XIA - Using SQL or NOSQL for Content Chunk Cache

Heron Yang (heronyang@cmu.edu)

***Abstract*— In this note, we studied properties of MySQL (SQL) and MongoDB (NOSQL) databases that we concern when any of them is applied to the content chunk cache in XIA.**

I. Senerio

There are 4 different actions will be applied on the table (or collection) we designed.

A. Insert

This happens whenever there's a new and unseen chunk.

B. Delete

This happens whenever the record amounts is above the limit. In both MongoDB and MySQL, it's done manually after a new insertion happened.

C. Select (Read)

This happens whenever anyone requests a certain chunk of data by CID.

D. Update (Write)

This happens whenever anyone requests a certain chunk of data since the updated_at timestampe should be updated.

II. LRU

Neither MongoDB nor MySQL supports LRU by default. The way to remove the old record is to manually sort and delete. (More studies are needed.)

III. File in Database

Most of the materials I found online suggested that storing files in database is a bad practice; however, for our purpose, lots of concerns may not be applied. [1]

A. When to store files in the database

1) ACID: Atomicity, Consistency, Isolation, Durability
2) easier to backup

B. When to store files in the file system

1) portability (not every databases supports the same file field)
2) it's harder to deal with binary file from the database user perspective

C. Implementation

MongoDB offers GridFS, which chop files into 255k chunks and stores them. **MySQL** offers MEDIUMBLOB field, which stores 16MB data, or LONGBLOB for 4GB data.

IV. Thread Safe

Accessing from different threads should be critical in our cases since we rarely "write" and "read" in the same time. Only in the case that one thread is reading, and the other is trying to update the timestamp. In this case, we don't really need to lock the data since it's okay to have race condition on the updated_at timestamp field.

MongoDB: things we will used are all threadsafe (MongoServer, MongoDatabase, MongoCollection and MongoGridFS) [2] **MySQL**: use atomic update, use transactions, etc. [3]

V. Experiment

The testings were focused on MySQL 5.7.15 (SQL) and MongoDB 3.2.0 (NOSQL) on GENI Ubuntu 14.04 VM. Both of the database were configured with followings table (or collection) structure.

CID	hashcode in string
chunk	filepath
ttl	integer
created_at	create time
updated_at	update time

They were running a same task: **insert 2000 cache data in a single thread, and only keep the top 500 least recently used data.** I got their execution time as below:

MySQL	64.20 sec
MongoDB	1.87 sec

VI. Summary

MongoDB is faster in our simple test, and could be a good candidate for XIA cache. LRU can be performed manually, and TTL collections are supported by MongoDB by default [**?**]. As our use case is simple and fixed, it's easier to store the data in database first. For the next step, we can move on developing the database APIs and testcases, then, start to bridge the data from XIA Xcache.

References

[1] http://programmers.stackexchange.com/questions/150669/is-it-a-bad-practice-to-store-large-files-10-mb-in-a-database
[2] http://stackoverflow.com/questions/6574515/is-mongodb-thread-safe
[3] http://stackoverflow.com/questions/2364273/how-to-make-sure-there-is-no-race-condition-in-mysql-database-when-incrementing