Report

ES15BTECH11002

Design:

Unique ID is passed to each of the threads.

Localtime function is used to get the time.

usleep function is used to simulate the critical section and remainder section time , the random number is generated and moded with 100000, to keep the sleep time limited to 0.1 sec.

The design of the algorithms is explained below.

pthread-barr:

Inbuilt library functions pthread_barrier_wait, pthread_barrier_init are used to initialize and enforce the barrier point property.

New-barr:

Basically 2 turnstiles are used in the design, one for implementing barrier point property and the other to prevent any bias towards a thread looping and running ahead of other threads.

Turnstile2 prevents a thread from getting ahead of other threads by looping and making the count n again.

Turnstile1 implements the barrier property.

Difficulties:

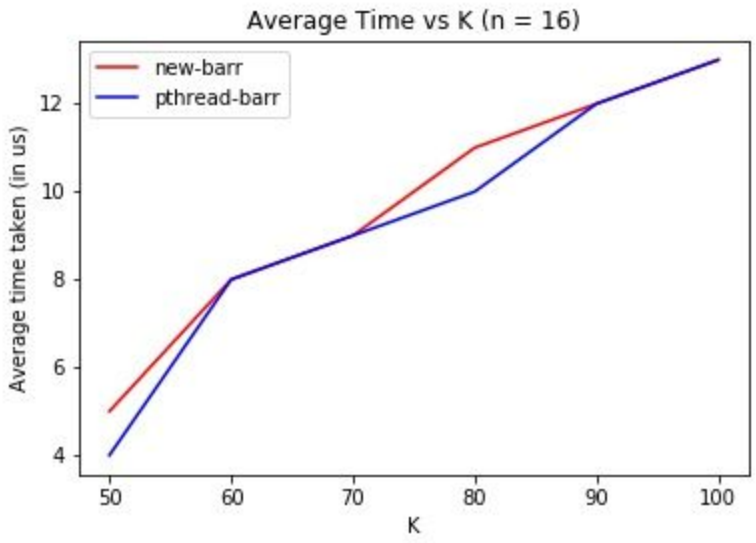
Turnstile2 had to be implemented to prevent a thread from looping and becoming ahead of others.

No difficulties were faced while implementing the built in algorithm.

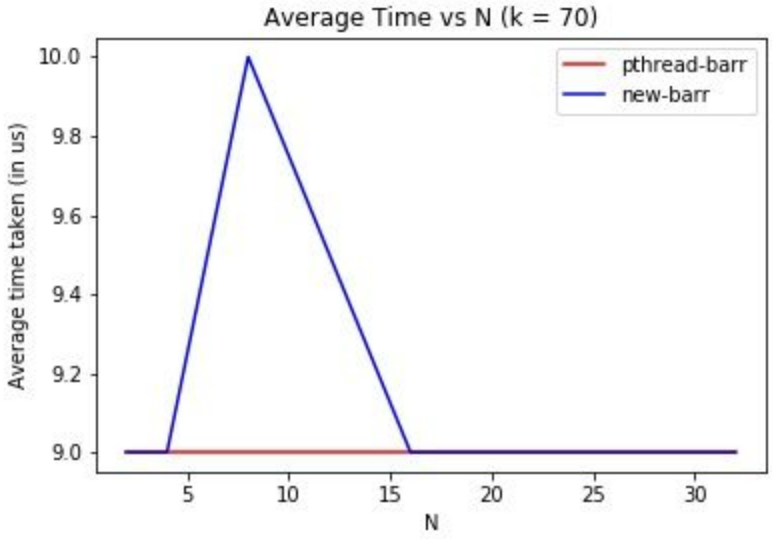
Graphs:

Note: Each point was run 3 times to compute the below graphs

Average time vs K



Average Time vs N:



Analysis:

The performance of both the algorithms are almost the same.

Average time increases almost linearly with the number of barrier invocations as can be seen from the graph.

Average time remains constant on increasing the number of threads with fixed number of invocations to the barrier.
This is expected as the threads run parallely, and the number of threads chosen here are less than the maximum number of threads on the cpu which the code was tested on.