Grayson Byrd
CPSC 8200 - Parallel Architecture
Project 2

Checking the Correctness of My Algorithm

To check the correctness of my algorithm, I used the following A and B matrices:

Matrix A	Matrix B
1 0 0 0 0 0	1 1 0 0 0 0
1 1 0 0 0 0	0 1 1 0 0 0
0 1 1 0 0 0	0 0 1 1 0 0
0 0 1 1 0 0	0 0 0 1 1 0
0 0 0 1 1 0	0 0 0 0 1 1
0 0 0 0 1 1	0 0 0 0 0 1

The resulting C matrix from multiplying matrix A with matrix B should be:

Matrix C

1 1 0 0 0 0
1 2 1 0 0 0
0 1 2 1 0 0
0 0 1 2 1 0
0 0 0 1 2 1
0 0 0 0 1 2

To test my algorithm, I used 9 processes on the 6x6 matrices. Each process was responsible for calculating a 2x2 block of the C matrix. To verify that my algorithm was producing correct results, I printed out the local values of the 2x2 C block after the process completed its matrix multiplication.

Here is the output from my code:

Rank: 6
-----C-----
0.000000 0.000000
0.000000 0.000000

Rank: 7
-----C-----
0.000000 0.000000
0.000000 0.000000

Rank: 3
-----C-----
0.000000 1.000000
0.000000 0.000000

Rank: 4
-----C-----
1.000000 2.000000
1.000000 2.000000

Rank: 5
-----C-----
0.000000 0.000000
0.000000 0.000000

Rank: 1
-----C-----
0.000000 0.000000
1.000000 0.000000

Rank: 2
-----C-----
0.000000 0.000000
0.000000 0.000000

Rank: 0
-----C-----
1.000000 1.000000
1.000000 2.000000

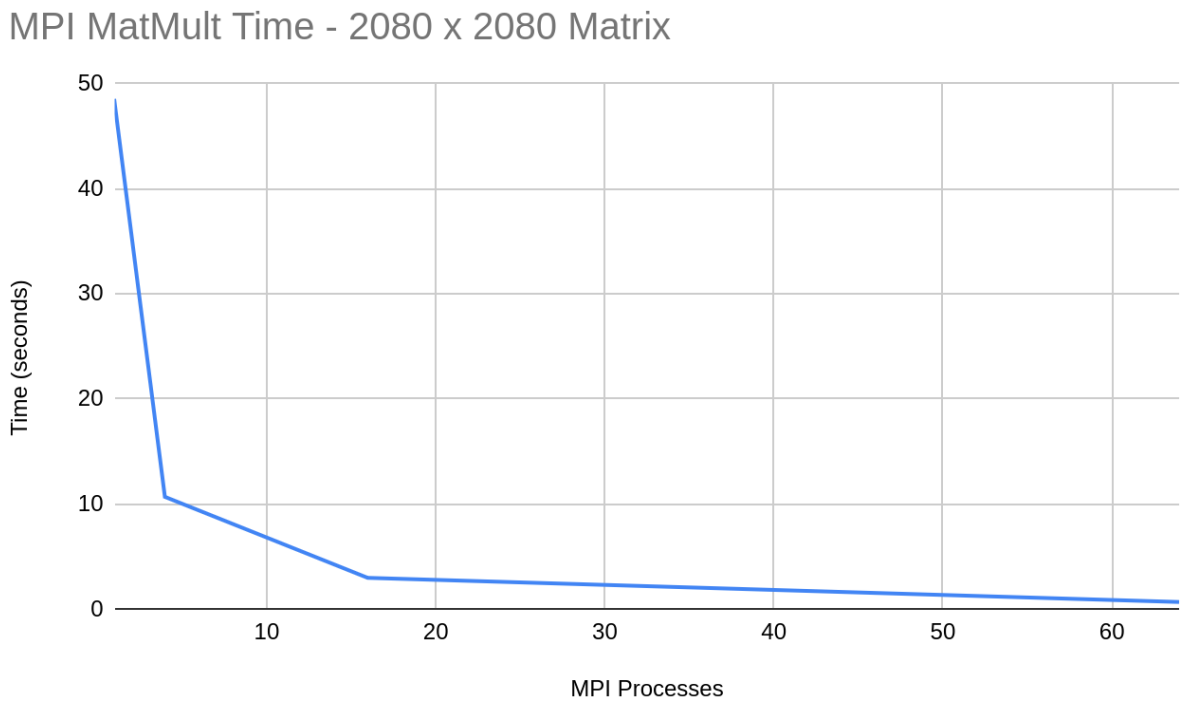
squareMatrixSideLength,4000,numMPICopies,9,walltime,0.160926

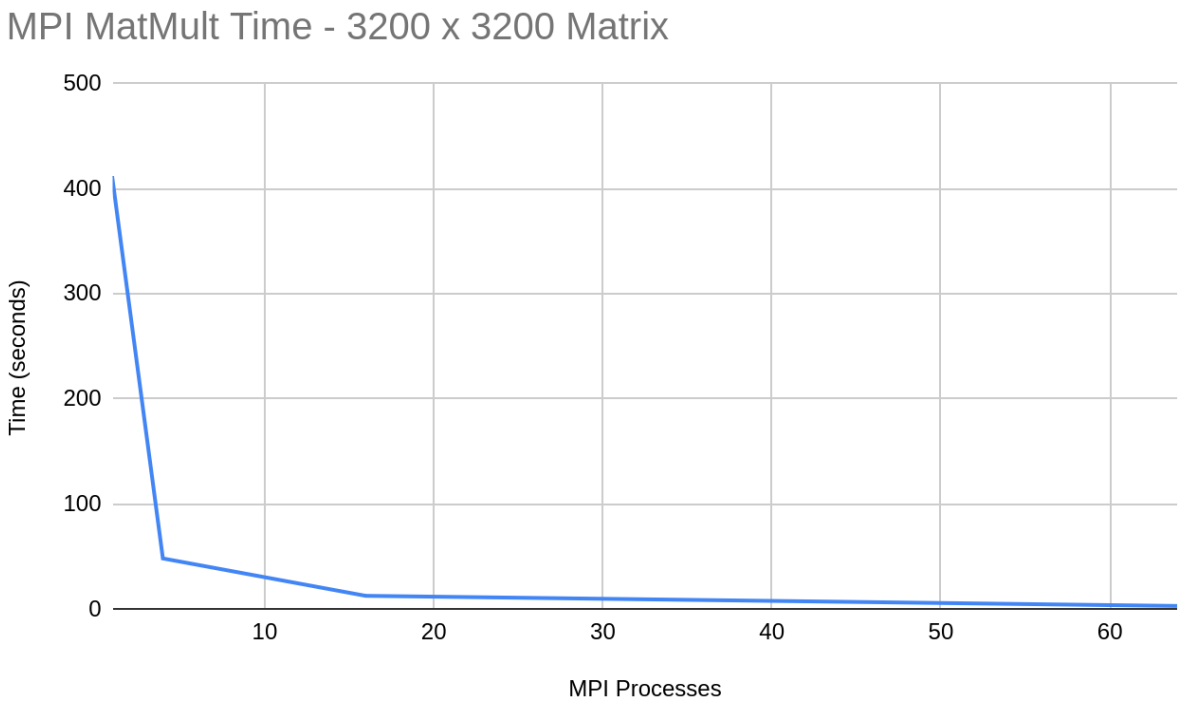
Rank: 8
-----C-----
2.000000 1.000000
1.000000 2.000000

As you can see, all of these values are correct assuming the C matrix is broken down in the following structure:

0 1 2
3 4 5
6 7 8

Finally, I tested the performance of my code with random matrices of various sizes with different numbers of MPI processes. The results can be seen below. I tested the performances on 4 nodes operating in parallel.





As you can see, there is a much larger speedup to processor ratio at the lower MPI processes. When going from 1 process to 4 processes, there is a much larger speedup than when going from 16 processes to 64 processes. This makes sense, as the more processes there are the smaller that data size for the matrix multiplication for each process. This means that since each process is calculating less, the memory access begins to become the bottleneck as the computation time decreases, so parallelizing the application with more processes will eventually stop producing better results.

References:

For this writeup, I just used this tutorial: https://mpitutorial.com/tutorials/