I first tested multithreading using several sleep(1) calls in the middle of processes to see if the second and subsequent threads would start and start processing. This helped me identify several of my initial issues such as only thread 1 processing the requests. Then, I made scripts with multiple get requests as well as put requests to test multithreading.

I tested logging using small files, binary files, and large files to see if my data was being logged as expected. I tested multithreading and logging using curl and the specified log flag: -l logfile

- Although GET requests are very fast, there is a good improvement in the multithreaded when compared to the single threaded program.
- The multithreaded program is 3 to 4 times faster than the single threaded program.
- The bottleneck is likely my hardware since cannot handle more than a certain number of threads. The write speeds could certainly be faster. There isn’t much concurrency available in dispatch since only one thread is dispatching the client sockets to the worker threads. There is a good amount of concurrency available in the worker threads, with a similar amount available in logging. Increasing the number of threads would allow for more concurrency in worker and dispatch since more work can be done at the same time.
- In real life, we wouldn’t log the entire contents of the file. That would be illogical since the entire data is being copied and stored again. If the logfile is stolen, the entire server’s data is as good as stolen too. Not only does logging take space, it takes up valuable compute resources too.