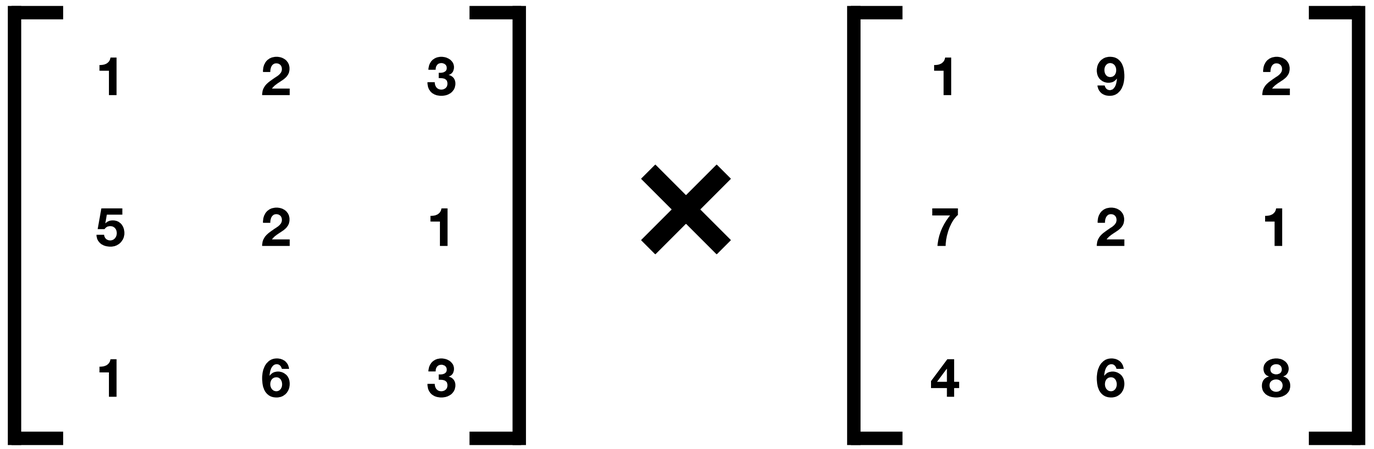
CSE 633 – PARALLEL ALGORITHMS

Matrix-Matrix Multiplication



Name: Parth Anand

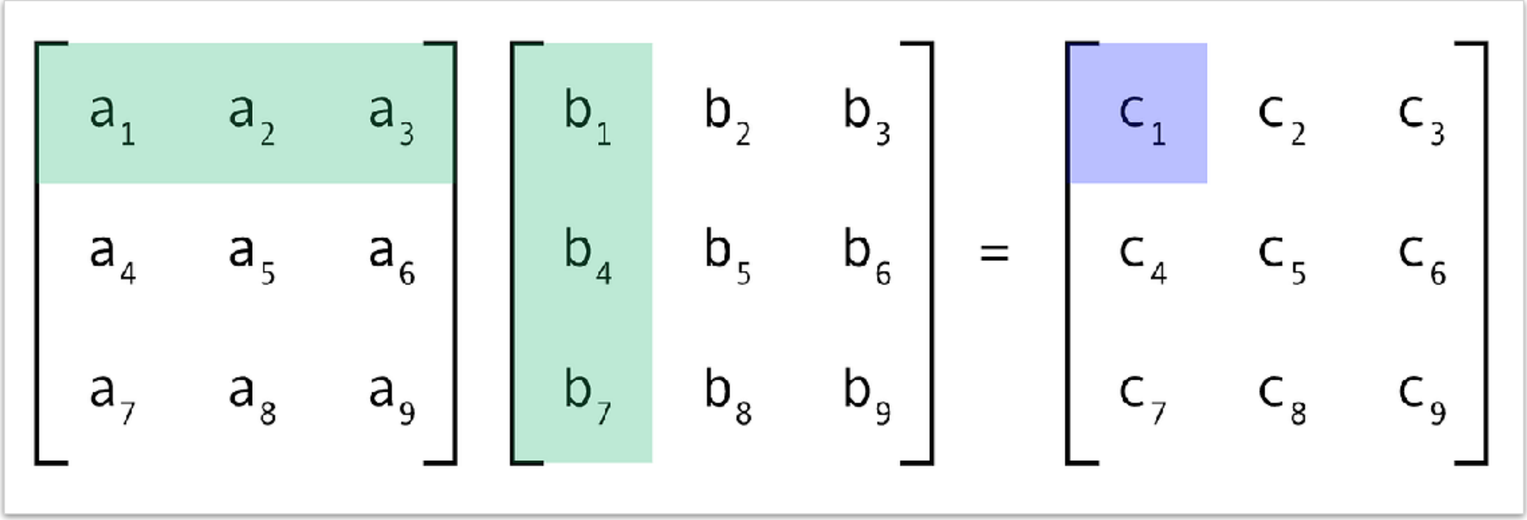
UBIT Name: parthana

Why is this an interesting problem?

(Real world use cases)

- **Computer Graphics**
- **Machine Learning**
- **Physics**

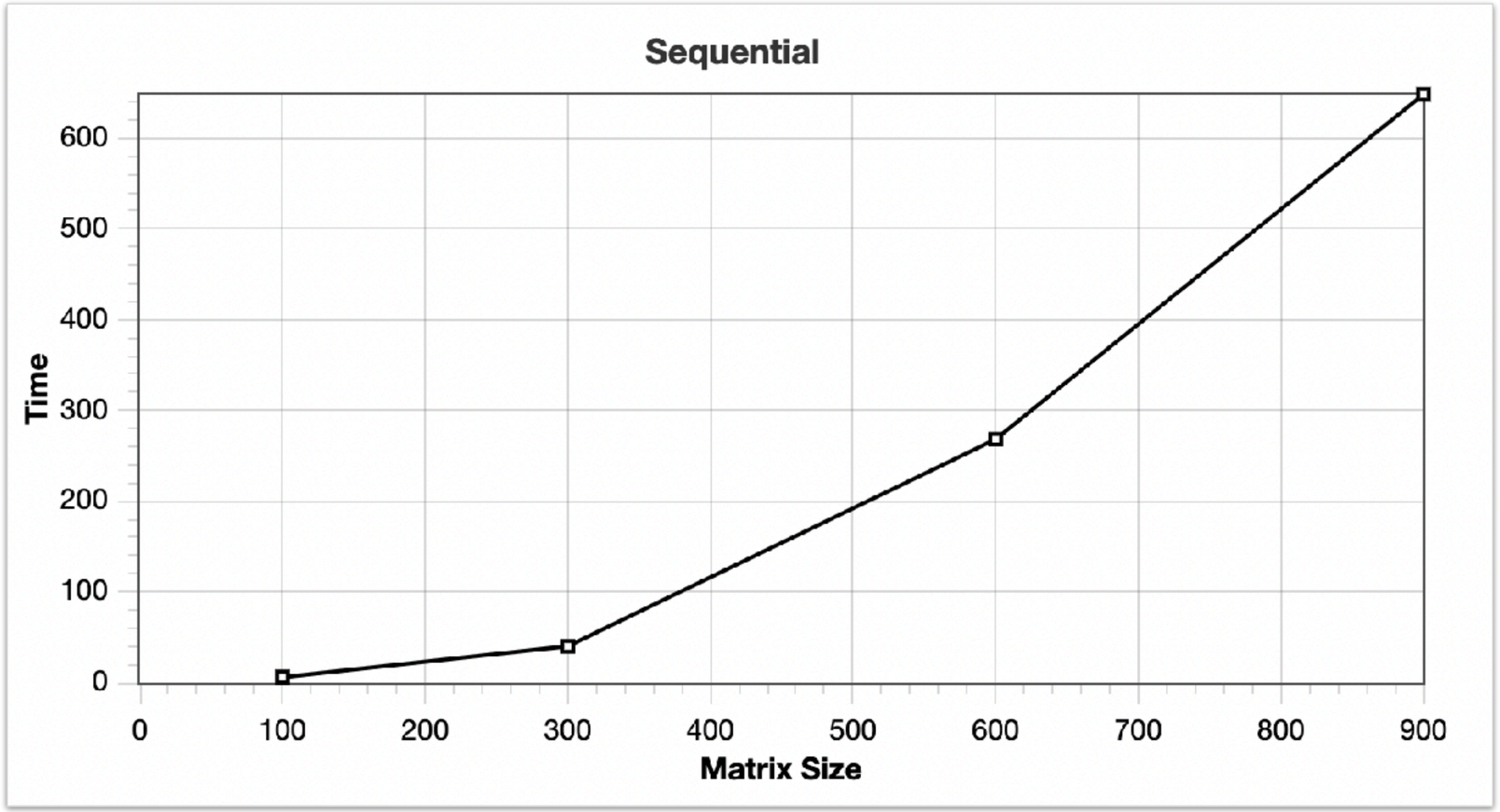
The sequential approach & time complexity



For Square Matrices of size N × N

Time Complexity = $O(N^3)$

The sequential approach (Graph)



The Parallel Approach (Cannon's Algorithm)

- Should be a square matrix
- Divisible by the square root of number of processors

Matrix A

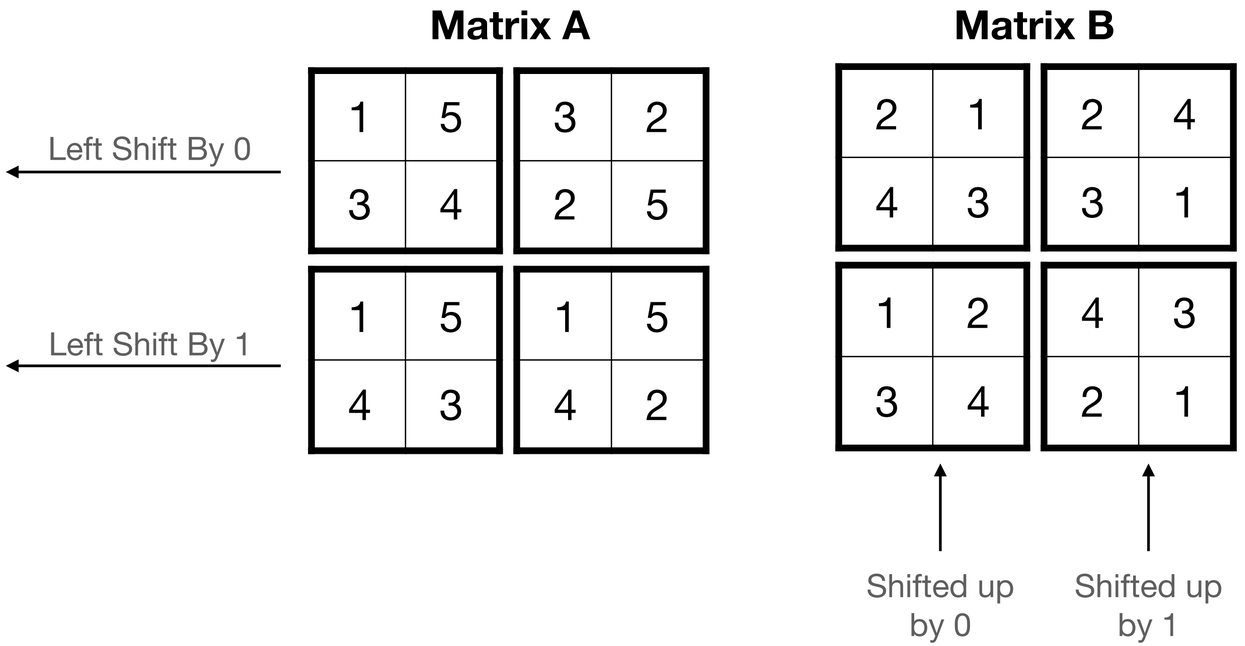
1	5	3	2
3	4	2	5
1	5	1	5
4	3	4	2

×

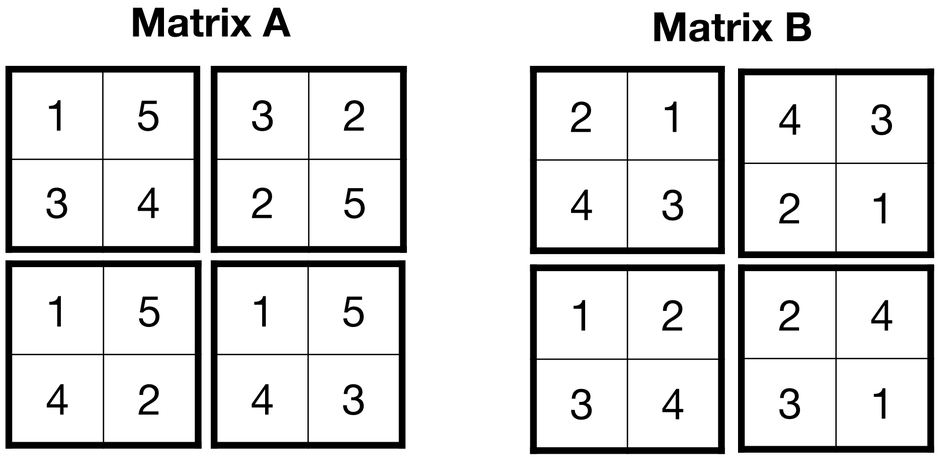
Matrix B

2	1	2	4
4	3	3	1
1	2	4	3
3	4	2	1

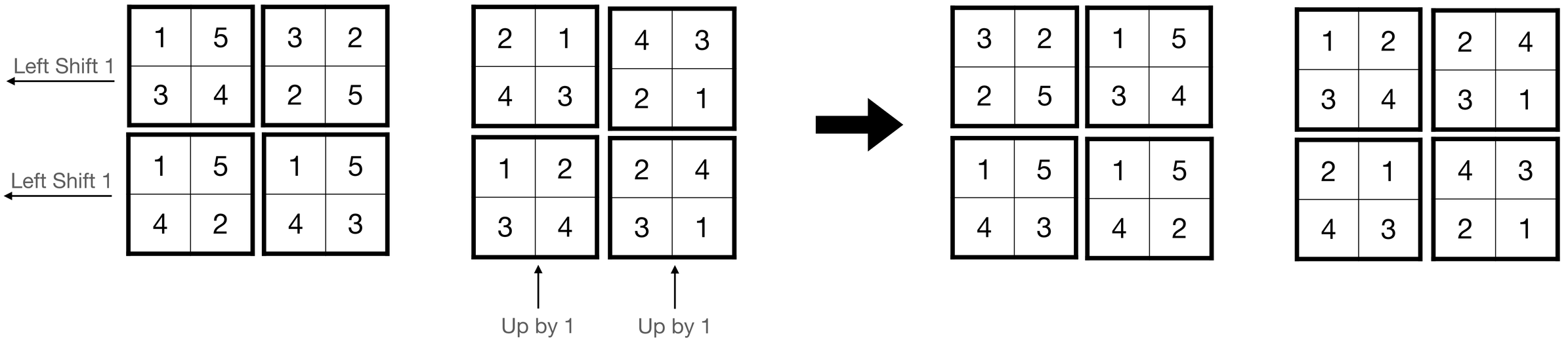
Divide the larger matrices
Into smaller sub-matrices

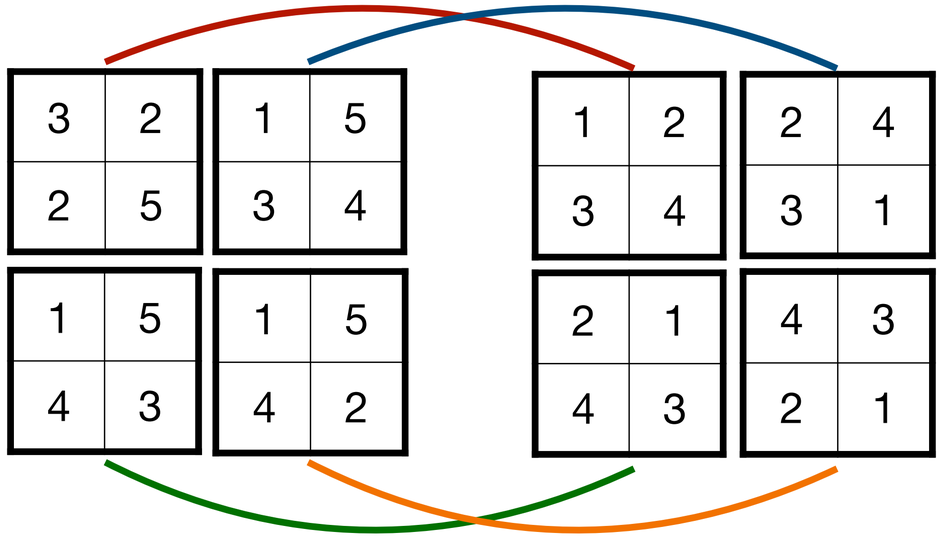


After the Initial Alignment



Step-1 (Shift Operation & Sub-Matrices Multiplication)

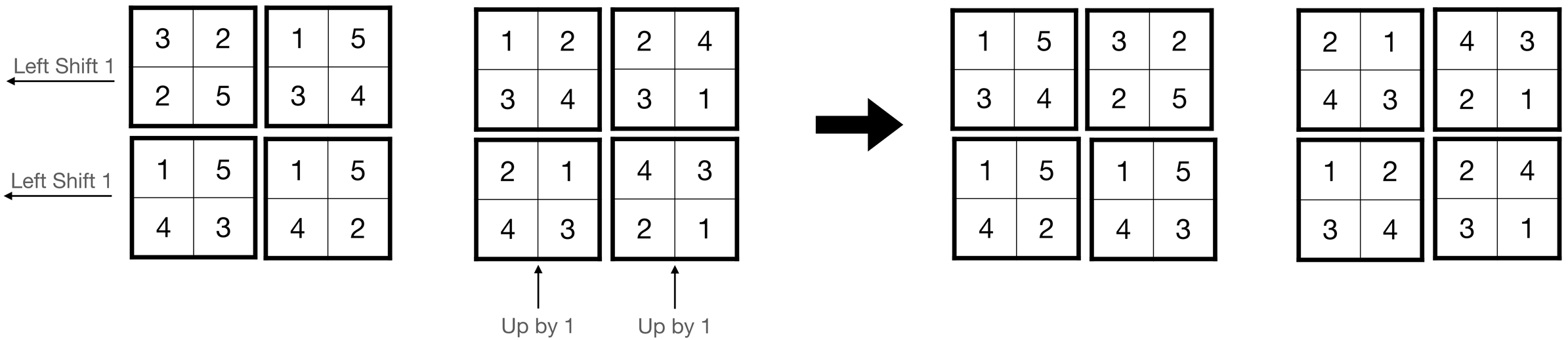


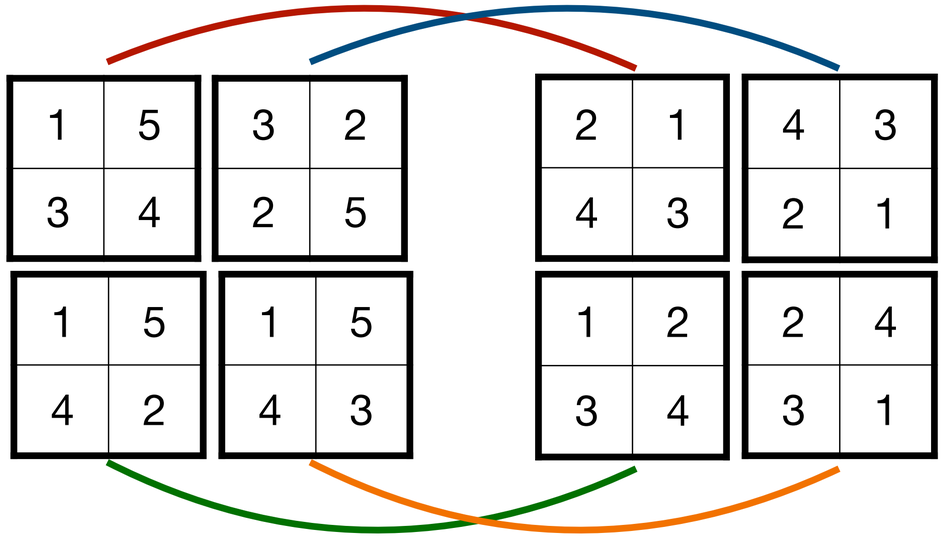


C1

9	14	17	9
17	24	18	16
22	16	14	8
20	13	20	14

Step-2 (Shift Operation & Sub-Matrices Multiplication)

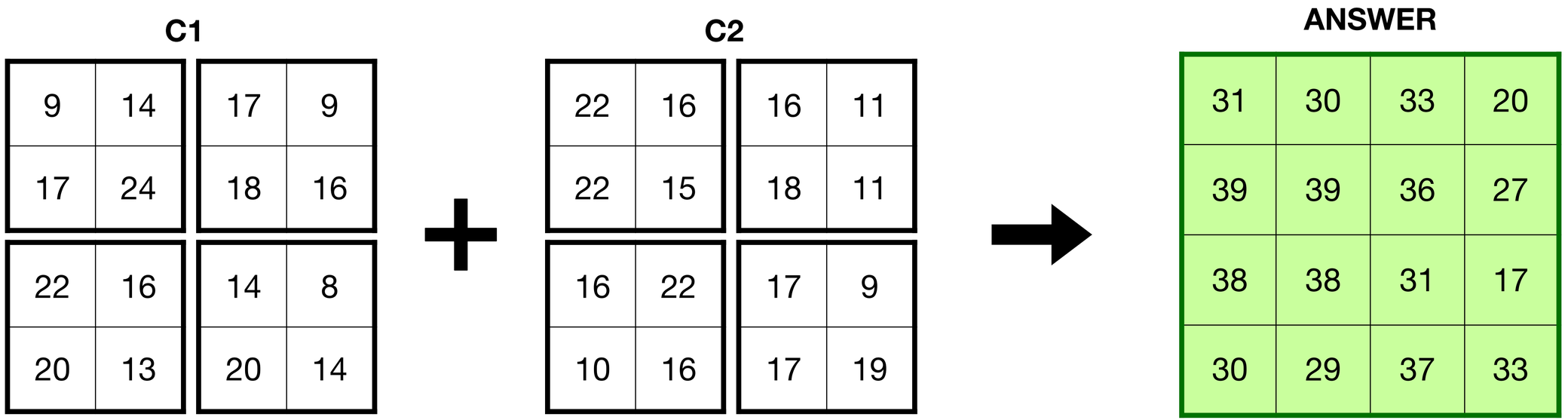




C2

22	16	16	11
22	15	18	11
16	22	17	9
10	16	17	19

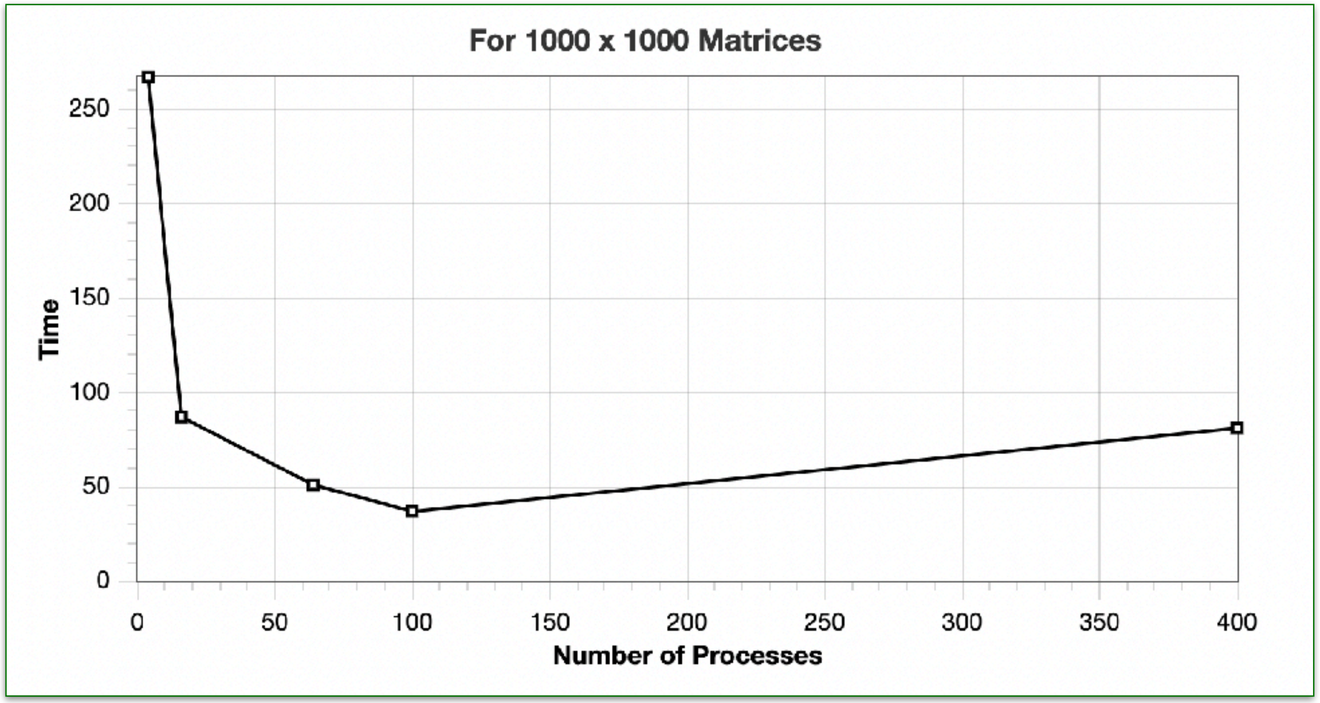
Add the 2 intermediate matrices to get the answer



What I did for Midterm Presentation (Not The Right Way)

For 1000 x 1000 Matrices

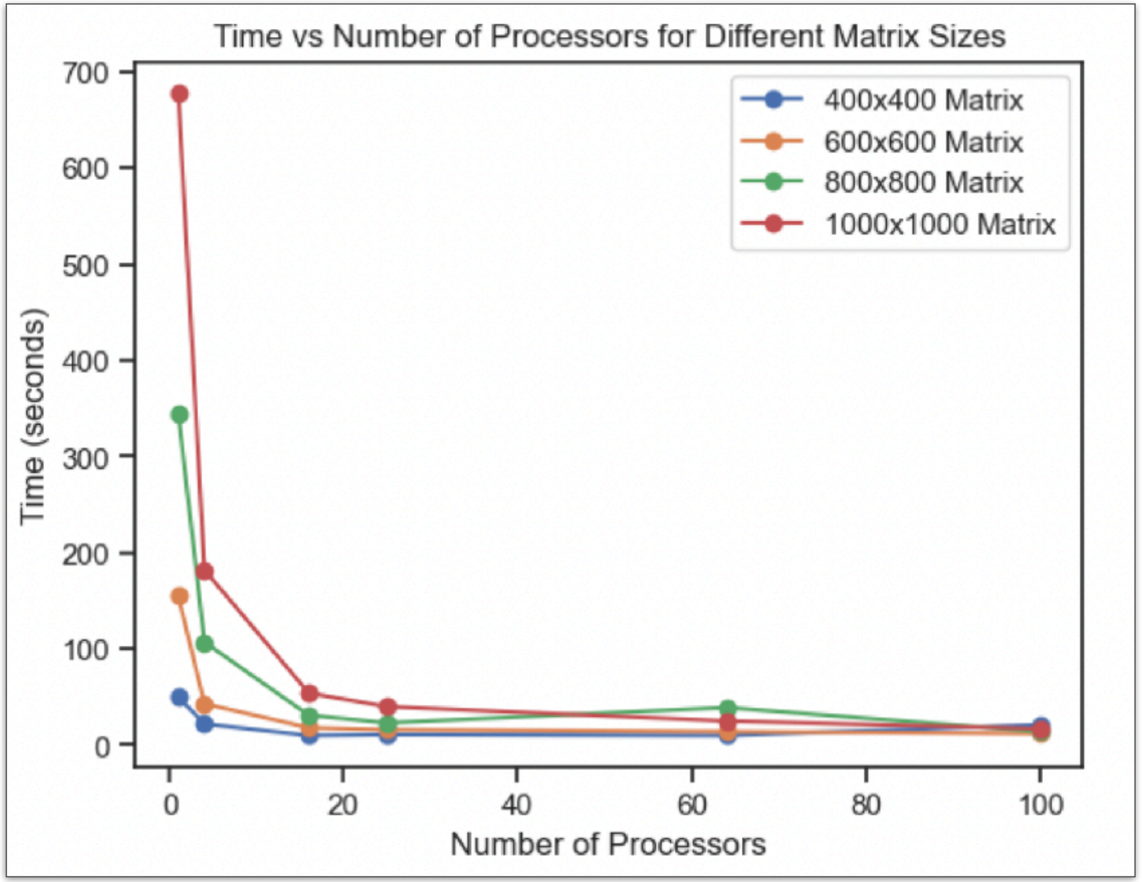
Number of Processes	Distribution of Processes	Time (Seconds)
4	Nodes: 1 Cores/Node: 4	267
16	Nodes: 1 Cores/Node: 16	87
64	Nodes: 4 Cores/Node: 16	51
100	Nodes: 10 Cores/Node: 10	37
400	Nodes: 40 Cores/Node: 10	81



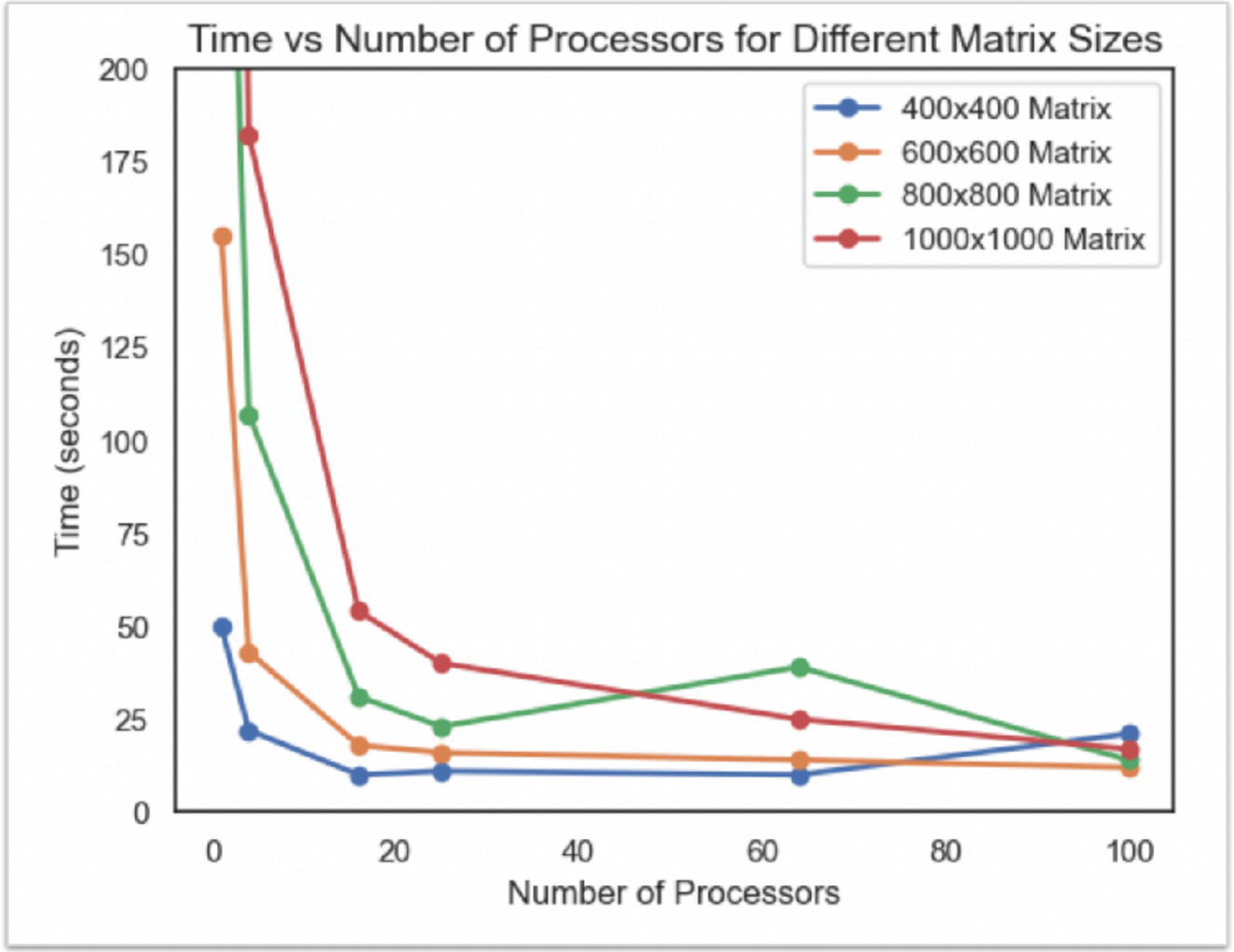
Post Midterm

Time vs Number of Processors

PEs	Nodes	400 x 400 Time(s)	600 x 600 Time(s)	800 x 800 Time(s)	1000 x 1000 Time(s)
1	1	50	155	345	677
4	4	22	43	107	182
16	16	10	18	31	54
25	25	11	16	23	40
64	64	10	14	39	25
100	100	21	12	14	17



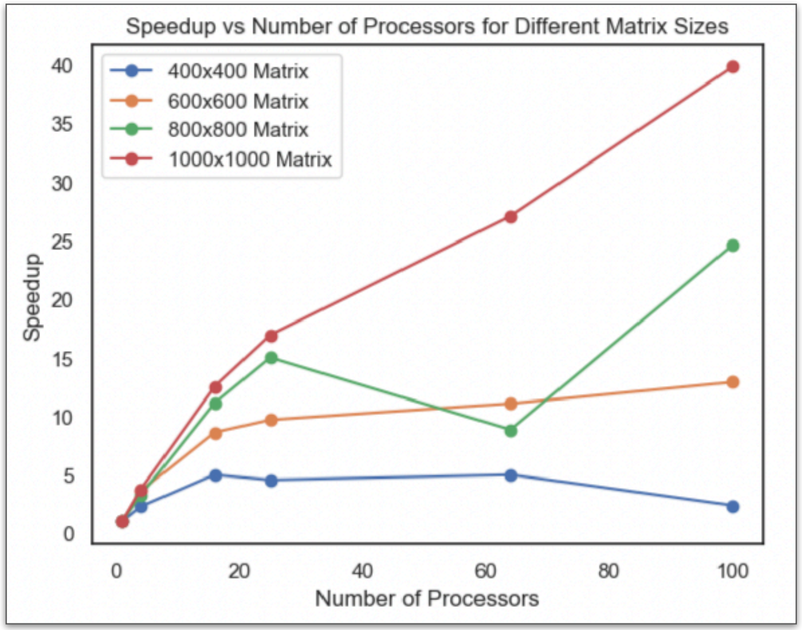
Zoomed Version



Speedup vs Number of Processors

Amdahl's law

PEs	Nodes	400 x 400 Speedup	600 x 600 Speedup	800 x 800 Speedup	1000 x 1000 Speedup
1	1	1	1	1	1
4	4	2.27	3.6	3.22	3.72
16	16	5	8.61	11.12	12.53
25	25	4.54	9.68	15	16.92
64	64	5	11.07	8.84	27.08
100	100	2.38	12.9	24.6	39.82



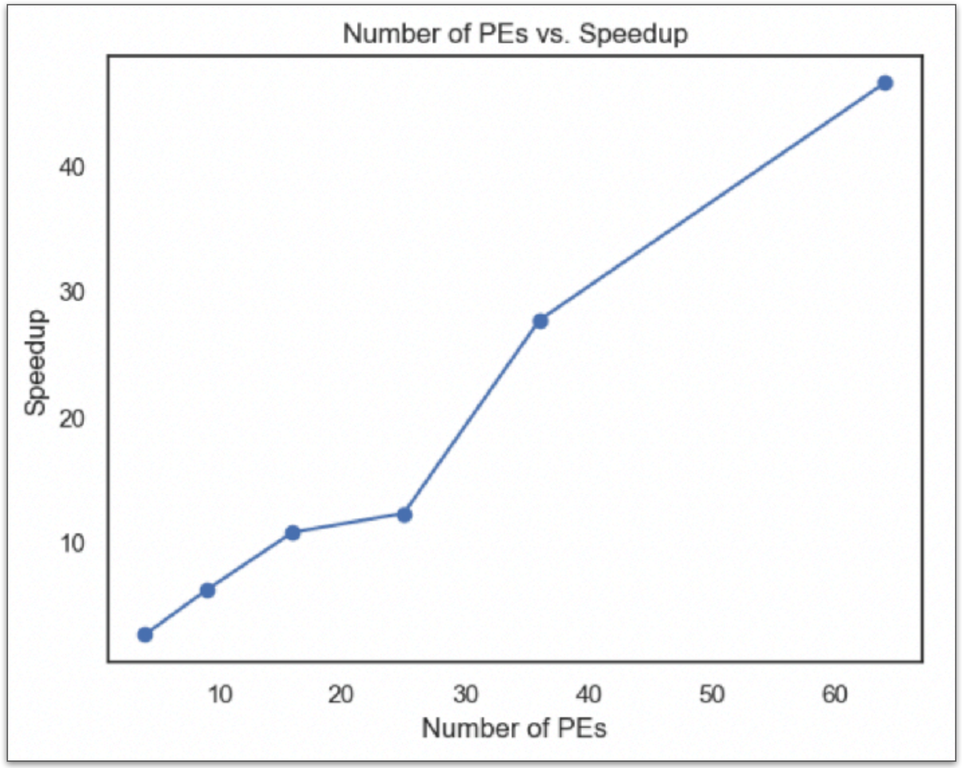
Gustafson's law

Source: Wikipedia

> Gustafson's law addresses the shortcomings of Amdahl's law, which is based on the assumption of a fixed problem size, that is of an execution workload that does not change with respect to the improvement of the resources. Gustafson's law instead proposes that programmers tend to increase the size of problems to fully exploit the computing power that becomes available as the resources improve.[2]

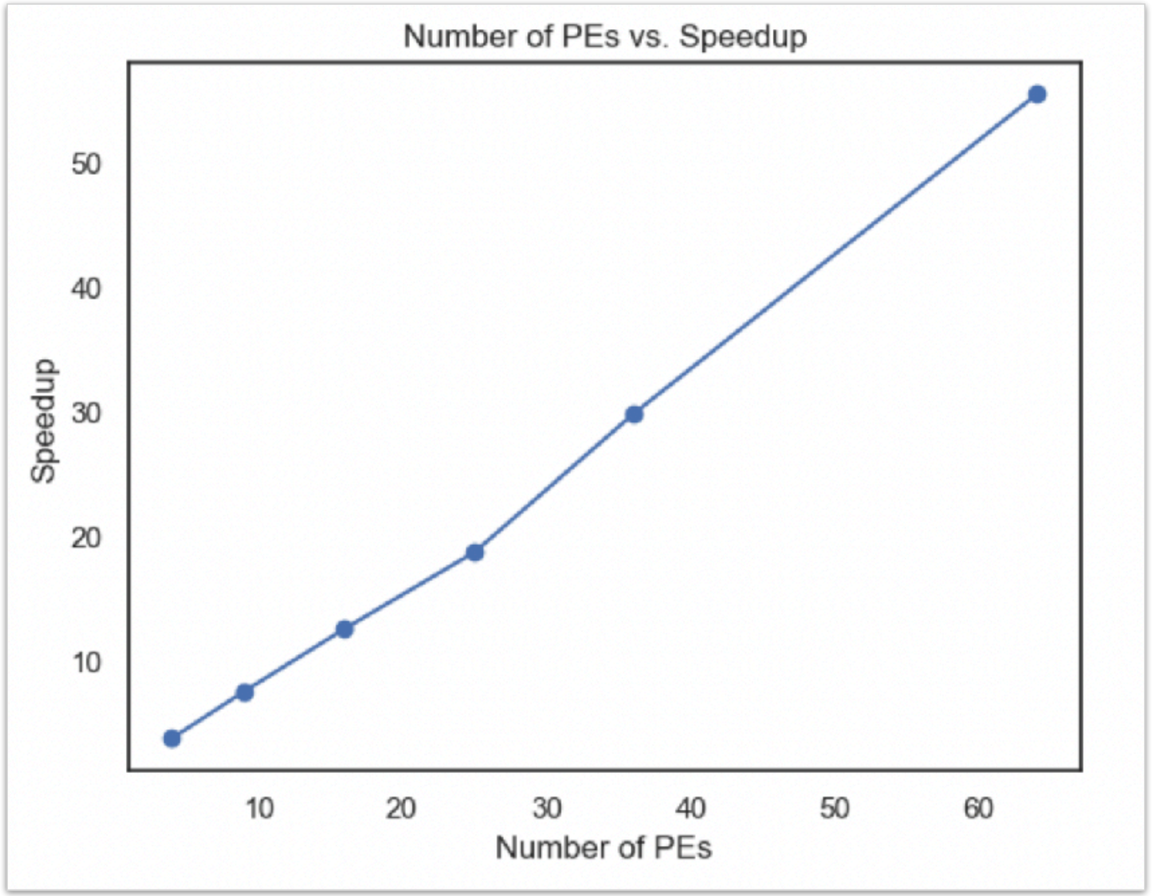
200 x 200 Per Processor

PEs	Matrix Size	Time (s)	Speedup
4	400 x 400	19	2.63
9	600 x 600	24	6.2
16	800 x 800	31	10.8
25	1000 x 1000	53	12.3
36	1200 x 1200	41	27.7
64	1600 x 1600	58	46.6



400 x 400 Per Processor

PEs	Matrix Size	Time	Speedup
4	800 x 800	89	3.77
9	1200 x 1200	151	7.53
16	1600 x 1600	215	12.58
25	2000 x 2000	280	18.73
36	2400 x 2400	301	29.8
64	3200 x 3200	363	55.5



Next Steps

- I would like to explore OpenMP to implement matrix-matrix multiplication
- Combination of OpenMP and MPI

References

- AMCS Lecture (Speedup & Amdahl's law)
- https://dl.acm.org/doi/pdf/10.1145/263580.263591
- https://mpi4py.readthedocs.io/en/stable/tutorial.html
- https://docs.ccr.buffalo.edu/en/latest/
- https://en.wikipedia.org/wiki/Amdahl%27s_law
- https://en.wikipedia.org/wiki/Gustafson%27s_law