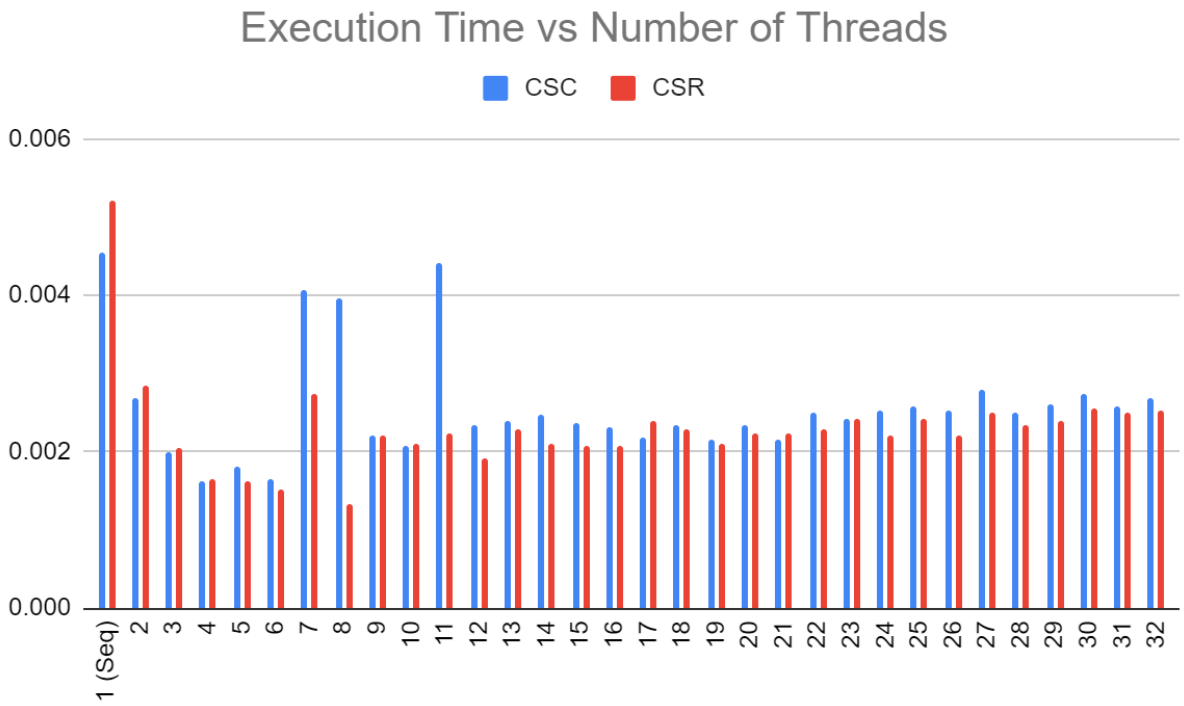
Huy Dinh Tran
htran197
ID: 862325308

PA1 Part 2 Report

Execution Time (in second ran with 8 iterations using matrix3.txt)

CSC	
Threads	ExecutionTime (sec)
1 (Seq)	0.004556
2	0.002687
3	0.001989
4	0.001628
5	0.001802
6	0.001651
7	0.004067
8	0.003969
9	0.002201
10	0.002078
11	0.004411
12	0.002331
13	0.002404
14	0.002466
15	0.002356
16	0.002305
17	0.002183
18	0.002338
19	0.002157
20	0.002345
21	0.002161
22	0.002496
23	0.002428
24	0.002537
25	0.002582
26	0.002519
27	0.00279
28	0.002512
29	0.002598
30	0.002732
31	0.002576
32	0.002687

CSR	
Threads	ExecutionTime (sec)
1 (Seq)	0.005217
2	0.002839
3	0.002047
4	0.001655
5	0.001616
6	0.001512
7	0.002741
8	0.001328
9	0.002202
10	0.002091
11	0.002233
12	0.00191
13	0.002291
14	0.002099
15	0.002079
16	0.002087
17	0.002387
18	0.002281
19	0.002105
20	0.002236
21	0.002233
22	0.002294
23	0.002425
24	0.002209
25	0.002414
26	0.002197
27	0.002499
28	0.002333
29	0.002383
30	0.002544
31	0.002503
32	0.002538



Note: My machine has an Intel Core i7-6700K 4 Cores 8 Threads with 32GB DDR4-2133

From the result above, we can clearly see that parallel beats sequential computation up to 3.93X for CSR and 2.8X for CSC. I got the best performance using 4 threads for CSC and 8 threads for CSR. Higher numbers of threads performed worse because of the extra overhead and communication when we use more threads which decreases the performance in our cases.

In addition, CSR runs a bit faster than CSC in most cases. One reason that I could think of is CSR improves cache locality because the format indexing and pointer work with the cache better.

Also, for CSC, I got slightly different decimal rounding outputs between sequential and parallel (after 3rd decimal place). I think this is because of some internal OpenMP problems that cause these differences since CSR outputs are consistent and identical.