Kurt Andersen

CS 615

PA4 Matrix Multiplication

April 27, 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:

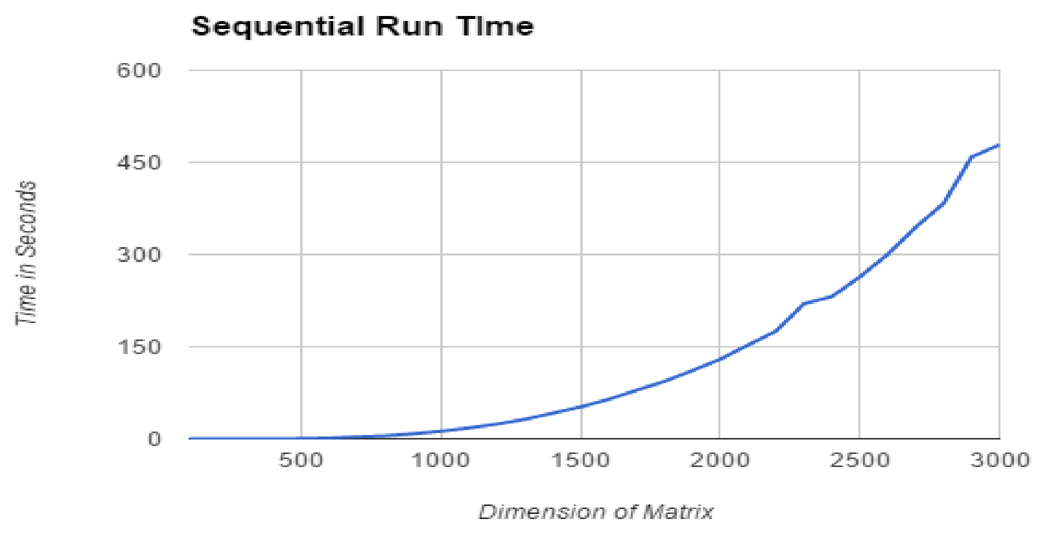
When I wrote the code for the sequential portion of this project I used google to find a simple algorithm. I then used the same idea from the online code and implemented it with the mpi library as well. I read values into my matrices in line with a constant seeded time. Then I started my timer once values 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.

Section 2: Updates to the file and changes as per peer feedback

During the peer review process I personally did not receive too much advice on what needed changing within my code. The main bit of feedback that I got from all of my reviewers is that I need to add functions. I added main functions that drastically reduced the amount of code in my main function, as well as reducing overall lines throughout the program. I know not having functions in during the first submission is a little grotesque to view the code, but the code was operable. I was mainly in a rush to get the code done, and did what needed to be done to complete it, and in my case it was not the most efficient structure. With the extra time to make changes based on feedback I was able to make my code a lot nicer.

Another small thing that I received was to do my calculations for my rotations differently. I did not change how I did the calculations for these because I feel as though the way I have my mesh set up, and the way my math works there is no other way. When doing column rotations I have a third case, only for the 0th column. This is because the top of the 0th column is mesh number 0. In my checks for the other rotations, it is possible that the value becomes 0. I added this extra case here so when data in the 0th column is 0, it will not wrap back to the bottom of the first column. I also did not reduce the amount of barriers that I have within my code because I want to keep my code running synchronously. I want it synchronously because I like knowing that all of the processes are at the same exact spot the entire time in running the program.