COMP 8005
Assignment 1
January 26 / 2015

Geoff Dabu
A00817395

Disk Contents

+ 8005_A1Report.pdf
+ Processes
    - 8005A1_Processes.c
    + ComputationLogFiles_Processes
    - processes
    - ProcessTimes.txt
+ Threads
    - 8005A1_Threads.c
    + ComputationLogFiles_Threads
    - threads
    - ThreadTimes.txt

** ComputationLogFiles_Processes and ComputationLogFiles_Threads contain the log files that were created by each process/thread during execution. In these log files you can expect to see all the prime numbers up until the specified maximum (number of computations).

Program

This experiment consists of two separate programs, one implements threads and the other implements processes. Both programs work on two tasks; a mathematical computation, and I/O activity.

Mathematical Computation - Find all the prime numbers over a chosen range.
I/O activity - Write all the prime numbers to a newly created .txt file.

Both programs allow the user to specify the final number of a range starting at 0 on to which prime numbers are searched for, as well as the number of threads/processes the program will create to perform these tasks. These options are specified through command line arguments. Each thread/process will perform the exact same tasks.

The reason for choosing these tasks is for consistency amongst all the threads/processes. Each thread will analyze the exact same numbers, and require all the same resources, and the same goes for processes.

Execution

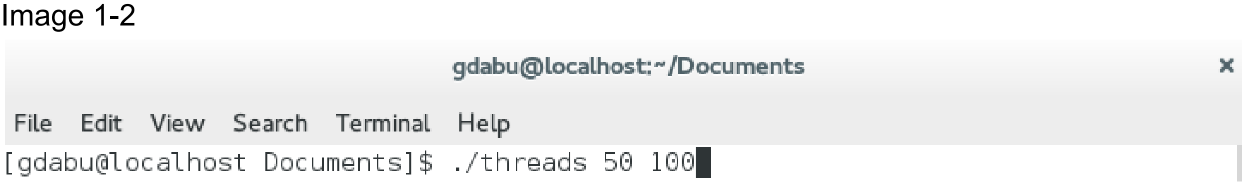
Shown below are some examples of how to execute both the multi-processed and multi-threaded programs.

Both programs take two arguments. The first is the number of processes/threads to be created and the second is the number of computations to calculated.

Image 1-1

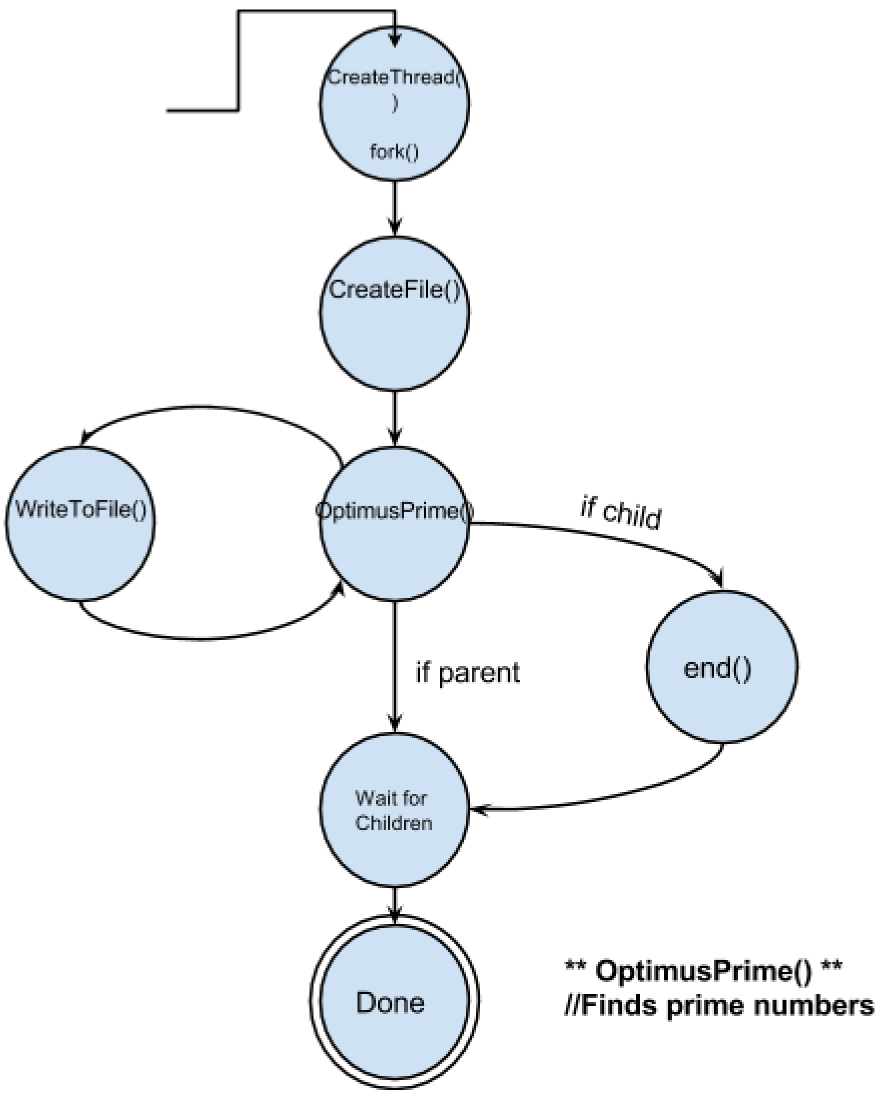
This command will run the thread application with 50 processes and 100000 computations

Image 1-2



This command will run the thread application with 50 threads and 100 computations

Design



CreateThread( )
fork()
CreateFile()
WriteToFile()
OptimusPrime()
if child
end()
if parent
Wait for Children
Done
** OptimusPrime() **
//Finds prime numbers

Testing

Test Sampling

Average completion times were taken from five test results for each combination of 5, 25, 50 processes/threads and 100, 100000, 1000000 computations. The log times can be found in the appendix at the end of this report.

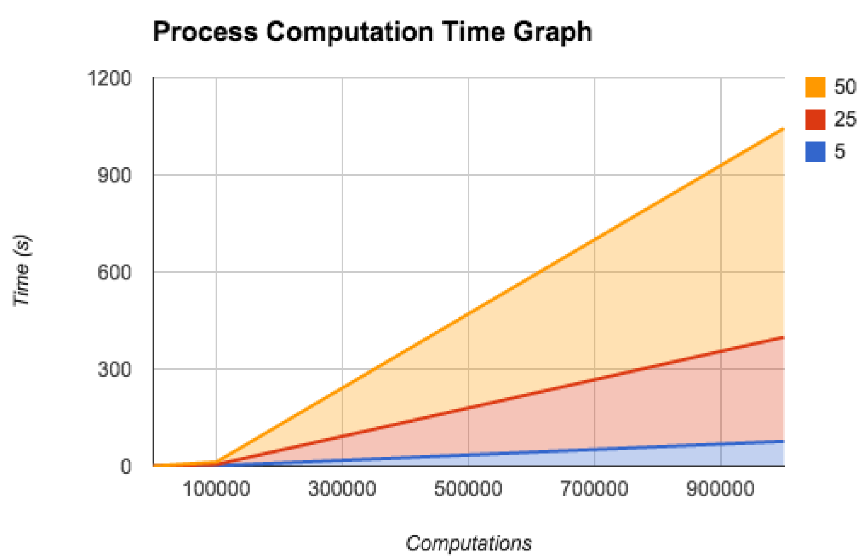
Result Set 1

Table 1-1 Average Completion Times for Processes

Processes	5 Processes	25 Processes	50 Processes
100 computations	0.00119 s	0.00519 s	0.01164 s
100000 computations	0.96270 s	3.86775 s	7.70174 s
1000000 computations	76.872 s	321.707 s	645.209 s

The table above tabulates the average completion times of 100, 100000, and 1000000 mathematical computations (calculating prime numbers) through the use of 5, 25 and 50 processes. The blue cells represent completion times that are shorter than the corresponding thread time.

Graph 1-1 Average Computation times across 5, 25 and 50 processes



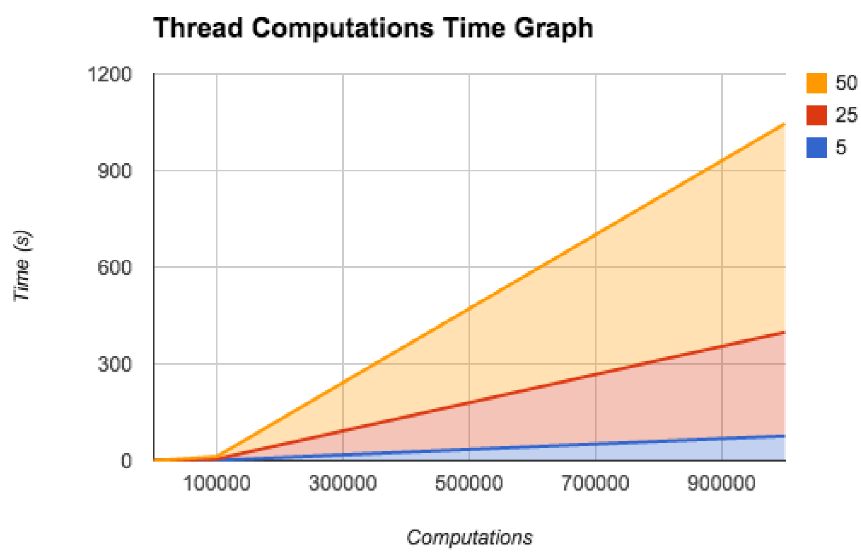
The Graph above illustrates the increase in time relative to the number of processes.

Table 1-2 Average Completion Times for Processes

Threads	5 Threads	25 Threads	50 Threads
100 computations	0.00045 s	0.00195 s	0.00418 s
100000 computations	0.96727 s	3.84726 s	7.71225 s
1000000 computations	76.892 s	321.681 s	646.969 s

The table above tabulates the average completion times of 100, 100000, and 1000000 mathematical computations (calculating prime numbers) through the use of 5, 25 and 50 threads. The green cells represent completion times that are shorter than the corresponding process completion times.

Graph 1-2 Average Computation times across 5, 25 and 50 threads



The Graph above illustrates the increase in time relative to the number of threads.

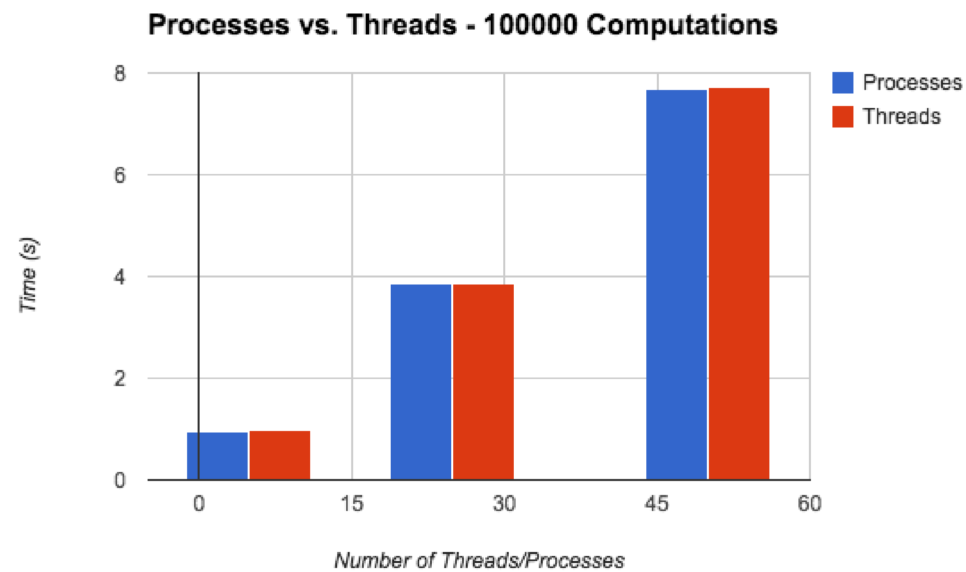
Result Set 1 Observations:

Tables 1-1, and 1-2 show that threads are noticeably better in performance when the number of computations (100) are minimal, but when there are more computations

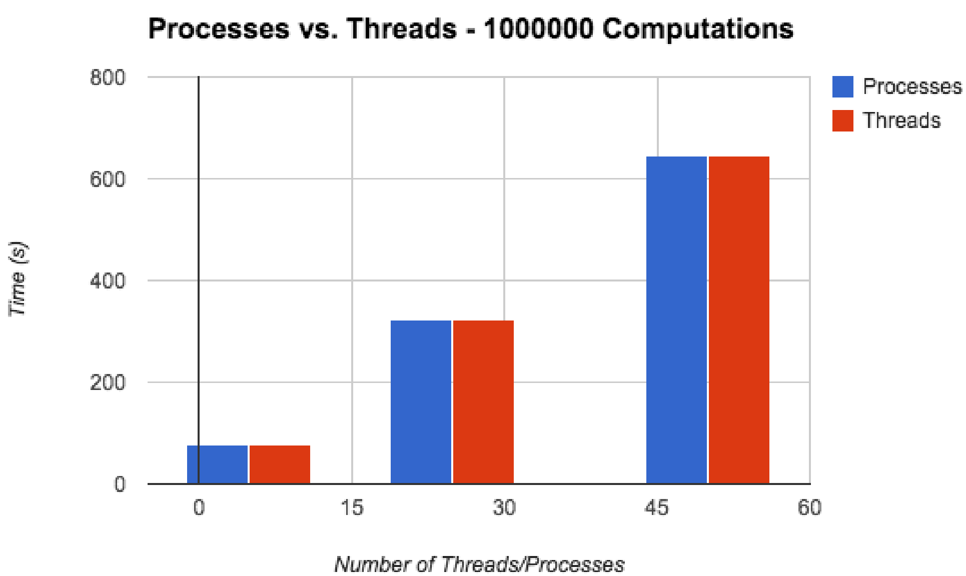
(100000 and 1000000) the performance difference between threads and processes is miniscule. Result Set 2 will better illustrate these findings.

Result Set 2

Graph 2-1

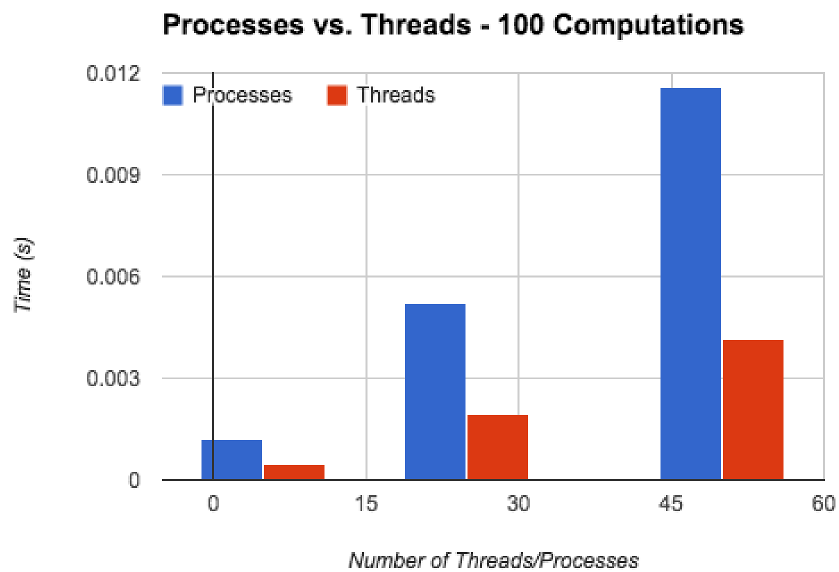


Graph 2-2



Notice how in both the 100000 and the 1000000 computation graphs that the completion times between processes and threads are near identical, no matter how many threads/processes were created.

Graph 2-3



Graph 2-3 is the only graph where there are noticeable differences in performance. As you can see threads have shorter completion times across the board.

Conclusion:

Based on the information that was drawn from this experiment, it can be inferred that threads have an advantage over processes when it comes to creation time. Both the multi-processed and multi-threaded applications were able to complete the 100 computations under a fraction of a millisecond. When these tasks are so small, the creation of either a process or a thread will have a more noticeable effect on the completion time - as shown in Graph 2-3.

To conclude this experiment, I would have to say that the performance of either multi-threaded or multi-processed application should not restrict you from using one over the other, unless your application is designed to continuously create new processes, which work on small tasks, if so, then I would suggest either using threads or implementing a process pool.

Appendix

Process Completion Times

TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.001112
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.001059
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.001249
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.001180
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.001338

TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:0.950049
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:0.950352
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:0.978743
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:0.982244
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:0.952114

TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:77.104984
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:76.761262
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:77.020695
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:76.638683
TOTALPROCESSES: 5|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:76.838745

TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.005831
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.005142
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.004871
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.005152
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.004952

TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:3.862602
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:3.846850
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:3.917430
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:3.847722
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:3.864269

TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:320.726162
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:322.835426
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:322.241342
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:321.912303
TOTALPROCESSES: 25|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:320.823402

TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.010277
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.017854
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.010770
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.010474
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100|| ELAPSEDTIME:0.008828

TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:7.686481
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:7.687796
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:7.744540
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:7.691245
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 100000|| ELAPSEDTIME:7.698888

TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:641.104157
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:647.060028
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:648.888945
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:645.563528
TOTALPROCESSES: 50|| TOTALCOMPUTATIONS: 1000000|| ELAPSEDTIME:643.432093

Thread Completion Times

TOTALTHREADS:5 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.000387
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.000454
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.000428
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.000460
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.000511

TOTALTHREADS:5 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:0.973646
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:0.972753
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:0.961375
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:0.953068
TOTALTHREADS:5 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:0.975485

TOTALTHREADS:5 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:76.656787
TOTALTHREADS:5 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:76.604704
TOTALTHREADS:5 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:77.425613
TOTALTHREADS:5 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:77.129405

TOTALTHREADS:5 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:76.643840

TOTALTHREADS:25 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.003148
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.001650
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.001769
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.001525
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.001673

TOTALTHREADS:25 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:3.842227
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:3.843188
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:3.840194
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:3.852644
TOTALTHREADS:25 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:3.858048

TOTALTHREADS:25 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:318.536757
TOTALTHREADS:25 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:322.994860
TOTALTHREADS:25 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:321.613892
TOTALTHREADS:25 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:320.575290
TOTALTHREADS:25 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:324.684543

TOTALTHREADS:50 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.008143
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.002958
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.003563
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.003088
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100 || ELAPSEDTIME:0.003165

TOTALTHREADS:50 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:7.703165
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:7.698181
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:7.761072
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:7.701417
TOTALTHREADS:50 || TOTALCOMPUTATIONS:100000 || ELAPSEDTIME:7.697393

TOTALTHREADS:50 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:640.060057
TOTALTHREADS:50 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:643.823816
TOTALTHREADS:50 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:650.872568
TOTALTHREADS:50 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:651.634573
TOTALTHREADS:50 || TOTALCOMPUTATIONS:1000000 || ELAPSEDTIME:648.452347