DC. This means that TC can ensure ACID properties on the multiple two or multiple databases. We will implement 2 DC structure to show this functionality is possible.

8. Lessons that we have learned

Given the size and nature of cloud environment, many cloud service relaxes some of ACID properties on the cloud database. However, we learned that by dividing Transactional Component (TC) and Data Component (DC), it is possible to strengthen ACID properties on the cloud database. This TC-DC structure is so flexible that we could implement multiple databases (DC) on the cloud database. Although our logic is simplified due to time constraint, it will be a good experience to implement extended transaction processing on the cloud without simplification.

9. References

D. Lomet, A. Fekete, G. Weikum and M. Zwilling. Unbundling Transaction Services in the Cloud, 2009.

J. Levandoski, D. Lomet, M. Mokbel and K. Zhao. Deuteronomy: Transaction Support for Cloud Data, 2011.

S. Das, D. Agrawal, and A. Abbadi. ElasTraS: An Elastic Transactional Data Store in the Cloud, 2009.

http://www.cs.washington.edu/education/courses/csep545/12wi

http://www.ics.uci.edu/~cs223/projects/