CS455 Project 3 Whitepaper

Alex Lewtschuk and Kai Sorenson

Boise State University

CS455

Amit Jain

05-02-2024

Project 3

The purpose of this whitepaper is to outline the design choices that we decided on for part three of the Identity Server project. We will discuss the individual aspects of our project design as follows.

Client-Server Model

For our client server model we used a one-shot client with individual servers rather than a servent model. This one-shot model allowed us to develop the client and server side by side without worrying about overlapping functions. In our project the client connects to the server and serves its function before terminating at completion. Our server on the other hand runs indefinitely to serve clients and communicate with other servers for elections and coordination.

Coordinator

Our coordinator uses a bully algorithm that compares an arbitrary process id that is randomly assigned upon server startup. This algorithm is fairly scalable as each server is assigned a random number within a specified range of numbers. The server's listener and sender threads then send this information to the other servers to have an election and through the bully algorithm the server with the largest "pid" becomes the coordinator. If this server at any point dies, the remaining servers have another election and the new coordinator is chosen. If a new server joins it automatically subordinates itself to the established coordinator, even if the pid is larger than that of the coordinators. This subordination only lasts till the next election at which case if the new server has the largest pid it will become the coordinator and if not it will remain subordinate.

Consistency Model

Since Redis already provides features to implement a master-slave replica model in a server cluster, we do not have to implement our own. Note that since clients will only be able to connect to the coordinator, we do not need to worry about our servers synchronizing their data immediately. For this task, Redis uses an eventual consistency model where all of the data is congruently replicated, but not necessarily with immediacy.

Server Synchronization

Using REDIS as our database allowed us to use the inbuilt feature for REDIS synchronization. Specifically our system uses the default asynchronous replication method that REDIS provides. Asynchronous replication allows us send replicas to the other servers without waiting to hear if there is a response back from the destination servers. This lets the system maximize throughput and reduce latency which increases performance.