ZOOKEEPER APPROACH

Joshua Gu

ID: 34466505

To achieve fault tolerance and consistency across replicated Cassandra apps, I used Zookeeper's data storage services in this approach. I used a strategy similar to Paxos RSM's for consistency, where the replicas elect a long lived leader. All client requests are forwarded to the leader as a PROPOSAL, the leader broadcasts a PREPARE for that request with an associated ID (which is the slot number), and the leader collects PREPARE-ACKS for that request. The leader organizes prepared requests in a queue by order of slot number. The leader then broadcasts DELIVER messages for requests at the head of the queue (lowest slot number) once they receive majority PREPARE-ACKS. Whenever a replica receives a DELIVER message, it executes the message on the Cassandra instance if that message is the next slot number it expects.

Zookeeper was used for fault tolerance. Under an election directory, each replica has an ephemeral node associated with it. The ephemeral nodes form an increasing order, as each time a replica is started/restarted, its node is added to the end of the list of nodes. The replicas agree that the head of the list of nodes is the leader, and all replicas watch that ephemeral node for any sudden deletion. If the node is deleted, all nodes agree on the new leader which is the head of the list. To achieve checkpointing, an approach similar to the checkpoint/restore functions of Gigapaxos were used. The leader is responsible for checkpointing as it manages the write operations. The leader reads all rows from its Cassandra database, stores it in a HashMap, encodes the HashMap as a json string, and writes that value in a persistent Zookeeper node

called state. Whenever a replica restarts, it calls restore, which reads the value of state and sets its Cassandra database to that state. All replicas call restore whenever the leader fails as well.

All tests would pass on my local laptop, but there are tests on Gradescope that fail. I mostly see test 36 and test 39 to be the ones failing. Given how they pass on my laptop, I suspect that it is due to Gradescope having a different processing speed than my laptop, such as my laptop executing the messages for consistency faster. This makes me think that Gradescope's slower speed causes some hidden concurrency/timing issues to show themselves in my application. Since both test cases involve restarting downed follower replicas, I imagine there is some timing error that occurs in my code for when a follower restores its state and processes messages that the leader is already proposing.

GIGAPAXOS/RSM APPROACH

Joshua Gu

ID: 34466505

To achieve crash fault tolerant and totally ordered consistency across replicated Cassandra applications, I used the gigapaxos approach where I implemented the 3 functions execute, checkpoint, and restore. To ensure each of these three functions were atomic, I wrapped them in synchronized(this) for thread safety. In the execute function, the application would send the String request to the Cassandra database by the keyspace. My implementation assumes there

is just 1 table in the keyspace, which is as far as the tests go as well. After the request is successful, the execute function returns true. Any consensus and crash recovery is handled by Gigapaxos, including the recovery of executions that alive replicas processed.

To implement checkpointing and restoring application state to a checkpoint, in the checkpoint function, the application queries the database for all rows in the table. The rows are stored in a hashmap of Strings as keys and values. The key is the combination of the primary key name and the value of the primary key. The value is the concatenation of all key value pairs associated with the primary key. The checkpoint function finishes by returning the JSON string representation of the hashmap.

When restore is called, the JSON string is recovered and converted back into the hashmap. We then iterate over all key value pairs and call the UPDATE and SET request to set all rows in the database to the version we have in the hashmap. With this setup, all tests pass.