CSE 168: Lab 01: RPC

Martin Urueta

LAB: Tuesday 7:30pm-10:20pm

Adam Weingram

September 20, 2022

What was the round-trip time (RTT) or latency of a message between one gRPC client and one gRPC server?

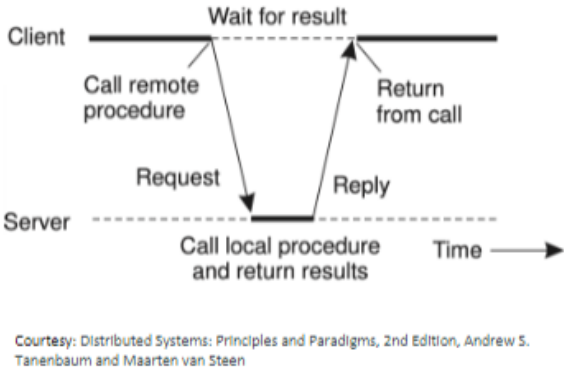

Figure 1: Synchronous RPC (Remote Procedure Call)

In order to create a Synchronous RPC as shown in Figure 1, I would first create a client and a server using Docker gRPC. After creating the container for Docker, I would need to run the "./greeter_server" and once that's up and running; I would call the server using "./greeter_client". Round-trip time(RTT) is when the client sends a message to the server and the server will reply back to the client. In my example, The server replies to the client with the message of 'hello world'. In order to calculate the latency of round-trip time between the client and server. I used a wall clock library called chrono that is able to calculate the latency of a message from the server to the client. I used this library in the greeter_client.cc to find the time it takes to do a RPC. In order to calculate the time I called "auto start = high_resolution_clock::now();" before RPC start and "auto stop = high_resolution_clock::now();" after RPC finished. Then to figure total time I used the library function "auto duration = duration_cast<microseconds>(stop - start);" to calculate latency. The latency is 1.21 milliseconds of a message between one gRPC client and one GRPC server.

```
./greeter_client
TOTAL TIME 0.00121 Seconds
```

What was the throughput (i.e., requests/sec or messages/sec) of one gRPC server when one gRPC client is running and when two gRPC clients are running?

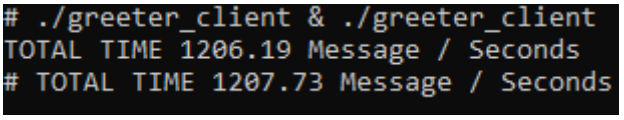

Figure 2: 2 client running

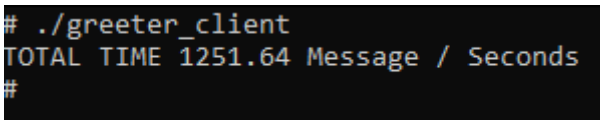

Figure 3: 1 client running

Throughout is the amount of RPC calls per second. The way we calculate the amount of message/ second is to call multiple messages to the server and receive a reply. I created a for loop to resend the same message to the server and to receive the same reply back to the client. I called 100 messages to the server to receive 100 replies back to the client. In order to find the message / second I used the same wall clock library to find latency and calculate the total time of each latency. We used simple algebra to determine our messages / seconds. (Total messages) / (total time) == 100 messages / (total time) we got a total 1251.64 messages per second as shown in figure 3. When we run 2 gRPC clients on one server we get 1206.19 messages per second on one client and 1207.73 messages per second on the second client as shown in figure 2. We add up our total throughput to be 2,413.92 messages per second. We got double the amount of messages when we run two clients parallel in one server. This is because running multiple clients will be better cause of it being logarithmic. Too many clients could also cause slower time per message because it can slow down the server.