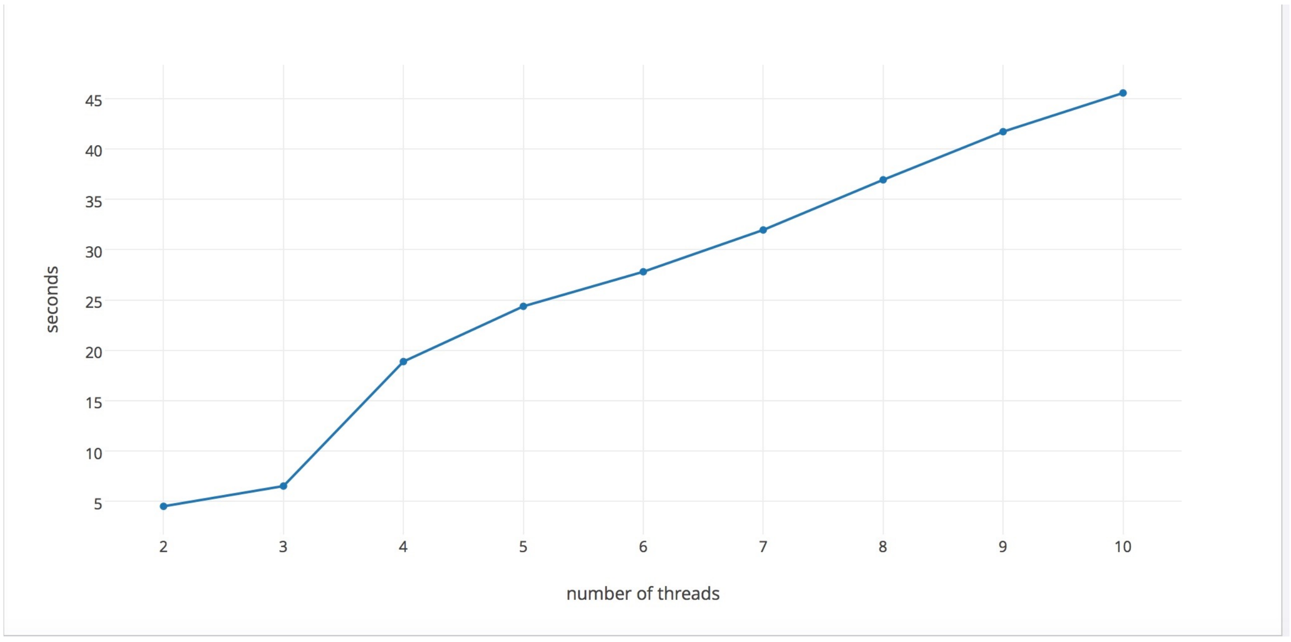
14105_pdf1

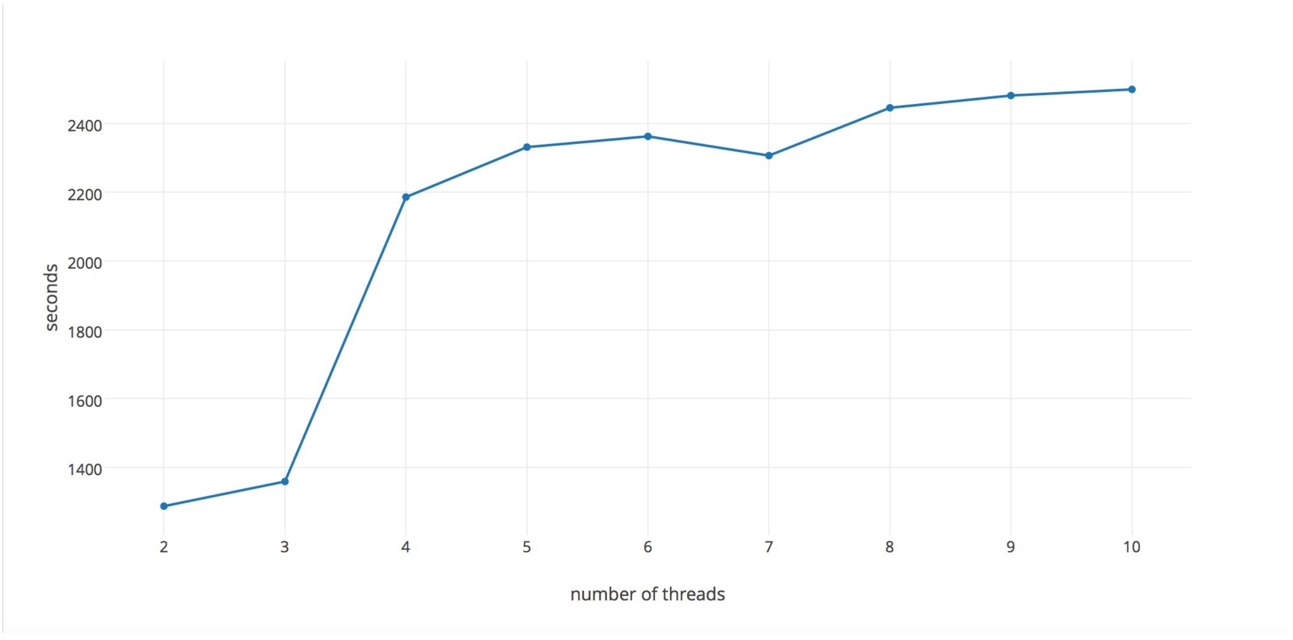
Terminal data 14105_mtp_c1.c

```
time taken 4.510094 by 2 threads
counter value is :1000000000
time taken 6.520679 by 3 threads
counter value is :1000000000
time taken 18.880633 by 4 threads
counter value is :1000000000
time taken 24.369433 by 5 threads
counter value is :1000000000
time taken 27.801459 by 6 threads
counter value is :1000000000
time taken 31.952530 by 7 threads
counter value is :1000000001
time taken 36.932462 by 8 threads
counter value is :1000000000
time taken 41.712978 by 9 threads
counter value is :1000000005
time taken 45.555567 by 10 threads
counter value is :1000000000
```



Terminal data for 14105_mtp_c2.c

```
time taken 1287.323887 by 2 threads
counter value is :1000000000
time taken 1358.906661 by 3 threads
counter value is :1000000000
time taken 2185.798551 by 4 threads
counter value is :1000000000
time taken 2330.960687 by 5 threads
counter value is :1000000000
time taken 2362.239612 by 6 threads
counter value is :1000000000
time taken 2306.337459 by 7 threads
counter value is :1000000000
time taken 2445.263935 by 8 threads
counter value is :1000000000
time taken 2480.893435 by 9 threads
counter value is :1000000000
time taken 2498.741621 by 10 threads
counter value is :1000000000
```



As we can see the lock unsafe is much faster than the lock safe program but it doesn't guarantee the counter value to be exact 1 billion. The time to execute with n threads keeps on increasing as n reaches 10, this is happening due to fact that the code is being run on a dual core processor, the ideal number of threads that should run this program is n+1 i.e 3, where n is the number of cores, 2 threads in our case, can perform the CPU operations and 1 thread can handle I/O.

The large differences in the timings for a corresponding number of threads in non safe and safe code is due to the large overhead like context switching that is helping to share the data between different threads side by side and due to the data that is being shared among the threads.

The CPU and the I/O have there limits to, it might be that the processor is fast, it may be computing increment operation very fast but the I/O may not be fast enough so the CPU may be waiting for the I/O to complete. So, in this case I/O being the culprit is increasing our execution time.

The locks ensure that the critical section is only executed by one thread at a time, it ensures correctness always. Though the execution time increases due the overhead by the locks, but we have to make a choice, its trade-off between performance and correctness.

In terms of performance non safe counter is faster, but in terms of efficiency safe counter is much more reliable, the time taken is large due to overheads.

The time taken to execute to execute the threads increases as we increase the number of working threads.