Ryan Devaney
5/2/2017
CS 415
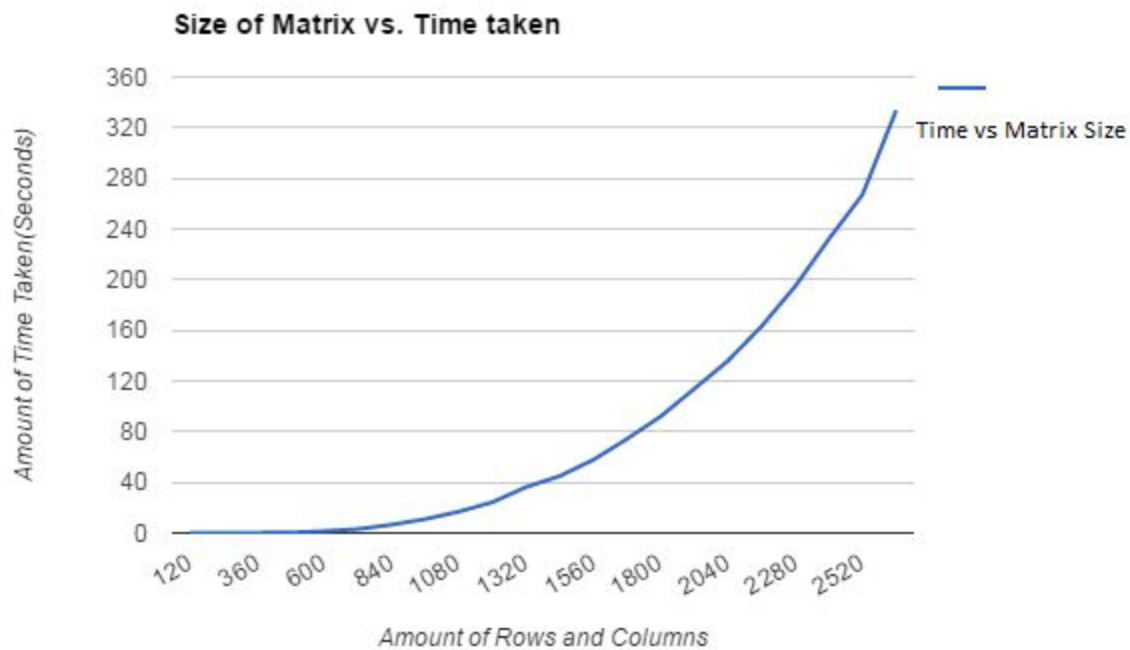Pa04 - Matrix Multiplication

Introduction

This is the report for project 4, matrix multiplication. This goal of this project was to implement matrix multiplication in both sequential and parallel and compare the time it takes to implement both.

Sequential

The first part of this assignment is to write and time sequential matrix multiplication. In order to do this, I read in the size of matrix from the command line. It is important to note that only square matrices of size 120 x k, where k is any integer more than 1, was used. This is because to compare it to parallel, 120 must be used because it can be divided by any number of perfect squares. Next I allocate memory for two matrices and a third one that will hold the resulting matrix. Next is input random values between 1 and 5 into the first two matrices and 0 into the third. Finally 3 for loops are used to do the actual calculation and put into the resulting matrix. The timing for the sequential matrix multiplication starts right before the for loops and ends right after the for loops. A total of 10 trials were run and then averaged to reach the times listed below.

Amount of Rows and Columns	Average time Taken(seconds)
120	0.006490851
240	0.02870466
360	0.1757363
480	0.3131466
600	1.421064
720	3.052699
840	6.524932
960	10.95011
1080	16.82777
1200	24.304
1320	36.26744
1440	44.92931
1560	57.818
1680	74.239
1800	92.01015
1920	113.837
2040	135.938
2160	163.1867
2280	194.7875
2400	232.1
2520	267.1188
2640	333.6505

Below is a graph showing the size of the matrix against the time taken. This graph shows an exponential growth as you would expect from matrix multiplication.

Size of Matrix vs. Time taken

Amount of Time Taken(Seconds) (y-axis): 0, 40, 80, 120, 160, 200, 240, 280, 320, 360

Amount of Rows and Columns (x-axis): 120, 360, 600, 840, 1080, 1320, 1560, 1800, 2040, 2280, 2520

Time vs Matrix Size

Parallel Improvement

Most of the comments I received for my code review was to fix small formating issues. In addition, I had to set a flag for printing out the matrices instead of just commenting the code out. Another thing that was changed after the peer review was being able to take input files for the matrices. I recently learned about sendReceiveReplace, so I plan to update my code using that before the timing aspect of the project is due.

Parallel Timings

Run Time

RunTime	4	9	16	25
120	0.004761695	0.004971	0.005719	0.27942
240	0.009226085	0.009943985	0.0101987	0.0084407
360	0.0261494	0.01915275	0.01213095	0.0193489
480	0.0488892	0.0427158	0.02464215	0.027308
600	0.107684	0.06467065	0.035969	0.041307
720	0.297458	0.1068438	0.0569532	0.05983
840	0.32489	0.148534	0.0918109	0.084687
960	0.453428	0.25845	0.133853	0.104983
1080	0.921445	0.499328	0.177032	0.163976
1200	1.25801	0.705017	0.229788	0.212825
1320	1.7354	0.625194	0.330892	0.294921
1440	2.4897578	0.804685	0.4186	0.335108
1560	4.5285	1.034	0.554306	0.516628
1680	8.93156	1.37049	0.65128	0.59138
1800	13.6461	2.0694448	1.18796	0.89854
1920	21.5219	2.1096221	1.34537	1.0685
2040	31.6298	2.7228	3.88122	1.98762
2160	36.6729	3.0917982	2.31981	2.7347
2280	44.6549	6.9152	3.4349	3.8104901
2400	52.0332	9.67009	5.1983	4.60487193
2520	58.94318	17.049	8.3018	6.40982
2640	73.205213	21.2285	11.19807	9.18709
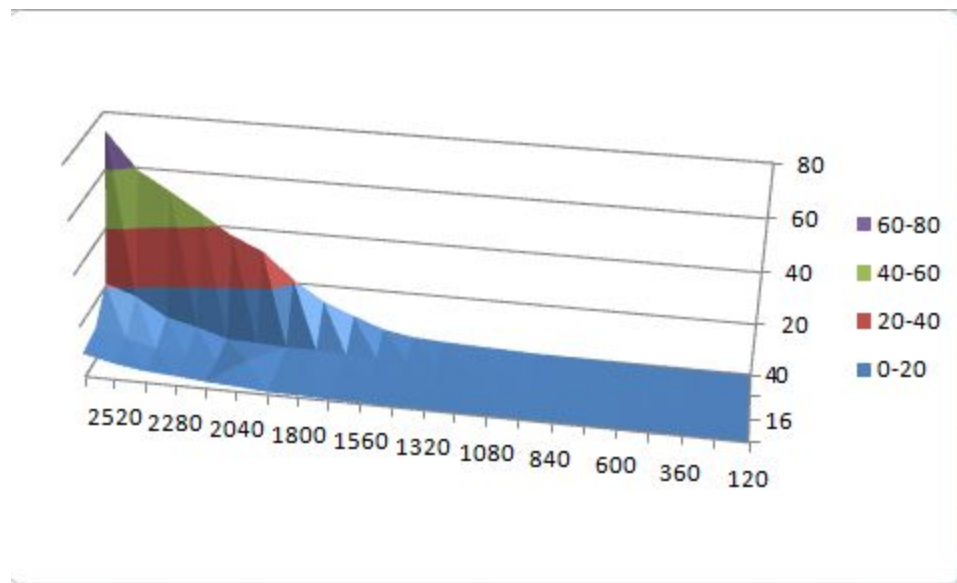
Speedup

speedup	4	9	16	25
120	1.363138756	1.305743512	1.134962581	0.02322972944
240	3.11125033	2.886635489	2.814541069	3.400744014
360	6.720471598	9.175512655	14.48660657	9.082495646
480	6.405230603	7.330931412	12.70776292	11.46721107
600	13.19661231	21.97386295	39.5080208	34.40249837
720	10.26262195	28.5716064	53.60013134	51.0228815
840	20.08351134	43.92887824	71.06925213	77.04762242
960	24.1496114	42.36838847	81.80698229	104.3036492
1080	18.26237052	33.70083392	95.05496181	102.6233717
1200	19.31940128	34.4729276	105.7670549	114.1971103
1320	20.89860551	58.00989773	109.6050675	122.9734064
1440	18.04565488	55.8346558	107.332322	134.0741194
1560	12.76758308	55.91682785	104.3070073	111.914182
1680	8.311985812	54.16967654	113.9893748	125.5351889
1800	6.742596786	44.4612729	77.45222903	102.3996149
1920	5.289356423	53.9608492	84.61389804	106.5390735
2040	4.297782471	49.92581166	35.02455414	68.39234864
2160	4.449789899	52.78051459	70.34485583	59.67261491
2280	4.36206329	28.16802117	56.70834668	51.11875241
2400	4.460613608	24.00184486	44.64921224	50.40313901
2520	4.531801644	15.66771072	32.17601002	41.67336992
2640	4.557742356	15.71710201	29.79535759	36.31732137
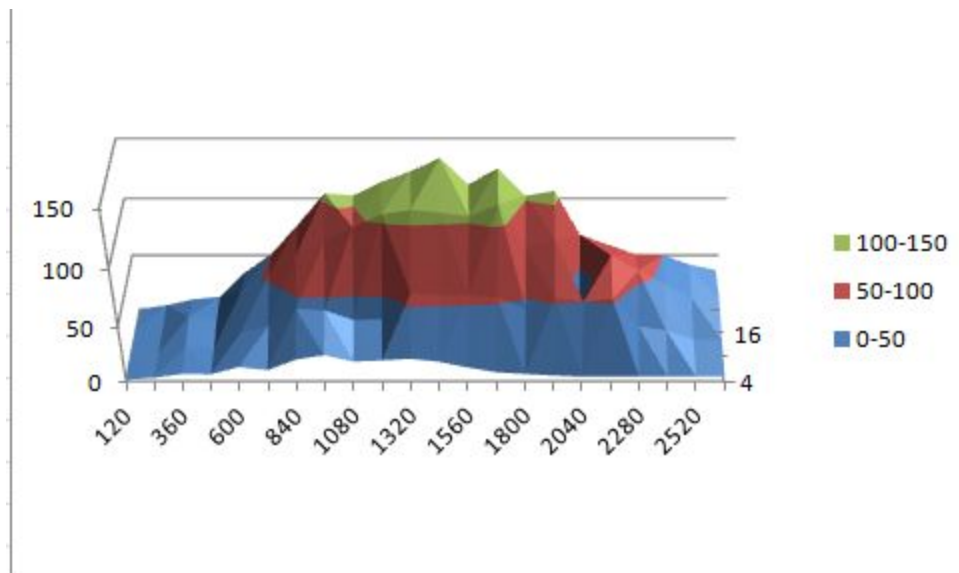
Efficiency

Efficiency	4	9	16	25
120	0.3407846891	0.1450826125	0.0709351613	0.00092918917776
240	0.7778125825	0.3207372765	0.1759088168	0.1360297606
360	1.680117899	1.019501406	0.9054129108	0.3632998258
480	1.601307651	0.8145479346	0.7942351824	0.4586884429
600	3.299153078	2.441540328	2.4692513	1.376099935
720	2.565655487	3.174622933	3.350008208	2.04091526
840	5.020877836	4.880986471	4.441828258	3.081904897
960	6.037402851	4.707598719	5.112936393	4.172145966
1080	4.565592629	3.744537102	5.940935113	4.104934869
1200	4.829850319	3.830325289	6.610440928	4.567884412
1320	5.224651377	6.445544192	6.85031672	4.918936258
1440	4.511413721	6.203850644	6.708270127	5.362964776
1560	3.191895771	6.212980873	6.519187958	4.476567279
1680	2.077996453	6.018852949	7.124335923	5.021407555
1800	1.685649196	4.940141433	4.840764314	4.095984597
1920	1.322339106	5.995649911	5.288368627	4.261562939
2040	1.074445618	5.547312407	2.189034633	2.735693946
2160	1.112447475	5.864501621	4.396553489	2.386904596
2280	1.090515822	3.12978013	3.544271667	2.044750097
2400	1.115153402	2.666871652	2.790575765	2.01612556
2520	1.132950411	1.740856746	2.011000626	1.666934797
2640	1.139435589	1.746344668	1.86220985	1.452692855