Kurt Andersen

CS 615

PA4 Matrix Multiplication

April 13, 2017

Introduction:

For this project we were assigned to create a sequential program to conduct matrix multiplication. The purpose of this project is to have an algorithm that has a terrible run time complexity when run sequentially, and eventually show the massive speed-up we can get when changing it to run in parallel.

Process:

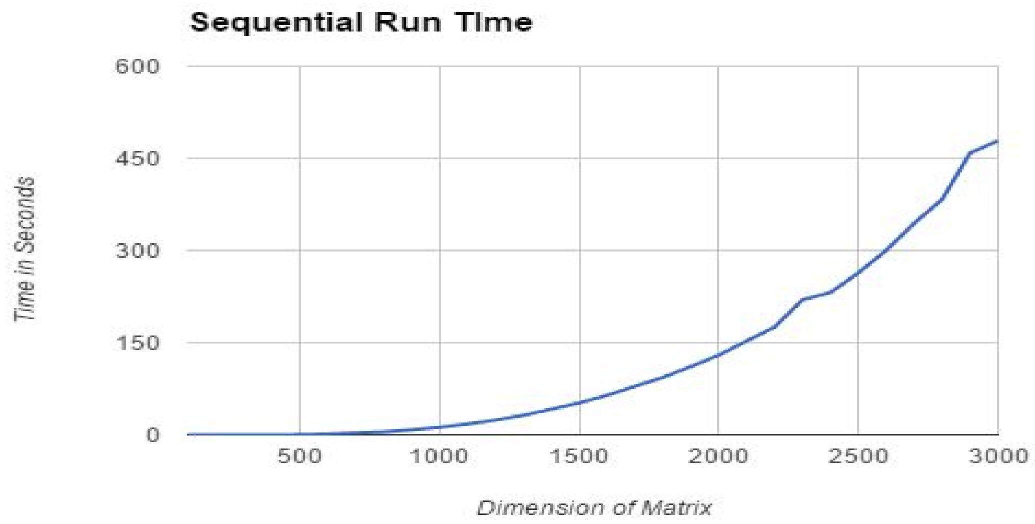
When I wrote the code for the sequential portion of this project I used google to find a simple algorithm. I then used the same idea from the online code and implemented it with the mpi library as well. I read values into my matrices in line with a constant seeded time. Then I started my timer once values were read into both matrices. From there I used a triple nested loop that found the appropriate solution matrix.

The only problem I ran into when doing the sequential portion was an error in my own code. When I was finding the sum of all the products for each cell in the matrix, I had forgot to make it find the sum of all the products. I was only taking the last multiplication for each cell. It was easily noticed and quickly fixed.

Results:

The data I gathered for the sequential portion of this project holds true to the N^3 time complexity of a standard matrix multiplication sequential algorithm. The time taken to complete each step increases exponentially. Overall this was a successful test to gather the proper data and set a good baseline for when I start getting data for the parallelized version.

Sequential Runtime Data			
Dimension	Time	Dimension	Time
100	0.002855	1600	64.1704
200	0.0205755	1700	79.0898
300	0.058846	1800	93.3977
400	0.151583	1900	111.124
500	0.360497	2000	129.475
600	1.45057	2100	152.826
700	3.32793	2200	175.332
800	4.94701	2300	219.879
900	8.1845	2400	231.449
1000	12.2958	2500	263.567
1100	17.618	2600	300.304
1200	23.9488	2700	344.359
1300	31.8389	2800	383.233
1400	41.74	2900	458.639
1500	51.9035	3000	478.483



The graph on the previous page shows the growth in time compared to the dimension of the matrix being calculated. This exponential growth in this graph strongly reflects that of an N^3 time complexity. As we increase our size, the time it takes between the calculations exponentially increase. This graph has almost no outliers and will construct a strong base for the future on this project.

Conclusion/Future Work:

Overall the sequential portion of this project was a 100% success. The data gathered shows the exponential growth perfectly, and there are no crazy outliers in the data. For the future on this project, I will use what I gathered here for a strong baseline for the parallelized version of this project.