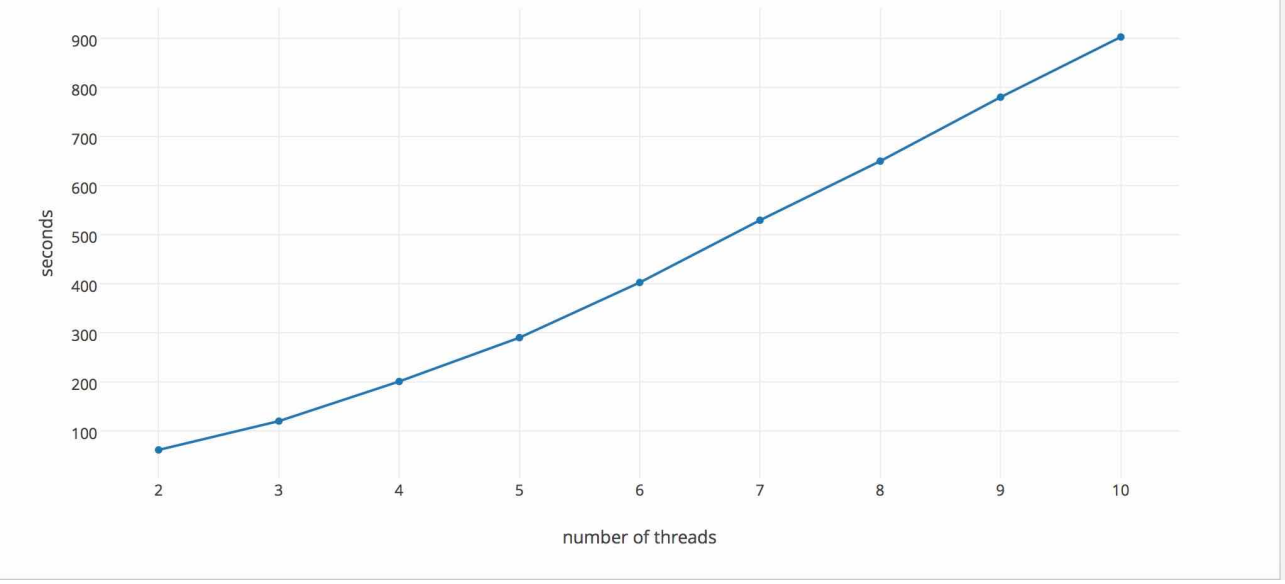
14105_pdf2

Terminal data of 14105_mtp_c1_sem

time taken 61.317453 by 2 threads
counter value is :1000000000
time taken 120.050278 by 3 threads
counter value is :1000000000
time taken 200.868382 by 4 threads
counter value is :1000000000
time taken 290.209962 by 5 threads
counter value is :1000000000
time taken 402.708413 by 6 threads
counter value is :1000000000
time taken 529.704038 by 7 threads
counter value is :1000000000
time taken 650.245612 by 8 threads
counter value is :1000000000
time taken 780.568543 by 9 threads
counter value is :1000000000
time taken 903.345664 by 10 threads
counter value is :1000000000



A mutex lock can only be released by the thread that has acquired it, while semaphore can signal from any thread to a semaphore locked by some other thread.

A mutex lock that allows only 1 threads be in the critical section and a semaphore does the same as a mutex but allows n number of threads to enter the critical region.

A mutex lock is acquired by a task and therefore must also be released by the same task. This helps to fix problems like accidental release, recursive deadlocks and priority inversions that are witnessed in binary semaphores.

Though semaphores are faster in terms of performance but are not as reliable as mutex locks as the data is shared and can be corrupted easily.

In my case, semaphore are faster than mutex locks

mutex are more efficient than semaphore but are slower than them, so there is a trade off between them basically.