Aaron Nguyen
1585632
Assignment 2 Writeup

Testing

For this lab the majority of the testing involved unit testing each function and using curl and netcat to test my code. Curl was used in the later stages of the program test GET and PUT header processing and responses in the server. The unit were used on the internal functions of the main functions that took care of GET and PUT. Most of the testing was spent using netcat to make sure the program was still functional while also testing multithreading. Telnet a linux command was used in the initial process of the code to test the activation and connection of a server to a client. Telnet was also used to test how to sending and receiving messages worked in c++ and with sockets.

MultiThreading	**real 0m2.629s**	**user 0m0.031s**	**sys 0m0.017s**
Single Threading	**real 0m4.153s**	**user 0m0.030s**	**sys 0m0.019s**

Is there any difference in performance?

The experiment results show that Multithreading is faster compared to Singlethreading

What is likely to be the bottleneck in your system? How much concurrency is available in various parts, such as dispatch, worker, logging? Can you increase concurrency in any of these areas and, if so, how?

The dispatcher thread is most likely the bottleneck in assignment 2. This is true because there is only one thread used as the dispatcher controls the assignment of tasks to the worker threads and the worker thread can not do anything unless assigned a task. If the dispatcher thread can not assign tasks as fast as the tasks are completed. The program will slow down because of the dispatcher. In assignment 2 the dispatcher has no concurrency because there is only one thread being run at a time to dispatch tasks. While workers and logging have multiple threads and locks which allow for more concurrent work to occur. Although this example did not test logging it made me realize that logging has less concurrency when compared to workers because logging there is still a wait time for writing to the file. To increase concurrency in logging and dispatcher is the increase in number of semaphore and buffs to improve concurrent use of dispatcher.