ASSIGNMENT 02

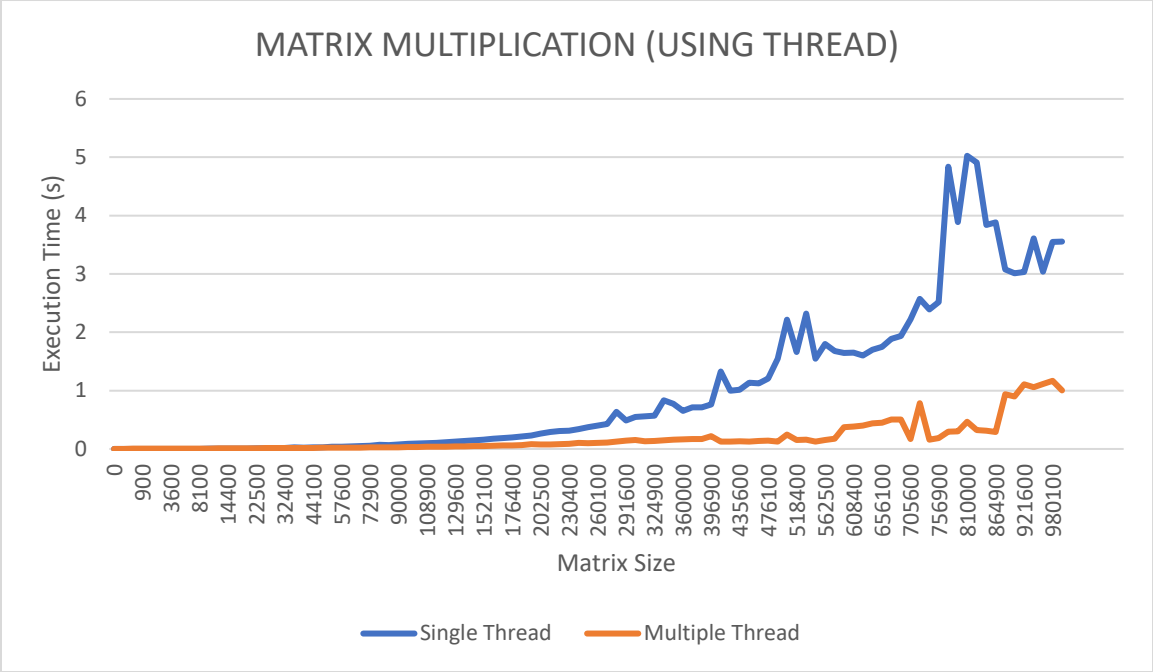
NAME : KARTHICK T.

REGISTRATION NO : EN93899

INDEX NO : 19/ENG/050

SUBMISSION DATE : 24/12/2021

Single threaded process contains execution of instructions in single sequence. Multiple threaded process allows execution of multiple parts of a program at the same time. Here multiplication was done using POSIX Pthreads and total number of threads involved in multiplication is equal to first matrices row count. So, these threads calculate rows of solution matrices.



The above graph shows matrix multiplications of matrix size ranging from 0 * 0 to 1000 * 1000 by 10 elements in each row and column, with elements range between 0.0 – 100.0, done on both single thread and multiple threads.

Table 01: Data for graph

No. of elements	Single thread (execution time in s)	Multi thread (execution time in s)
0	0.000000	0.000000
100	0.000000	0.002000
400	0.000000	0.003000
900	0.000000	0.004000
1600	0.001000	0.004000
2500	0.000000	0.004000
3600	0.001000	0.004000
4900	0.001000	0.005000
6400	0.002000	0.007000
8100	0.002000	0.007000
10000	0.003000	0.008000
12100	0.003000	0.009000
14400	0.005000	0.011000
16900	0.006000	0.010000
19600	0.007000	0.011000
22500	0.010000	0.012000
25600	0.011000	0.014000
28900	0.013000	0.015000
32400	0.016000	0.016000
36100	0.025000	0.015000
40000	0.021000	0.016000
44100	0.027000	0.018000
48400	0.030000	0.020000
52900	0.036000	0.022000
57600	0.039000	0.021000
62500	0.044000	0.023000
67600	0.050000	0.023000
72900	0.053000	0.028000
78400	0.070000	0.027000
84100	0.066000	0.030000
90000	0.077000	0.029000
96100	0.087000	0.034000
102400	0.094000	0.035000
108900	0.098000	0.036000
115600	0.106000	0.037000
122500	0.115000	0.041000
129600	0.127000	0.042000
136900	0.136000	0.046000

144400	0.148000	0.048000
152100	0.160000	0.051000
160000	0.173000	0.055000
168100	0.188000	0.058000
176400	0.200000	0.063000
184900	0.215000	0.065000
193600	0.231000	0.085000
202500	0.262000	0.076000
211600	0.292000	0.079000
220900	0.307000	0.082000
230400	0.311000	0.086000
240100	0.338000	0.105000
250000	0.375000	0.100000
260100	0.398000	0.105000
270400	0.429000	0.108000
280900	0.634000	0.126000
291600	0.488000	0.143000
302500	0.549000	0.155000
313600	0.557000	0.131000
324900	0.573000	0.137000
336400	0.834000	0.150000
348100	0.771000	0.158000
360000	0.652000	0.163000
37210	0.713000	0.170000
384400	0.712000	0.169000
396900	0.765000	0.218000
409600	1.325000	0.126000
422500	0.998000	0.126000
435600	1.014000	0.132000
448900	1.135000	0.125000
462400	1.124000	0.139000
476100	1.209000	0.144000
490000	1.547000	0.125000
504100	2.218000	0.248000
518400	1.660000	0.155000
532900	2.319000	0.157000
547600	1.545000	0.127000
562500	1.799000	0.152000
577600	1.676000	0.177000
592900	1.644000	0.373000
608400	1.652000	0.383000
624100	1.600000	0.398000

640000	1.700000	0.438000
656100	1.750000	0.451000
672400	1.888000	0.507000
688900	1.935000	0.502000
705600	2.223000	0.172000
722500	2.573000	0.786000
739600	2.394000	0.161000
756900	2.520000	0.188000
774400	4.839000	0.296000
792100	3.891000	0.299000
810000	5.025000	0.467000
828100	4.912000	0.325000
846400	3.839000	0.311000
864900	3.883000	0.289000
883600	3.077000	0.940000
902500	3.011000	0.900000
921600	3.033000	1.106000
940900	3.609000	1.057000
960400	3.040000	1.111000
980100	3.548000	1.168000
1000000	3.554000	1.005000

We can see that multi-threaded multiplication is much efficient when number of elements gets large. But for small number of elements single thread is efficient as multi-threaded program needs to do extra works for 'multi-threading' (like creation of threads, initialization, joining, etc..), which will be negotiable if number of actual work increase (in this scenario, for large no of elements).