CS5460 Operating System - Spring 2013 HOMEWORK 3

Student: Binh Nguyen

February 28, 2013

1 Compilation note:

There are 4 flags for conditional compilation:

1. FENCE: enable multicore Barkery Algorithm lock.
2. SPIN_LOCK: enable spin lock.
3. FAIR_SPIN_LOCK: enable fair spin lock.
4. MUTEX: enable pthread mutex lock. **Exactly one** of these above flags is set in compiling command to make the compilation success. For example, if you want to *multicore Barkery Algorithm lock* for the mutual exclusion, then the compiling command looks like:
   gcc -DFENCE -o2 -Wall -Werror -Wextra problem_7.c -o problem_7 -lm -pthread

2 Results:

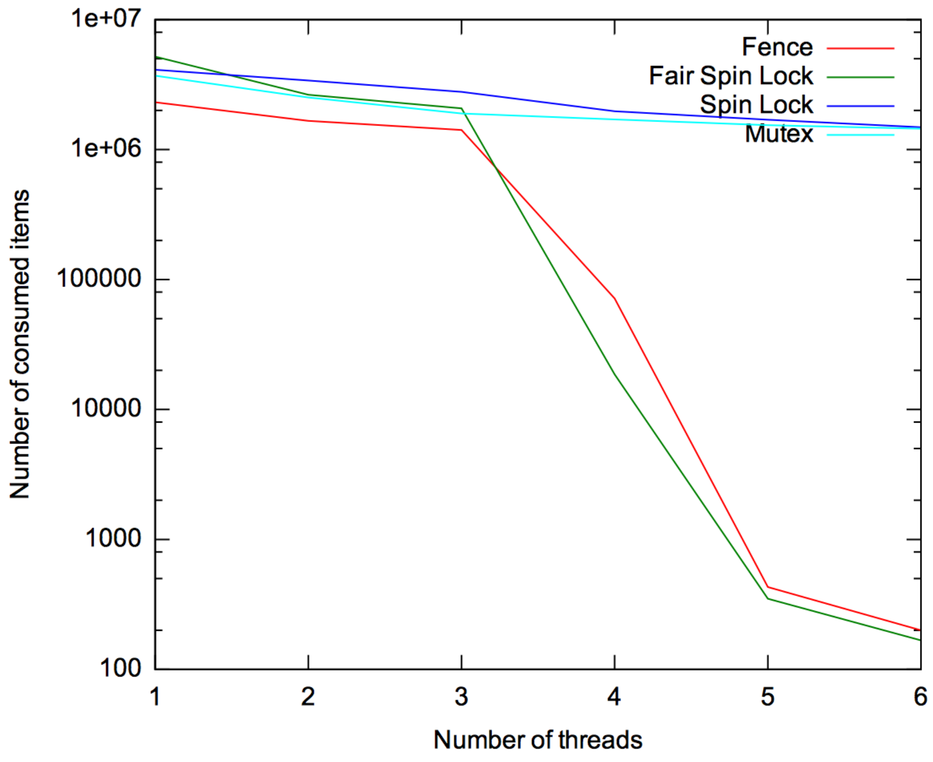

Figure 1: Number of threads and number of consumed items relation

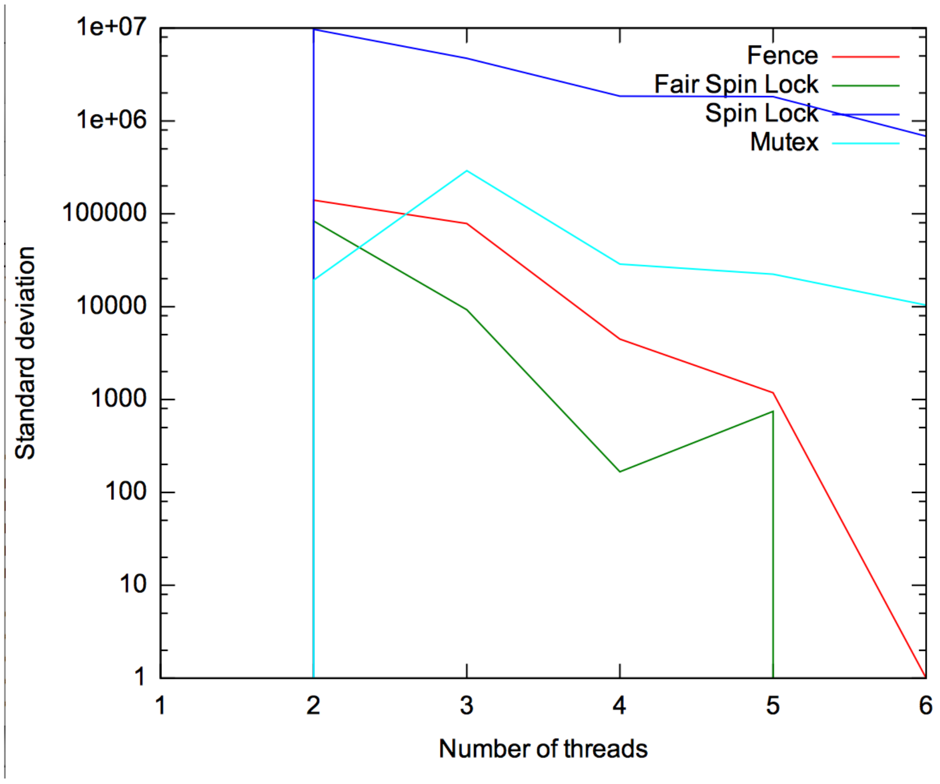

Figure 2: Number of threads and standard deviation of number of consumed items relation

3 Conclusions:

3.1 Number of consumed items:

1. Pthread mutex and Spin Lock are likely to scales the best and performs stably with an order of millions items consumed per second. Other locks do *scale* well (dramatically go down with 10+ threads).

3.2 Standard deviation of number of consumed items:

1. Despite of having high performance, Spin Lock and pthread mutex also have a high standard deviation.

2. Fair Spin Lock has the lowest standard deviation.

4 Some explanations:

1. Pthread mutex appeared to be among the best performance in term of performance. This satisfied what is expected because pthread mutex allows threads to *yield* CPU when they fail to obtain the lock.

2. Spin lock also appeared to have high performance. This is because spin lock uses *test and set*, which is basically an atomic read of memory, and only write to memory when the lock is free.

3. Although fair spin lock and fence have a good performance when the number of threads is less than 6, they do *not* scale well. This is because: (1) Fence lock need to synchronize all memory accesses of *all* threads before it is able to continue, when the number of threads is high, and multiple of them are doing busy waiting, then the waiting time could increase significantly. (2) Fair Spin Lock uses an atomic addition, however, unlike spin lock, fair spin lock has to compare the atomic addition returned

value with it own ticket number, this compare involves busy waiting, which lead to the degradation of performance when number of threads increased.

4. Fair Spin Lock appeared to have the lowest standard deviation. However, compared with the high performance, the standard deviations of Pthread mutex and Spin Lock are acceptable.

5. When doing this experiment on multiple CADE lab machines, I observed that the performance may varies on different machines due to different loads and different scheduling behaviors. In some cases, the performance could descrease to the order of hundreds items per second. However, overall, pthread mutex is likely the most outperformed and stable mechanism.