James Nail
2/27/2023

PThreads!

In this assignment, we had to create 3 nontrivial programs that would add up numbers in 3 different ways: free-for-all, using pthread locks, and using test-and-set (TAS) locks. The performance of all three was interesting, and time spent was very erratic (I did not get exact measurements), and I have reached a couple conclusions.

First, the free-for-all. I ran several tests using x = 10 and y = 10,000. The result should have been 100,000, but that is not what I consistently got. I got 100,000 1 time, and several 95,000, several 80,000, and several 45,000. Runtimes were wildly different, ranging from 0.1s to 0.3s, which is a lot in variance for a computer.

Second, the pthread locks. My tests were using x = 10 and y = 10,000. This finally ensured I got the right answer every time, that being 100,000, however runtime suffered a bit, jumping to 0.2s to 0.3s. I believe this is because all threads are forced to share the resource, which makes runtime suffer a hair. The answer, each time, was 100,000, which is a major improvement over free-for-all.

The final one analyzed was test-and-set locks. My tests were using x = 10 and y = 10,000. Once again, I got the right answer every time using this method, since it ensured only one thing could access the critical section at a time, something that was far from insured in the free-for-all method. Runtime was almost identical to pthread locking, clocking in at 0.2s to 0.3s, with some possible variance my poor phone stopwatch could not account for.

In conclusion, free-for-all was the fastest but definitely least accurate method. Answers were all over the board. Pthread and test-and-set were similar in runtime, but both produced completely accurate results, so in my opinion pthread and test-and-set locks are the better options.