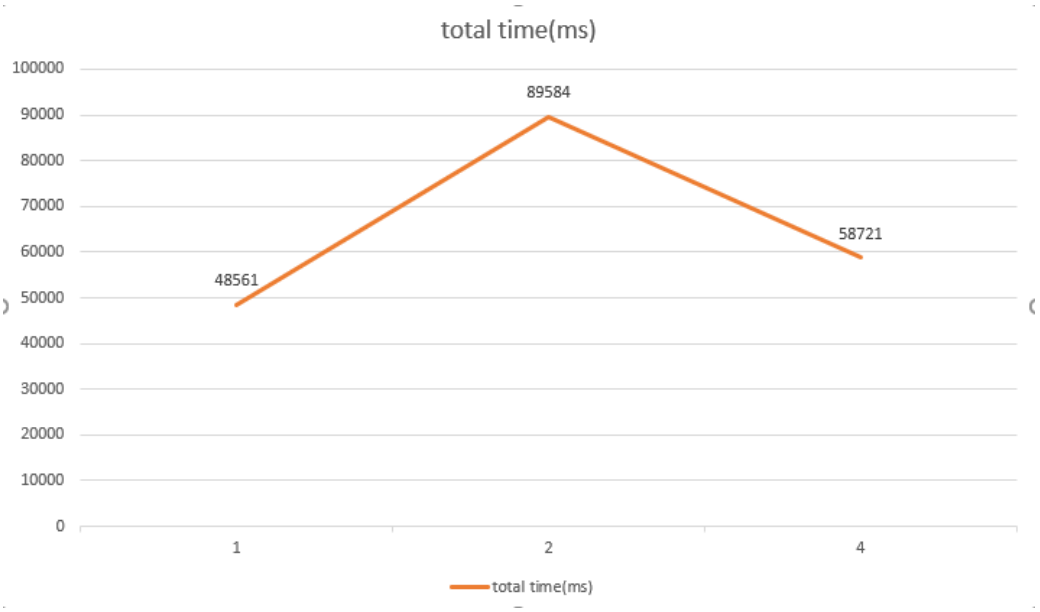
Homework 4 Report: Scalability analysis

by73 xl346

System version: Ubuntu 20.04.4 LTS

#core	thread pool capacity	#client threads	#request per thread	total time (ms)	avg time per thread	throughput (request/ms)
1	5	5	2000	21077	4215.4	0.4744508232
1	5	5	5000	48561	9712.2	0.5148164165
1	5	10	2000	39944	3994.4	0.500700981
1	5	10	5000	102866	20573.2	0.486069255
1	10	10	5000	116447	11644.7	0.42937989
2	5	5	5000	89584	17916.8	0.279067691
4	5	5	5000	58721	11744.2	0.425742068
1	10	10	10000	292147	29214.7	0.342293434
2	10	10	10000	285668	28566.8	0.350056709
4	10	10	10000	277807	27780.7	0.359962132
4	20	20	5000	270478	13523.9	0.369715836
1	1	1	2000	5003	5003	0.399760144
4	5	1	2000	4644	928.8	0.430663221
4	5	1	5000	13040	2608	0.383435583
4	50	50	10000	2199016	43980	0.227374425

Below is a scatter plot that shows the total amount of time taken (in ms) for different number of cores used



We observe that as we increase the number of cores and/or increase the number of threads while maintaining the total number of requests we send, there was no major decrease in total time used. We also observe that during all tests, our database, PostgreSQL, consumed way more CPU percent usage than our actual server did. For heavier loads (10k requests per thread), Postgres could easily take around 75% CPU usage, while our server only used around 10%. Thus, we decided that I/O operations that are related to the database formed the bottleneck of our entire system.

We can also see that the system achieves better throughput when less cores and less requests were used. This indicates that communication overheads such as context switching also had a major impact on the performance, since there will be more context switches as the number of cores goes up. Our result from the last experiment with 50 threads running in 4 cores supports this assumption by showing that it was not always better with more threads, since the throughput was only around 0.227.

In a nutshell, our system did not scale well with extra cores, and most of the efficiency lost were due to wait for database to complete I/O operations.