Analysis on Single Thread and Multi-Thread Matrices Multiplication using pthreads

In this, matrix multiplication was done using single thread and multiple thread method and the matrix may have decimal numbers or integer numbers. And the matrices were generated using random numbers or two text file inputs.

Single threaded processes contain the execution of instructions in a single sequence (one command is processed at a time).

In multiple threaded processes allow the execution of multiple parts of a program at the same time. In this part, multiplication was done using POSIX Pthreads and multiplication threads are equal to first matrices row count.

Below graph is drawn using the random numbers between 0-100 (both decimal and integer) for both of the matrices in single thread and multiple thread process. Size of the matrix was calculated by the number of elements in the output matrix (10*10, 20*20, 30*30). In the analysis matrix size was increased by 10 elements in each row and column for 0*0 matrix to a 1000*1000 matrix.

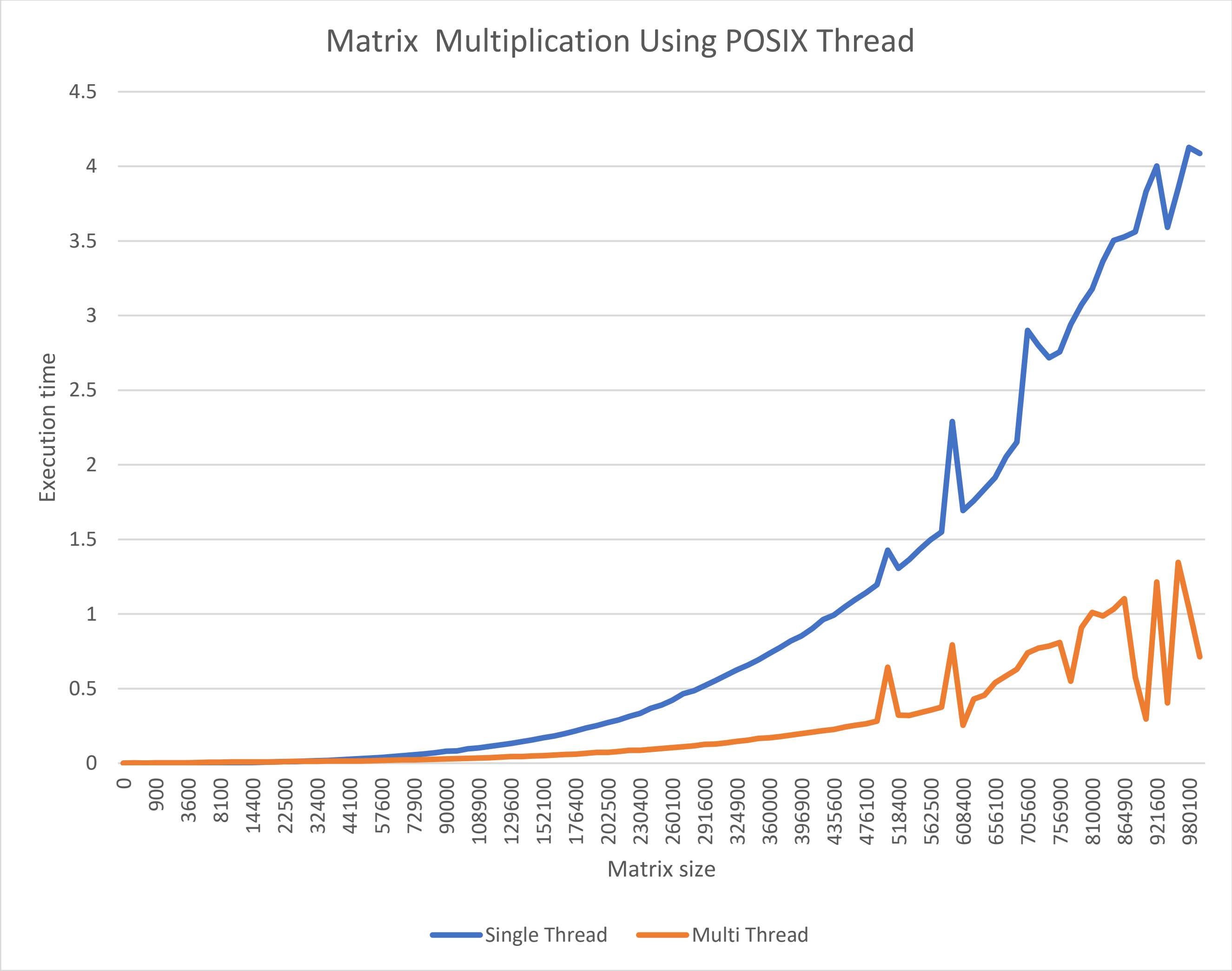


Figure 1 : Analysis on Single Thread and Multi-Thread Matrices multiplication

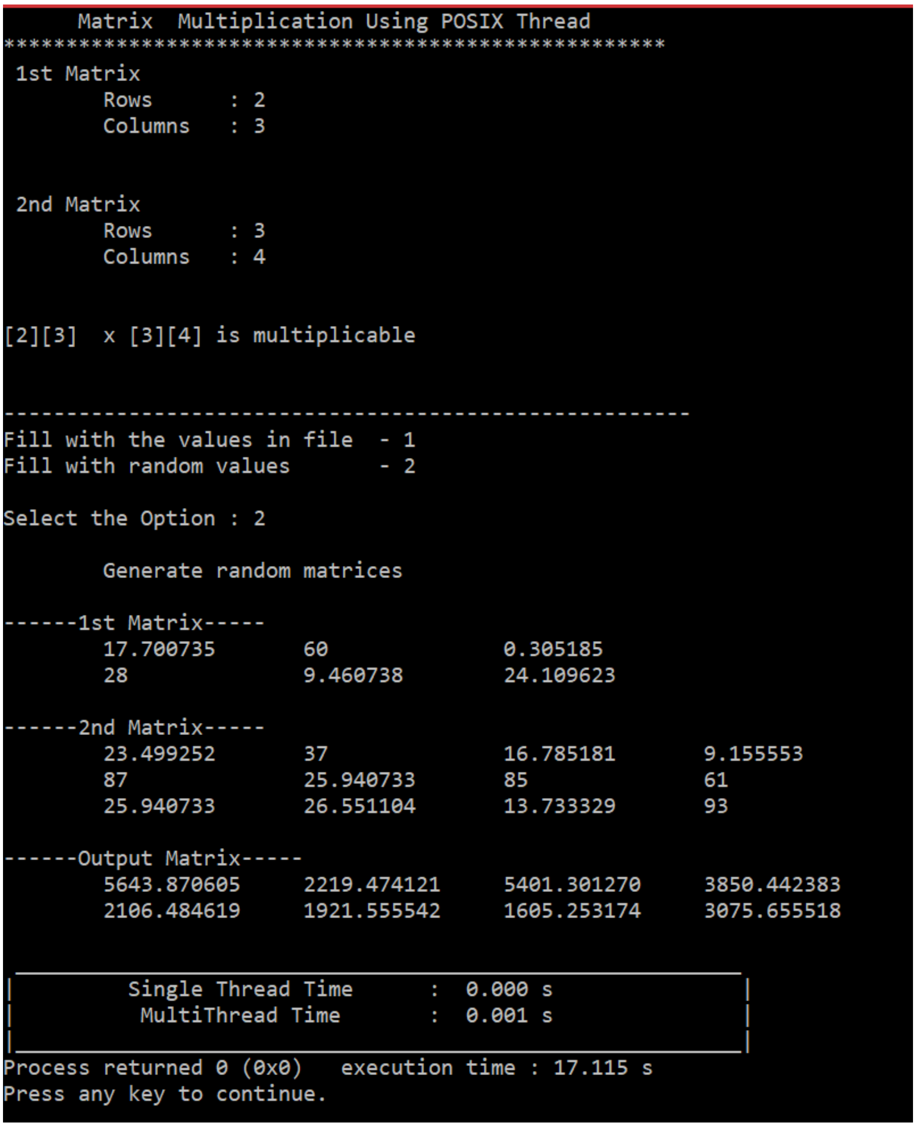
Table 1 : Execution times for multiplying matrices using single thread and multi-threading

Elements	Single Thread	Multi Thread
0	0	0
100	0	0.002
400	0	0.001
900	0.001	0.002
1600	0	0.002
2500	0.001	0.003
3600	0.001	0.003
4900	0.001	0.005
6400	0.001	0.006
8100	0.002	0.006
10000	0.002	0.008
12100	0.004	0.008
14400	0.005	0.008
16900	0.006	0.009
19600	0.009	0.009
22500	0.01	0.011
25600	0.011	0.012
28900	0.014	0.013
32400	0.016	0.013
36100	0.019	0.015
40000	0.022	0.015
44100	0.026	0.015
48400	0.03	0.015
52900	0.034	0.017
57600	0.039	0.019
62500	0.044	0.02
67600	0.05	0.022
72900	0.056	0.023
78400	0.062	0.024
84100	0.07	0.026
90000	0.08	0.028
96100	0.083	0.031
102400	0.096	0.032
108900	0.102	0.035
115600	0.112	0.037
122500	0.123	0.04
129600	0.133	0.044
136900	0.144	0.045
144400	0.156	0.048
152100	0.17	0.051
160000	0.182	0.054
168100	0.199	0.058
176400	0.216	0.06
184900	0.237	0.067
193600	0.253	0.073
202500	0.272	0.073
211600	0.291	0.078
220900	0.315	0.087
230400	0.335	0.087
240100	0.368	0.093
250000	0.39	0.098
260100	0.421	0.104
270400	0.465	0.111
280900	0.486	0.117
291600	0.519	0.127
302500	0.553	0.129
313600	0.589	0.136
324900	0.626	0.147
336400	0.657	0.154
348100	0.694	0.166
360000	0.735	0.171
372100	0.775	0.178
384400	0.82	0.188
396900	0.853	0.198
409600	0.902	0.209
422500	0.962	0.218
435600	0.993	0.227
448900	1.047	0.243
462400	1.097	0.255
476100	1.142	0.265
490000	1.197	0.283
504100	1.428	0.643
518400	1.306	0.322
532900	1.363	0.321
547600	1.433	0.339
562500	1.498	0.357
577600	1.549	0.376
592900	2.29	0.793
608400	1.693	0.255
624100	1.76	0.429
640000	1.837	0.455
656100	1.914	0.539
672400	2.054	0.585
688900	2.153	0.63
705600	2.901	0.739
722500	2.8	0.771
739600	2.718	0.786
756900	2.758	0.81
774400	2.94	0.549
792100	3.073	0.908
810000	3.178	1.01
828100	3.364	0.987
846400	3.503	1.033
864900	3.528	1.102
883600	3.562	0.575
902500	3.831	0.296
921600	4.002	1.214
940900	3.591	0.403
960400	3.852	1.346
980100	4.127	1.041
1000000	4.087	0.714

According to the above Figure 01 and the Table 1, at the beginning of the execution, the single thread execution is more efficient because a small workload can be handle easily using that and the multi-thread execution is very slow because, even if it is working in parallel, the workload is so small, so it doesn't balance out the overhead of creating, initializing and joining the threads.

But when the matrices size getting bigger (workload increases) and the multi-threading options gets better because more work can be performed in parallel and the overhead is very small when comparing to the calculation time.
In the given program when multiplication is done in row wise, solutions for multiple rows will be calculated parallelly and eventually all the solutions will be joined together and will preform a larger task in smaller time compared to single thread execution.

As a conclusion, multi-threaded matrices multiplication is better where large matrices involved-in and single thread is better where small matrices involved in.



```
      Matrix  Multiplication Using POSIX Thread
*****************************************************
 1st Matrix
        Rows      : 2
        Columns   : 3


 2nd Matrix
        Rows      : 3
        Columns   : 4


[2][3]  x [3][4] is multiplicable



-------------------------------------------------------
Fill with the values in file  - 1
Fill with random values       - 2

Select the Option : 2

        Generate random matrices

------1st Matrix-----
        17.700735       60              0.305185
        28              9.460738        24.109623

------2nd Matrix-----
        23.499252       37              16.785181       9.155553
        87              25.940733       85              61
        25.940733       26.551104       13.733329       93

------Output Matrix-----
        5643.870605     2219.474121     5401.301270     3850.442383
        2106.484619     1921.555542     1605.253174     3075.655518


 __________________________________________________________
|         Single Thread Time      :  0.000 s               |
|          MultiThread Time       :  0.001 s               |
|__________________________________________________________|
Process returned 0 (0x0)   execution time : 17.115 s
Press any key to continue.
```

Figure 2 : Output of the program