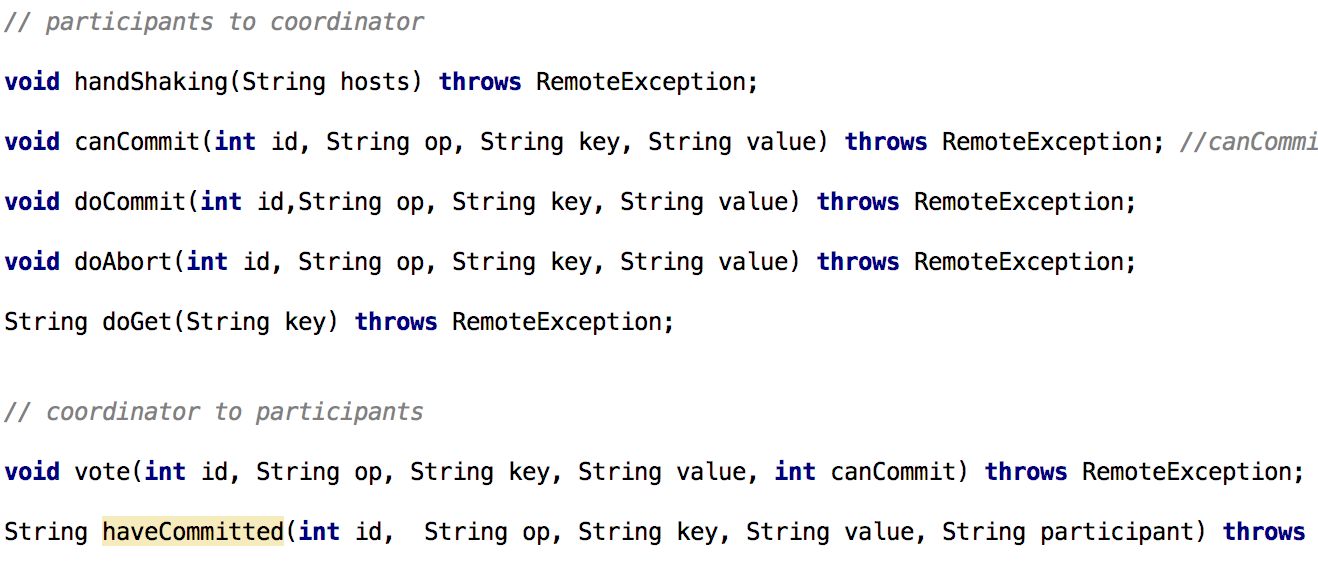
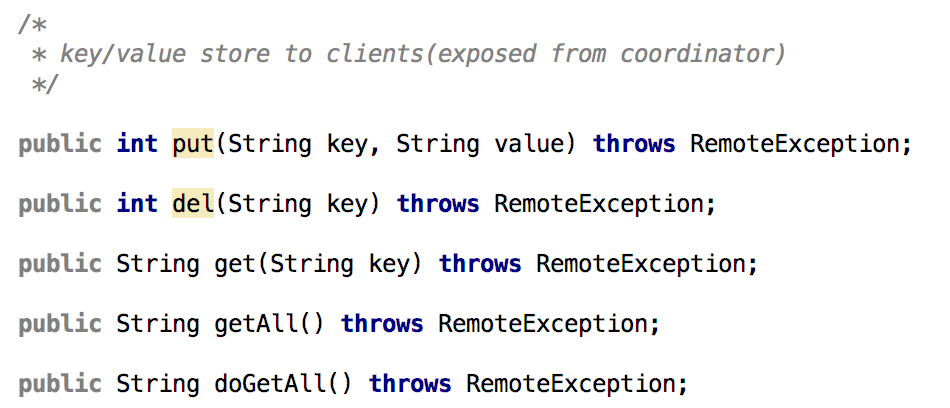
CS675\_ASSGN4 Kunling Zeng

**Structure of my code:**

* ***TPCNode*** is the Remote interface for 2PC protocol and key/value store, mainly divided into 2 categories:
  + 2PC protocol interface:



* + key/store interface



* ***TPCServer*** is the remote object implements *TPCNode*, each server (coordinator/participant) in the 2PC key/value store is a *TPCServer*.
* ***Coordinator*** is a subclass of TPCServer.

**RMI interface the replicas exposed to master:**

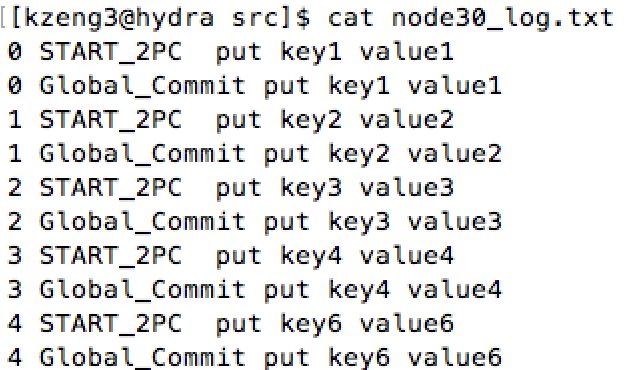
* ***handshaking***: for the master to involve the replicas in the 2PC key/value store
* ***canCommit***
* ***doCommit***
* ***doAbort***
* ***doGet***: called by the coordinator when it redirect a get(key) request from client to one of the replicas.

**Detecting failure:**

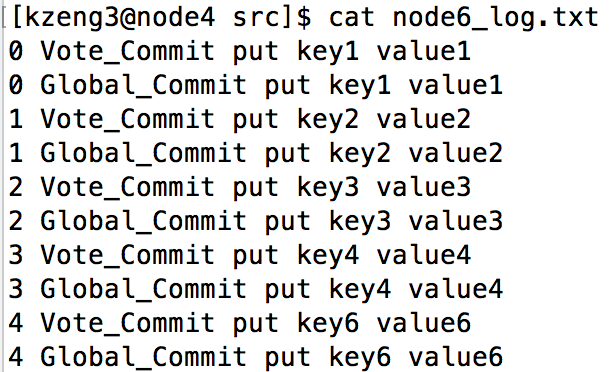
* between the ***coordinators*** and replicas: timeout and ***RemoteException***.
* master expose to client: no failure reported.
  + *get*: might return ***null***, but no distinguish between 1) no such key in database or 2) failure
  + *put/del*: always return 0. Might implement returning 1 as error in the future.

**Test:**

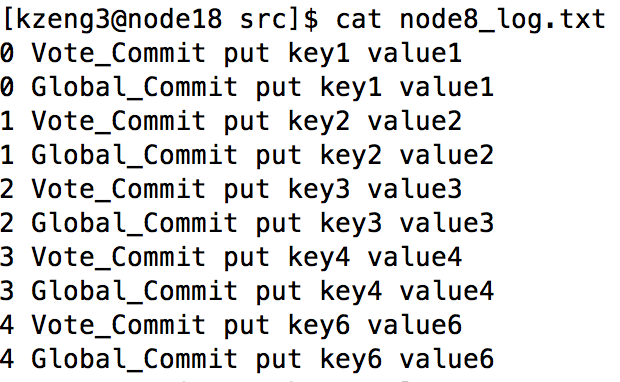
* Test is based on 3 *2PC* servers, 1 coordinator and 2 replicas, 2 clients
* Each line of log: *opId commit/vote/abort op key value*
* Single client issue sequential commands case is easier:



coordinator: node30



replica1: node6



replica2: node8

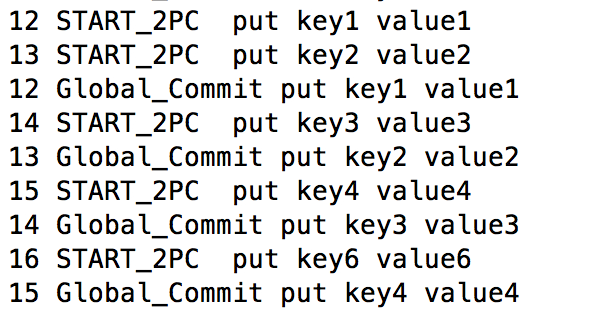
* + Could be observed the logs in different servers are consistent.
  + And the final state of 3 databases are consistent:



databases snapshot

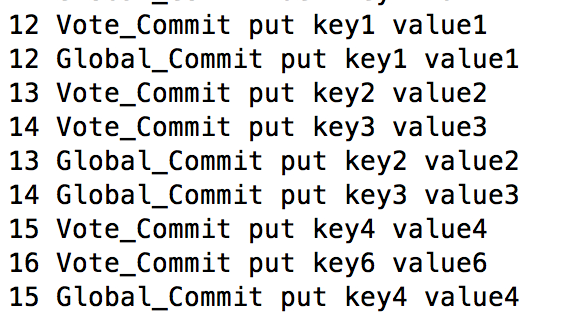
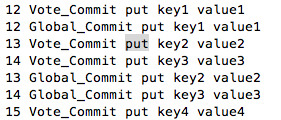
* Let 2 clients issue a sequence of commands, try to write the same key(make the client sleep for some period to make it more easy to incur conflict writing).
  + Concurrent Writing: different keys

-> From the log of coordinator(below), we could see multiple writing to different keys are concurrent:



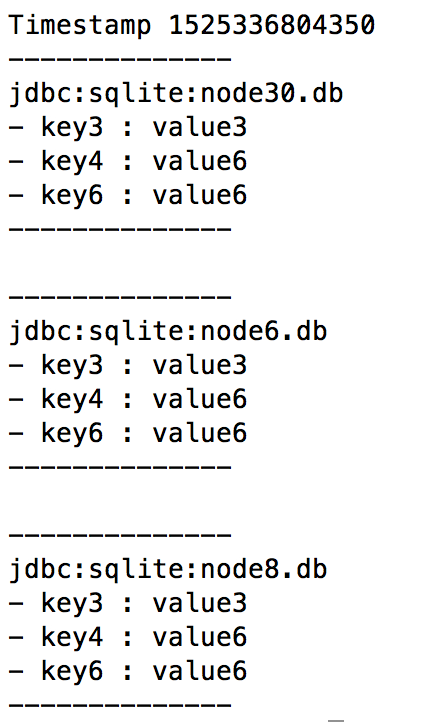
concurrent writing to key1, key2

-> From the log of 2 replicas, we could see they are concurrently writing different keys while maintaining consistent:



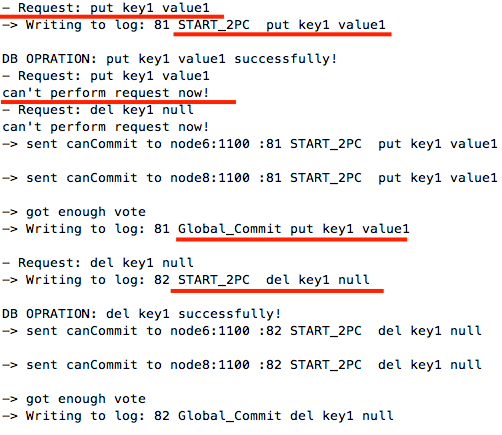
replica1 node6 replica2 node8

-> and eventually the databases remain consistent:



* + Concurrent writing: reject conflict writing:

-> one client issued “*put key1 value1*”, while master performing it, the other client also tried to write key1, being rejected. Only when *“put key1 value1*” is global committed, new write on key1 could started, i.e. the *“del key1”*



* + Couldn’t test abort successfully, while there are many conflict writing and compatible writing going on, couldn’t figure out what’s really going on, but the databases do maintain consistent state eventually.

Thank you!