Scalability Test Report for HW4

This Report is about the result of our Homework 4 scalability test of ECE568. It will talk about our test in two main parts: how to test our program use our test cases and the test result of our program.

First of all, to test our program, you can go to the testing directory under our gitlab repo, where you can find a file called scalability_test.cpp, compile that file as sTest and run the following command `./sTest`, after you started our server. Basically, in the saclability_test.cpp file, we use a while loop to send many requests to the server and record the average execution time.

After Run our server on different cores virtual machines, we get a conclusion that because we used muti-threading in our program, when the number of cores increases, our running time will become shorter.

Below is the result of our scalability test on one core, two core Linux VM, we generate 1000 requests each turn.

Test turns	First turn	Second turn	Third turn	Last turn
Execute time	124 seconds	132 seconds	133 seconds	128 seconds

Test turns	First turn	Second turn	Third turn	Last turn
Execute time	124 seconds	132 seconds	133 seconds	128 seconds

We can get the conclusion that the average run time of our VM is 126.75 seconds which is our average latency.

Because we have 1000 request per turn, the throughput is 1000/126.75=7.889 requests/second.

Below is the result of our scalability test on four core Linux VM, we generate 1000 requests each turn.

Test turns	First turn	Second turn	Third turn	Last turn
Execute time	13 seconds	18 seconds	20 seconds	22 seconds

We can get the conclusion that the average run time of our VM on four cores is 18.25 seconds which is our average latency.

Because we have 1000 request per turn, the throughput is 1000/18.25=54.8 requests/second. Because we have sent requests in the same order on two core and four core VMs, then, we can conclude that our program has a good scalability and running on Virtual machine with more cores can significantly reduce our latency and increase the throughput.