CSCI 320: Networking and Distributed Computing Project 02 Timing

Riley Kirkpatrick

April 12th, 2020

After 9 runs of gRPC timing, I selected the run with the median total time. The following times are from that run and are divided into how long each function took to time (all times in seconds).

GetProductsByID:	0.1805684150000002
GetProductsByName:	0.15384835100000016
GetProductsByManufacturer:	0.12924276300000015
GetOrdersByID:	0.07656160900000009
GetOrdersByStatus:	0.15875302599999985
GetProductsInStock:	0.11105593099999966
UpdateProducts:	0.4868030550000002
UpdateOrders:	0.3059147299999996
AddProducts:	0.1334436910000001
CreateOrders:	0.3591622100000009
Total:	2.095353781000001

All functions interacting with products use 500 products, whether that means the function retrieves, adds, or updates them. Similarly, all functions interacting with orders use 50 orders, whether that means the function retrieves, adds, or updates them.

The choice of 500 products comes from the fact that gRPC has a 4 MB message size limit and using 750 or 1000 (or more) products exceeds this byte limit. In the future I could modify my program to use bidirectional streaming to overcome this limitation, with the caveat of increased complexity. Another way to handle large amounts of products and orders is to call the gRPC functions multiple times and only pass up to 500 products/orders at a time.

I modified my gRPC timing to run the timing back to back 10 times and then sum the times. Similar to running the timing once, after 9 runs of gRPC timing, I selected the run with the median total time to run and display the following times in seconds.

GetProductsByID: 1.8775839970000003

GetProductsByName: 1.5515576980000172

GetProductsByManufacturer: 1.2014333719999954

GetOrdersByID: 0.6999692510000135

GetOrdersByStatus: 1.7190775080000011

GetProductsInStock: 1.238777718999994

UpdateProducts: 4.405793927999987

UpdateOrders: 2.0425504680000035

AddProducts: 1.4805720659999917

CreateOrders: 2.3826795900000075

Total: 18.59999559700001

According to groups that completed both gRPC and XML-RPC, gRPC runs twice as fast as XML-RPC does (or even better). This makes sense since gRPC uses HTTP/2 whereas XML-RPC uses HTTP. So one would expect a gRPC server running the same function calls to complete them faster since it has less overhead. If I were making a commercial application I would definitely use gRPC since it is faster. Also, since the functions implemented do not rely on being run on a gRPC server, the majority of the work is writing the functions, not writing the gRPC server. Thus I do not think that choosing gRPC vs XML-RPC by which server is easier to implement matters, especially not as much as the speed uplift.