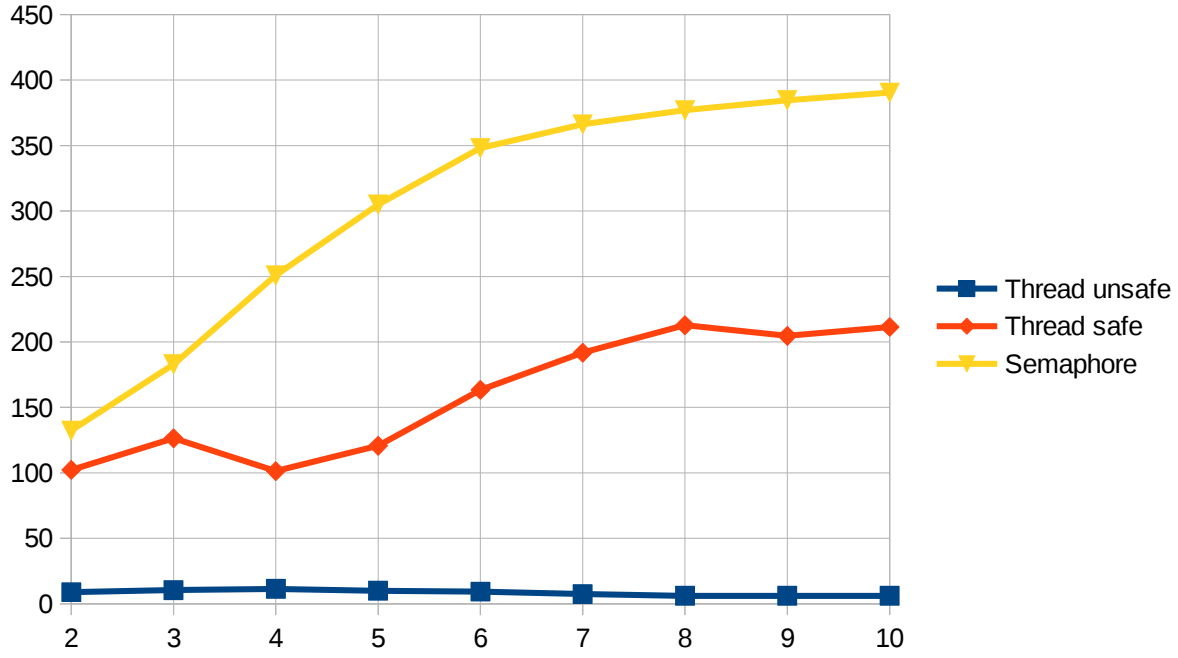
CSE231 (Operating Systems) Assignment 4 : OS Impact Analysis

Anshuman Suri (2014021)

All of the counters were made to count up to 10^9 (averaged over 50 iterations to account for other running programs) using 'x' threads, where 'x' was varied in [2,10]. The following are the results (data plots with their analysis).All tests were run on a machine with 8 virtual cores @ 2.8GHz (Turbo Boost upto 3.2GHz)

Thread safe (Semaphore) :

The method of using semaphores for counting uses locks for non-atomic operations and is thread safe.So, correctness is maintained for any number of threads.



The performance,however, seems to be worse than a thread safe counter using locks, even though mutex locks and semaphores block any calling threads which do not own the lock for the critical section.This is because the overhead of semaphores is a lot more than the overhead of a mutex lock. Semaphores are designed to be shared between different processes.So the value of a semaphore is stored somewhere else.Hence, the overhead of accessing that value and changing it makes it slower than a mutex lock.Moreover, a semaphore will switch to kernel as soon as it encounters a wait() or post().On the other hand, a mutex lock will do so only if the the code it accesses is locked by another process.

So, the performance of the semaphore based lock decreases with time,whereas correctness remains the same.To sum it up,for the given machine and task, using a spin-lock based counter (with 4 threads) turns out to be the optimal solution.