MP2.1: A Scalable Tiny SNS

Nachiket Umesh Naganure
UIN: 532008698
CSCE662: Distributed Processing Systems

System Design

The goal of the machine problem was to scale up the previosly implemented simple social network service using GRPC and c++.

1. **Requirements :**
   (a) Client contacts coordinator to get server details
   (b) Coordinator assigns clients to clusters and returns the server details.
   (c) Server registers itself with coordinator on start up
2. **Design approach Part 1: High Level Design**
   (a) We make use of GRPCs to satisfy our requirements. Mainly Unary RPCs and bidirectional RPCs.
   (b) We use structs, vectors to maintain details regarding clusters
   (c) New RPCs for servers are registerServer() to register with coordinator and heartbeat() to send keep alive messages to coordinator
   (d) New RPC for client is getServer() to get server details from coordinator on startup in order to establish connection.
3. **Desgin Approach Part 2: Low Level Design**
   (a) This won't be like a LLD, but we will discuss a few interesting aspects of the system's low level design.
   (b) Coordinator maintains a vector of structs for each cluster. The structs have server details.
   (c) On Startup the server will contact coordinator for registration and coordinator will store details in appropriate cluster.
   (d) On startup the client will contact coordinator to get ip address and port of its server.
   (e) Client uses get server rpc for above purpose.
   (f) Coordinator also keeps track of servers aliveness within threshold of 10 seconds. So when client requests a server, it won't return the server details unless the server has been alive through heartbeats.
   (g) We use a parallel thread to check heartbeats updates on the datastructures.

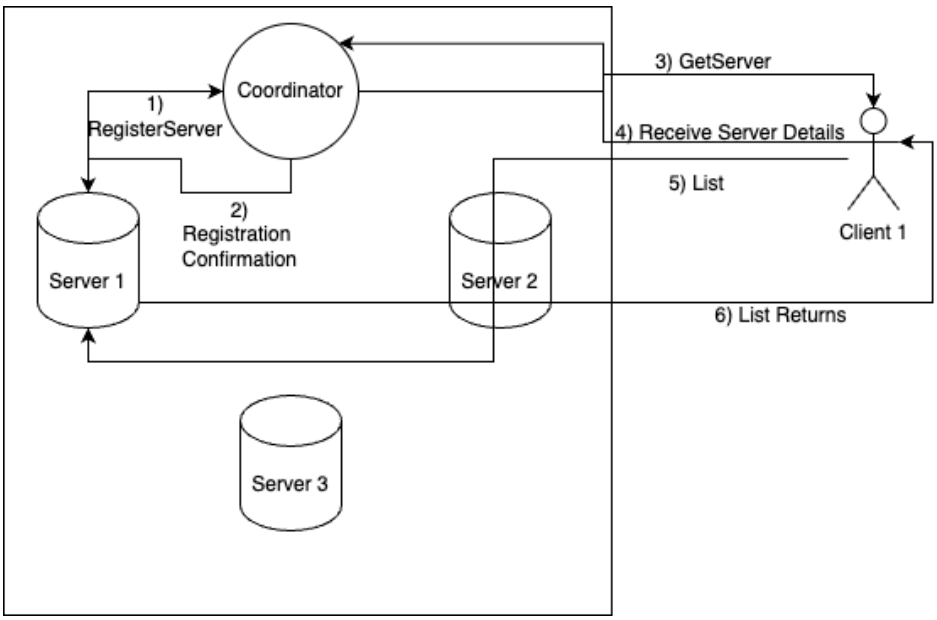

Figure 1: High Level System Diagram

Testing Scripts

To start coordinator, run ./coordinator -p 9090
To start server, run ./tsd -c 2 -s 2 -h localhost -k 9090 -p 3010
To start client, run ./tsc -h localhost -k 9090 -u 1

For ease of testing, I have created a script.
1) startall.sh - Starts coordinator and all three servers

1. Usage: startall.sh [start—stop]
2. start - Start the processes
3. stop - Stop the processes (kills all at once, coordinator and servers)