Project 2 - Executive Summary

Hasti Gheibi Dehnashi

The following Excerpt is Copied directly from the Project 2 Specs on Canvas:

"Part of your completed assignment submission should be an executive summary containing an "assignment overview" (1 paragraph, up to about 250 words) explaining what you understand to be the purpose and scope of the assignment and a "technical impression" (1–2 paragraphs, about 200–500 words) describing your experiences while carrying out the assignment. The assignment overview shows how well you understand the assignment; the technical impression section helps to determine what parts of the assignment need clarification, improvement, etc., for the future. The grade for your executive summary is based on the effort you put into the assignment overview and technical impression. In general, if you put some effort into your writing, you will receive full credit for your executive summary."

Assignment Overview

The scope of this project is to extend our multi-threaded remote Key-Value Store such that the system has multiple app servers containing each containing a replica of the datastore. The app servers will have a coordinator that makes sure the data is replicate across all servers. To deal with mutual exclusion during concurrent access to the datastore's I wanted to implement a locking on the datastore during transactions using Java reentrant locks. Unfortunately I was up against the clock on this project and I did not get to implement the locking mechanism. I also was not able to get the server docker container to locate the coordinator docker container. To run this service there are three necessary steps:

1) Run as many server instances, each on their own port
2) Run the coordinator, providing it with each of the server hostnames and ports
3) Run as many clients per server instance by providing the client with the server hostname and port

I wanted to establish a gRPC bidirectional stream for communication between the server and the coordinator because the two would be in regular communication, however, after implementing the bidirectional stream I was unable to communicate across it, so I had to revert back to standard single message channels. The external Client API did not change between project 2 and 3, however I added an additional service to project 3, CoordinatorService, that was exposed on the coordinator to be called by the servers. The API in the CoordinatorService, operate, is called by the server forwarding a client request to the coordinator. Upon receiving an OperateRequest, the coordinator makes calls to each server to see if it can operate on their datastores.

Technical Impression

“the technical impression section helps to determine what parts of the assignment need clarification, improvement, etc., for the future”

In response to the prompt above, taken directly from the project spec, I respond with the following:

I understand that bidirectional streaming should be used whenever a stream of requests and responses will be shared between two hosts. As this was the case with the app server and coordinator, a bidirectional stream would have been the more appropriate communication method, however, I felt it was challenging to implement. The locking mechanism on the datastore is in place to maintain mutual exclusion to make sure that concurrent access to a CS doesn’t result in inaccurate data. Since there are multiple app servers and multiple clients in a system like this, it is necessary to lock the datastore during transaction to maintain data integrity. In java this is done with reentrant locks or concurrent maps. Finally, to increase fault tolerance the datastore is replicated across all web servers in case one crashes. In this assignment I did not get as far as reconnecting clients to a different app server in case theirs crashes, but I would like to implement that in my final project for better transparency.