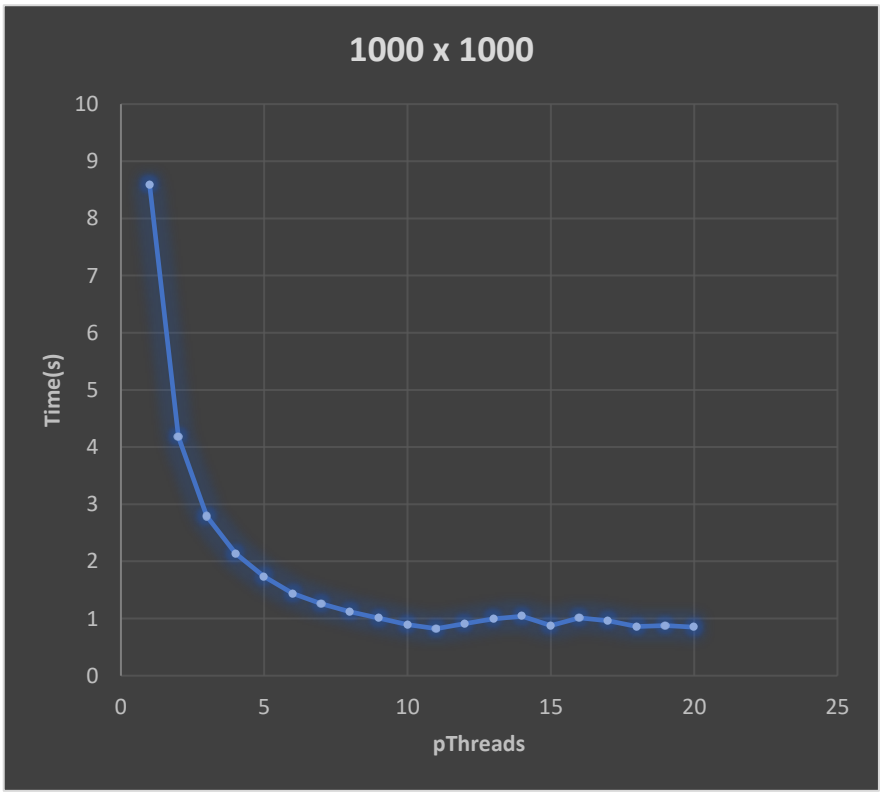
Chase Brown

OS 3453

Lab 1 Results

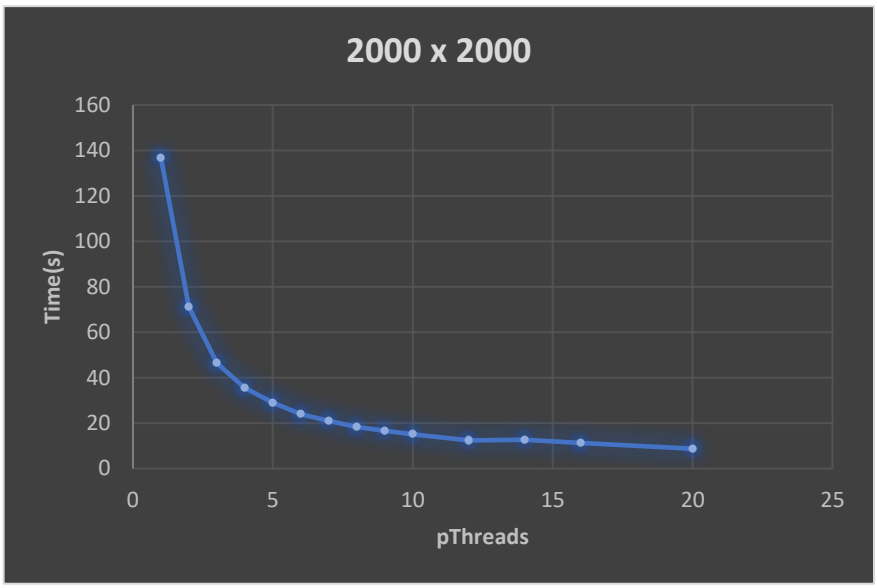
1000 by 1000 Matrix

pThreads	Time(s)
1	8.59403
2	4.17987
3	2.78639
4	2.1376
5	1.73392
6	1.43793
7	1.26012
8	1.12014
9	1.00476
10	0.893023
11	0.822993
12	0.908139
13	0.995816
14	1.0391
15	0.879382
16	1.01028
17	0.9593
18	0.860372
19	0.875373
20	0.849931



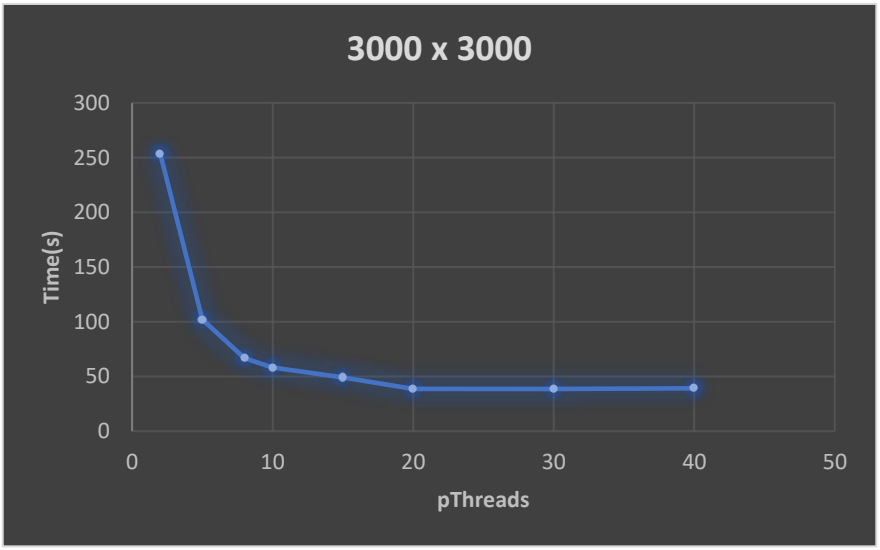
2000 by 2000 Matrix

1	136.784
2	71.1727
3	46.3846
4	35.5437
5	28.9983
6	23.9655
7	20.9887
8	18.3515
9	16.6629
10	15.1077
12	12.3448
14	12.6443
16	11.2665
20	8.73763

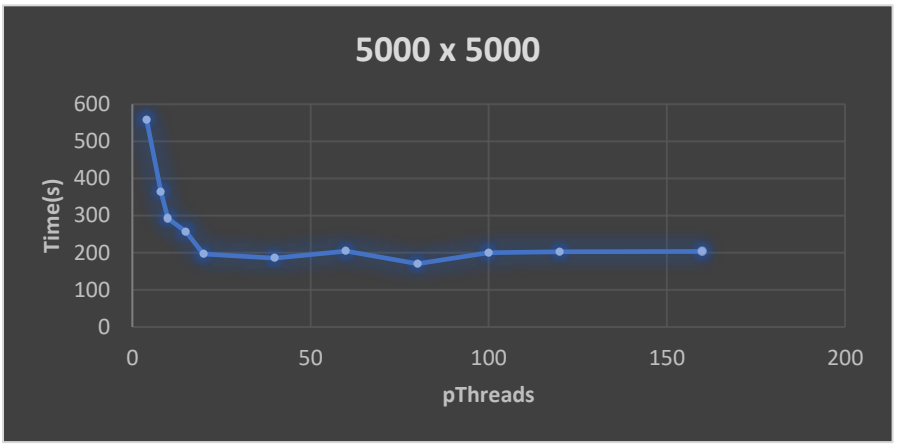


3000 by 3000 Matrix

2	253.581
5	101.994
8	66.6011
10	58.0246
15	49.0765
20	38.6199
30	38.6417
40	39.2872
50	36.1058
100	35.8379



5000 by 5000 Matrix	
4	558.499
8	364.499
10	292.516
15	257.151
20	196.531
40	185.23
60	204.508
80	170.379
100	200.357
120	203.002
160	203.109



The relationship between the graph didn't surprise me much that eventually the process would have to slow due to variables outside of the pthreads capabilities. The time vs pthreads shows a very steep slope when the threads are first starting to be increased. This is expected because splitting the task of work between two workers is going to give you twice the results for your time. This is shown to slow and eventually reach a horizontal asymptote depending on the size of matrices being calculated. The greater the number of calculations the higher this asymptote tends to be. This surprised me a little when comparing the 1000 matrix to 5000 matrix results. There are more than likely at least a few reasons this asymptote increases with the matrix sizes but I think that a very large reasoning is that even though 100 threads are sent out to do work they all still must come back and be joined in the end. This will begin to level off how fast your speed can get. Depending on the resources of the computers being used would be another large factor in the results. I was surprised to see such consistent graphs between the different size matrix sets.