Assignment 5

Inderpreet Singh Chera - 160101035

Gaussian Elimination

Gaussian elimination, also known as row reduction, is an algorithm in linear algebra for solving a system of linear equations.
Two different parallel implementations of gaussian elimination were performed, one was normal parallel implementation using broadcast strategy and the other used pipelining.

Experiments were performed by 1, 2, 4, 8 and 16 processes.

System Settings

All tests were done on **Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz** processor. This computer is **dual core** where each core has **2 threads**. Also during each experiment it was insured that **no other applications** were running in the background, so that we don't have any biased readings.

Implementations

Let's say we have matrix $A_{n \times n}$ and we want to perform gaussian elimination on it, which is we have to convert it into an upper triangular matrix so that we can easily use it to solve linear equations.

Gaussian Elimination

Algorithm is to first for each row r, first divide all the elements in the row by the element in its diagonal position. Then, send this row to all other rows, say x (basically doing a broadcast) so that they subtract this row r multiplied by the element at the r^{th} column of the x^{th} row from x row.

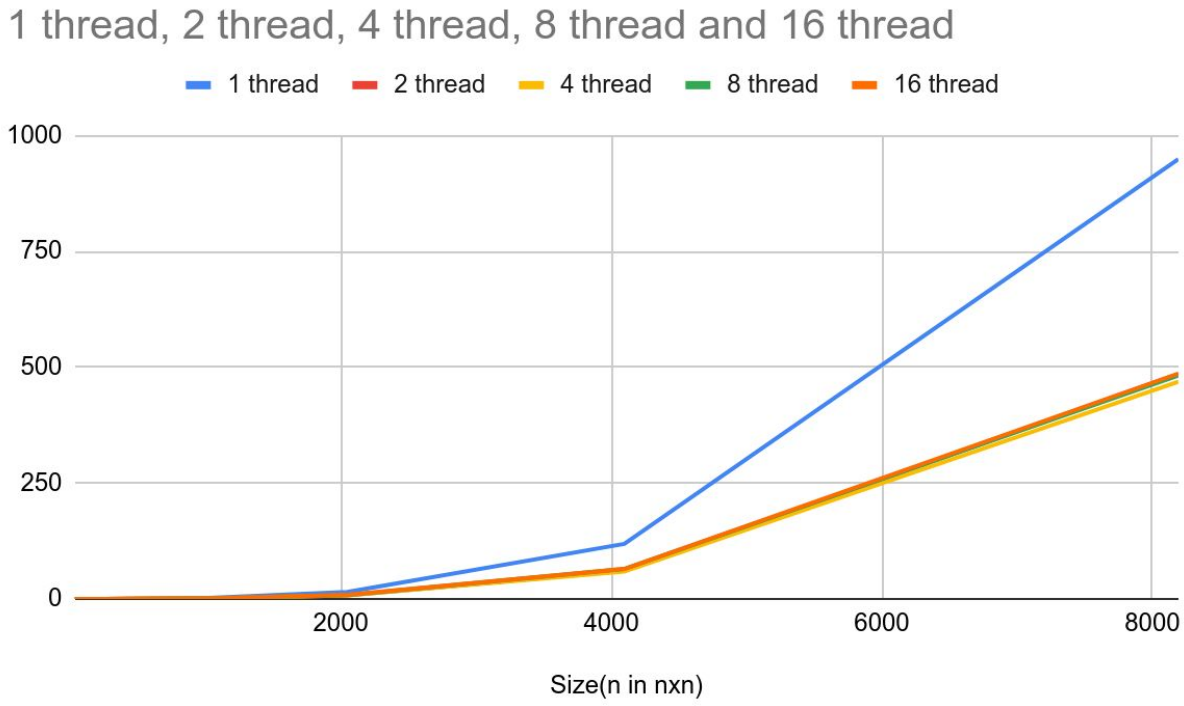
Pipelined Gaussian Elimination

Here instead of the row being broadcasted, it is only sent to the next row. When a row gets the row from which it has to subtract itself, it first sends the received row to the next row and then does the subtraction operation. Here, in my implementation, if a process gets the row which is to be subtracted, it subtracts it from all the rows that the process has. So, basically the row that has to be subtracted gets transferred maximum of number of processes times, rather than having it transferred number of rows time.

Observations

Gaussian Elimination on different number of processes:

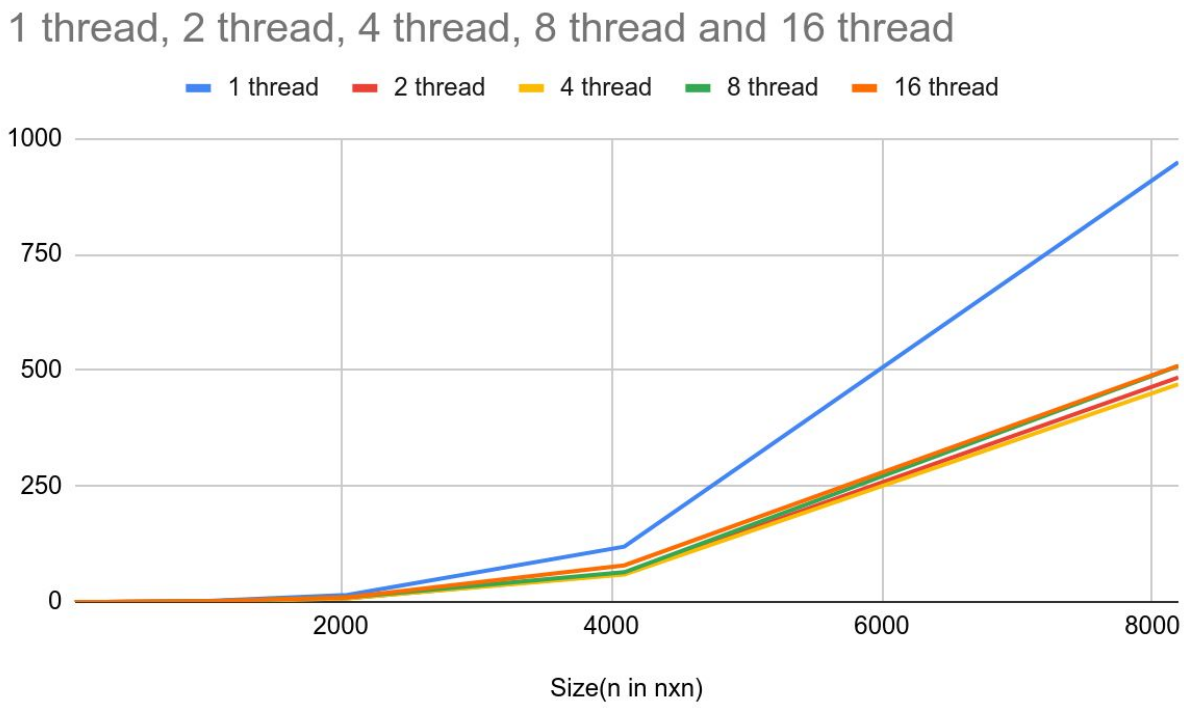
Size(n in nxn)	1 thread	2 thread	4 thread	8 thread	16 thread
32	0.000073	0.000139	0.000329	0.001415	0.002739
64	0.000883	0.000396	0.000596	0.002007	0.008992
128	0.005162	0.00549	0.002605	0.004983	0.014519
256	0.033885	0.01682	0.018473	0.024872	0.053533
512	0.236062	0.12194	0.120211	0.167232	0.195365
1024	1.892114	0.957433	0.932374	1.174951	1.125776
2048	15.033303	7.665704	7.446726	8.312522	8.50656
4096	118.849473	60.971784	58.97559	64.585363	64.577163
8192	949.328595	485.654603	468.675474	482.380602	486.457294



We observe that time taken by unit process is far more than taken by more number of processes. We also observe that time taken first reduces on increasing the number of processes and then increases after 4 processes are used.

Pipelined Gaussian Elimination on different number of processes:

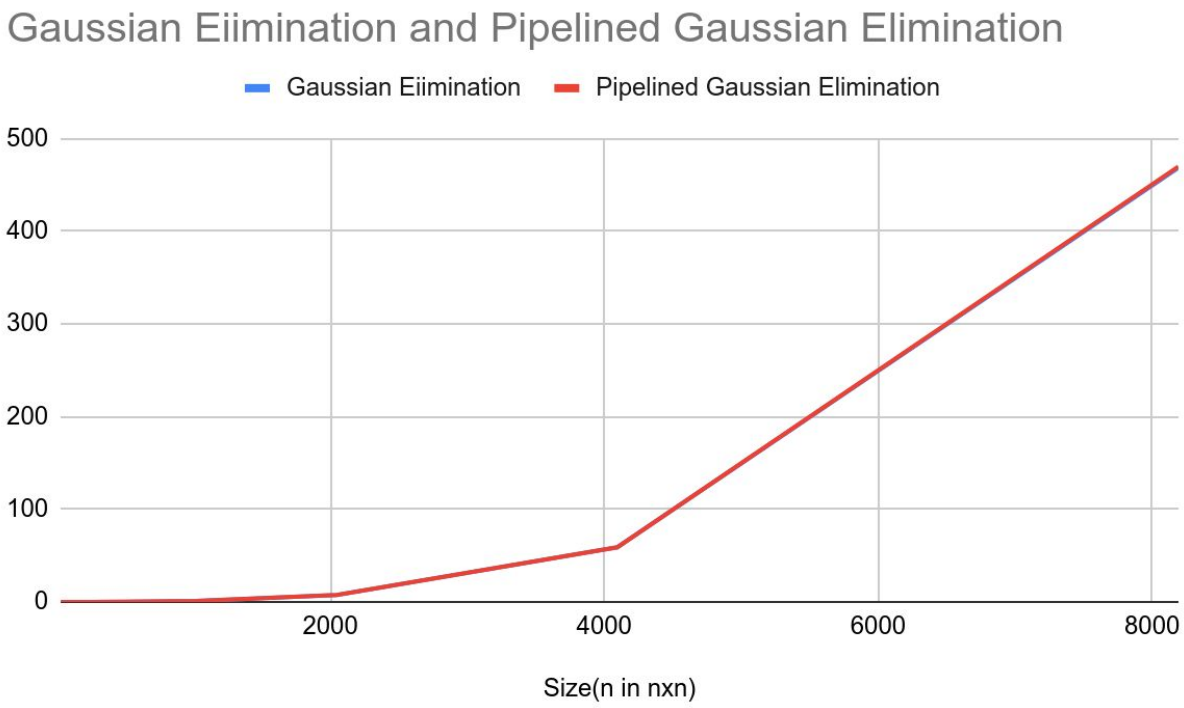
Size(n in nxn)	1 thread	2 thread	4 thread	8 thread	16 thread
32	0.000124	0.000146	0.000313	0.001083	0.003079
64	0.000697	0.000375	0.0006	0.001555	0.005886
128	0.00829	0.002782	0.002523	0.006873	0.008627
256	0.032445	0.015551	0.017095	0.04463	0.042318
512	0.236559	0.122648	0.119013	0.222355	0.24979
1024	1.868575	0.957981	0.934324	1.279926	1.459301
2048	15.007333	7.662732	7.41455	9.770436	10.013102
4096	119.316208	60.924691	58.996821	64.128273	78.922299
8192	949.238407	484.625476	470.018423	509.185117	509.979915



We observe that time taken by unit process is far more than taken by more number of processes. We also observe that time taken first reduces on increasing the number of processes and then increases after 4 processes are used.

Gaussian Elimination vs pipelined Gaussian Elimination using 4 threads:

Size(n in nxn)	Gaussian Eiimination	Pipelined Gaussian Elimination
32	0.000329	0.000313
64	0.000596	0.0006
128	0.002605	0.002523
256	0.018473	0.017095
512	0.120211	0.119013
1024	0.932374	0.934324
2048	7.446726	7.41455
4096	58.97559	58.996821
8192	468.675474	470.018423



We observe that both types of gaussian elimination takes almost the same time, although theoretically, the pipelined version of gaussian elimination should have taken less time than normal gaussian elimination.