HW4 Report

By Hongyang Li(hl404), Keyu Chen(kc487)

Scalability Test:

The project is built in C++ and uses its std::thread. Under the ./testing/scalability/ directory, we wrote a client.cpp which can imitate multiple client requests by creating multiple threads. To use it, just to enter ./testing/scalability/client [num of threads]. Then, it will send the server the num of requests randomly generated between create and transaction.

Experimentation Methodology:

The test was performed by generates 200 requests per run and measure the difference in runtime of total request under the condition of using 1, 2 or 4 cores. We use taskset -c 0,1,2,3 to control the cores. As we didn't implement thread pool, we cannot test for a big sum of requests, and we decided to create 200 threads.

Results:

As we didn't have enough time for testing, we only did this simple runtime test and created this chart for visualization.

Runtime under different cores

Running cores	1 core	2 cores	4 cores
1st	0.469714s	0.513943s	0.493191s
2nd	0.415065s	0.460001s	0.453052s
3rd	0.445627s	0.375497s	0.363022s
4th	0.464257s	0.420368s	0.493191s
5th	0.475787s	0.404713s	0.412393s
Ave. total runtime	0.45409s	0.434904s	0.44297s
Ave runtime/request	2.271ms	2.179ms	2.214ms

Analyzation:

The results were a little out of expectations. The Ave. runtime per request of 1 core is 2.271ms, which is the longest and is reasonable. However, the Ave. runtime per request of 2 cores is 2.179ms. It is a little longer than the Ave. runtime per request of 4 cores, which is 2.214ms.
We think the part of reason is that the sum of requests is not big enough to show the improvement on runtime with increasing num of cores. Another part of reason maybe that dealing with this task, there are no significant differences in runtime between 2 cores and 4 cores, but multiple cores show significant saving on runtime than only 1 core.