Surya Rani

Writeup Asg2

1.

After trying multiple large files with and without multithreading I can see clearly why we have learned and implemented this into our server. The ability to create multiple processes to handle multiple tasks simultaneously greatly increases the efficiency of the program. As throughput increases, or latency decreases we are able to increase parallelism of our program. The observable speedup with my program is slight but present in my program, if i were to have implemented multi-threading better it would be much more apparent.

My program though is limited with its lack of modularity, so i basically have each thread do the entire server function then unlock. With the time crunch to finish the assignment I thought this would be the best way to do it, as modularizing and only locking at certain parts would take much more time. If I did have more time this would be the next thing to do.

2.

Definitely by far the main bottleneck in my system is me locking the thread until it fully completes the entire server function so if we were to put or get a very large file then it would take a long time for one thread to finish and unlock so another thread could start. This is not the ideal and a good solution would be to modularize and only lock on parts that are changing shared resources.

The dispatcher is pretty good at concurrency as it is a single thread and continually adds to the queue. The worker as stated before is incredibly non -concurrent as each thread will finish the server first before releasing the lock of the mutex. This also affects my logging since I convert the data into hex before I release the lock adding even more time to each thread.

The main area I can increase concurrency is my doServer function. I just need to lock the threads until I'm done parsing the headers and then I can probably release the locks and let the threads do put,get, and head concurrently. If i were to implement this it would definitley increase the parallelism of the program greatly.

3.

Yeah logging the entire contents is a big no go. Firstly its a big pain in the ass to implement correctly. Secondly you would not want to log every single byte of a file because that would take forever especially on much larger files. In our program we are not testing huge amounts of data but in real life you never know how much data is being sent so it could be extremely time consuming, and energy consuming. If you must have logging, the best things to log are most likely the failures and why they failed.