Assignment 3 Report

Instructor: Ryan Shea

Name: Maoshun Shang

Student ID: 301280479

Introductions and Environment:

This report briefly illustrates the timing cost for 6 different scheme of locks including both library built-in versions and user defined versions. All measurements are done on physical CSIL machines running Ubuntu 16.04 LTS with Intel Core i7-6700 CPU.

For the implementation of the six versions of the locks, the first two uses the built-in versions and the remaining four are defined in “sync.h” and implemented in “sync.c”.

Testing Cases and Corresponding Results:

Initial test is done by the given values: thread number = 4; iteration number = 1000000; testID = 0; workOutsideCS = 0 and workInsideCS = 1. For all of the following test cases, each test case is run on CSIL machine for 3 times and take the average. The test results (in ms) are as follows:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
average	436.33	129.3	832.33	406.67	46	454

The second test case is generated by changing the thread number to 8 and run the test again without changing any other variables. The results are:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
average	961	567	2362.3	1083	96	1022.3

The third test case is changing the thread number to the origin value, 4, and set the work inside the critical section to 10. All the other field are kept same as initial. The followings are the results:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
average	551	251.3	1373.6	447.3	88	1106

Also, for more details, I changed the number of work inside the critical section, to an extreme value. With workInsideCS = 100 and other values same, the result are:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
average	1492	1859.7	5661.3	1095	847.7	1140.3

The fourth test case is given by reset everything to initial value and change workOutsideCS to 10. The result of this test case is:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
average	147.7	313	852.3	463.7	92	476

I have extreme value test case for this scenario as well. With other field the same as before and worOutsideCS = 100, the result is:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
average	744.3	398.7	906.7	665.3	766.7	541

Initial Guess:

According to the results in the first two test cases, I believe the main parameter that affect running time is iterations. Therefore, some additional test are generated based on my guess:

	mutex lock	Spin lock	Spin lock(TAS)	Spin lock(TTAS)	Back off lock	Ticket lock
i = 100000	41	20	80	32	7	51
i = 10^6	456	161	640	315	73	532
i = 10^7	4138	1116	9409	4940	713	5032
i = 10^8	43535	18532	94872	40665	8531	52695

Therefore, when the iteration (overhead of critical section) grows, the actual time cost grows exponentially. This scenario is significant for TAS spin lock because thread outside critical section keeps adding contention to the lock. However, same scenario will not happen on back off lock and ticket lock because they have less or no contention for the lock.

Insights:

Base on the experiment data above, the time taken is mainly proportional to thread number, iterations, and critical section cycles. Work outside the critical section does not affect too much because locks did not involved for that part. More precisely, when the thread number doubles, the time for all kinds of locks almost doubles except spin locks. Because time elapse for spin lock grows exponentially when the number of thread increases. Over all the user defined spinlocks, the most efficient is TTAS spin lock. The reason for time saving is the testing for lock value during lurking stage. Over all clocks, most time-consuming lock is TAS spin lock, which adding the overhead of the lock by all threads outside the critical section keep trying to unlock the lock and create contention. For all the six locks schemes, mutex (back off) lock and ticket lock are the most time-efficient locks. Back off mechanism reduce the lock contention by the thread sleep for a small period of time while ticket lock get a ticket for each thread and each thread wait until their turn. In addition, ticket lock avoids starvation while other locks cannot. However, the back off lock ask thread to sleep will lead to increasing risk of important interrupt missing. In spite of that, the size of critical section determined which sleep function to use. (I used usleep function in this case because critical section is short enough, which is less than 1 second, to use sleep function call.) Also, ticket lock does not allow threads to have their priority, which can lead to a defect although it is time efficient for users.

Conclusions:

Therefore, based on the experiments and the deduction above, it is up to the programmer to decide which lock to use. If the critical section is large, it is the best to choose ticket lock for both efficiency and starvation-avoidance. If the critical section is small enough, spin lock is the best choice because it is responsive and will not miss significant interrupts.