Wei Fei (2538810)
EECS678 Lab5 Report
Prasanth Vivekanandan
03/03/15

1. What accounts for the inconsistency of the final value of the count variable compared to the sum of the local counts for each thread in the version of your program that has no lock/unlock calls?

Because the original version does not have lock and unlock functions. If the program runs without lock/unlock functions, a lot of threads (more than one) will run and share data by storing data in order to put back to the memory at same time. So an interleaving will cause losing an increment of count by a thread.

2. If you test the version of your program that has no lock/unlock operations with a smaller loop bound, there is often no inconsistency in the final value of count compared to when you use a larger loop bound. Why?

Because there are a lot of threads are sharing data if we have a large quantity of threads without locking and unlocking when the program is running. And a bigger loop bound will have more threads, which leads to no inconsistency.

3. Why are the local variables that are printed out always consistent?

Because local variable will not be shared like shared data since they are in stack.

4. How does your solution ensure the final value of count will always be consistent (with any loop bound and increment values)?

We can use lock and unlock ways to solve it (“pthread_mutex_lock” & “pthread_mutex_unlock”). The lock step can let the running thread runs until the thread finished, and the unlock step can unlock the finished thread.

5. Consider the two versions of your ptcount.c code. One with the lock and unlock operations, and one without. Run both with a loop count of 1 million, using the time time command: "bash> time ./ptcount 1000000 1". Real time is total time, User time is time spent in User Mode. SYS time is time spent in OS mode. User and SYS time will not add up to Real for various reasons that need not concern you at this time. Why do you think the times for the two versions of the program are so different?

When I was running the original code with the command "bash> time ./ptcount 1000000 1", I did not see the actual time after I waited for a while. May be it would take a really long time.

The reason the times for the two version of the program are so different is because of the significant of lock and unlock functions. If we did not use these functions, a lot of threads would run at same time. However, if we add them, the system will lock the rest of threads when the target thread was running, so it will save too much time and improve algorithm efficiency. Also, the SYS time and user time are different in two version, too, it may also cause the difference between two versions.