CS578CC Programming Assignment 3.2

Instructor: Prof. V. Arun
CICS, UMass Amherst

Due: Dec 1, 2023

Name: Riya Deshpande
Student ID: 34011024

Name: Riya Danve
Student ID: 34038293

---

1 GigaPaxos Approach

1.1 Overview

To implement a fault-tolerant system in addition to the consistent system that we built earlier, we have used gigaPaxos as a Replicated State Machine (RSM) enabler. GigaPaxos provides a high-level API-like interface that can be wrapped around applications to enable the application servers to act as gigaPaxos servers or individual RSMs. Consensus properties like replica coordination, failure-recovery and ultimately fault-tolerance are taken care of by gigaPaxos implicitly.

As developers of the consistent and fault-tolerant application in this assignment, we have implemented the Replicable interface of GigaPaxos. The interface has the following three methods that need to be implemented

- boolean execute(Request request)
- String checkpoint(String name)
- boolean restore(String name, String state)

These methods are used to respectively execute a request, obtain a state checkpoint, and to roll back the state of a service. The detailed implementation of these methods specific to our application is explained in the following sections.

1.2 Method 1: execute(Request request)

In this overridden method, we check if the request received from the client is in the correct format as an object of RequestPacket and extract the actual value of that request. Further, we execute the request on the Cassandra database. In case there is an error in executing the request, we simply log it as an error, but the method always returns true.

1.2.1 Method 1.1: execute(Request request, boolean doNotReplyToClient)

This method is used when it is not required to send a response back to the client upon executing a request. The boolean flag set to true indicates that this request does not respond to a client. This method is called in scenarios like recovery of a server in a post-crash roll-forward case, or if this execution is part of inter-server communication.

In our application, we simply return the main execute(Request request) method so that the execution is atomically performed internally without responding to the client.

1.3 Method 2: checkpoint(String name)

The checkpoint method is called by gigaPaxos after a certain number of requests are executed. It is a safety feature to ensure that the state of the application is saved after an optimal amount of executions. When we checkpoint an application, it is easier for a recovered server to simply catch up to that state, rather than rolling through huge execution logs.

In this application, the checkpoint method returns the current state of the Cassandra database table. Initially, we perform a read operation on the table and store the values received in a properly formatted string variable. This variable is essentially returned by the method as the state of the application at that point.

1.4 Method 3: restore(String name, String state)

The most essential method in this implementation is restore. It is called when a server recovers from a crash. In this situation, the other replica servers might have moved forward in their slots and executed multiple requests from the client which the recovered server is unaware of. Instead of running through the execution logs, the server can simply recover to the most recent checkpointed state of the system. The method takes in the state returned by checkpoint, supplied to it, and restores the current server's state to that state.

Specifically in our application, the string returned by checkpoint is received by the restore function as a state. This string contains the values of the columns in the database table. The string is parsed to obtain these values and an insertion operation is performed on the recovered replica's table.
If there is any error in the application, the method logs that error, but always returns true.