PARALLEL MATRIX MULTIPLICATION

Presented by: Prithvisagar Rao

University at Buffalo The State University of New York

Problem Statement

Given a matrix A(nxn) and a matrix B(nxn), the matrix C resulting from the operation of multiplication of matrices A and B,

C = A x B is given as:

$$c_{i,j} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$$

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

To calculate one value in matrix C we need to perform *n* multiplications and *n-1* additions. For a matrix of size n^2 this results in n^3 calculations.

Sequential Algorithm

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        c[i][j] = 0;
        for (k=0; k<n; k++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
    }
}
```

As we can see, the sequential algorithm has 3 nested for loops which results in a O(n^3) time complexity.

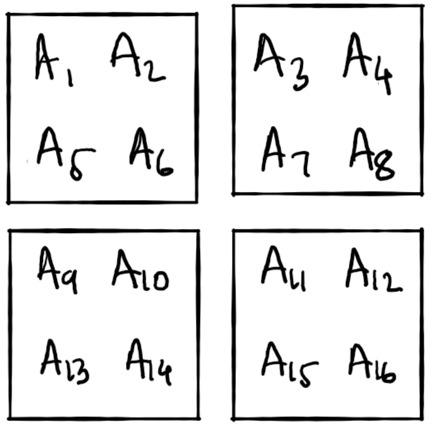
Parallel Algorithm

Parallel Algorithm for Matrix Multiplication

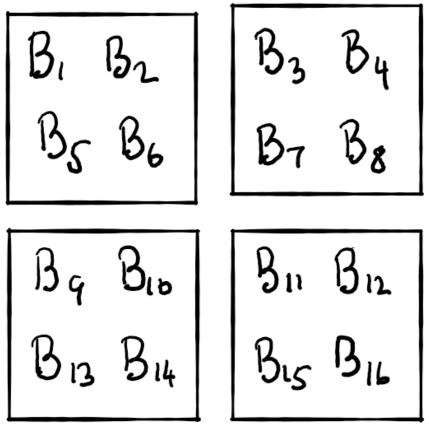
1. Partition A and B into P square blocks $A_{i,j}$ and $B_{i,j}$ where P is the number of processors available.
2. Ensure each process can maintain a block of A and B by creating a matrix of processes of size $P^{1/2}$ x $P^{1/2}$
3. The blocks are multiplied together and the results are added to the partial results in the C sub-blocks.
4. The sub-blocks of A are shifted one step to the left and the sub-blocks of B are shifted one step up.
5. Repeat this process for $P^{1/2}$ times

Parallel Algorithm

Divide the initial input matrix into P sub blocks and distribute the data to their processes



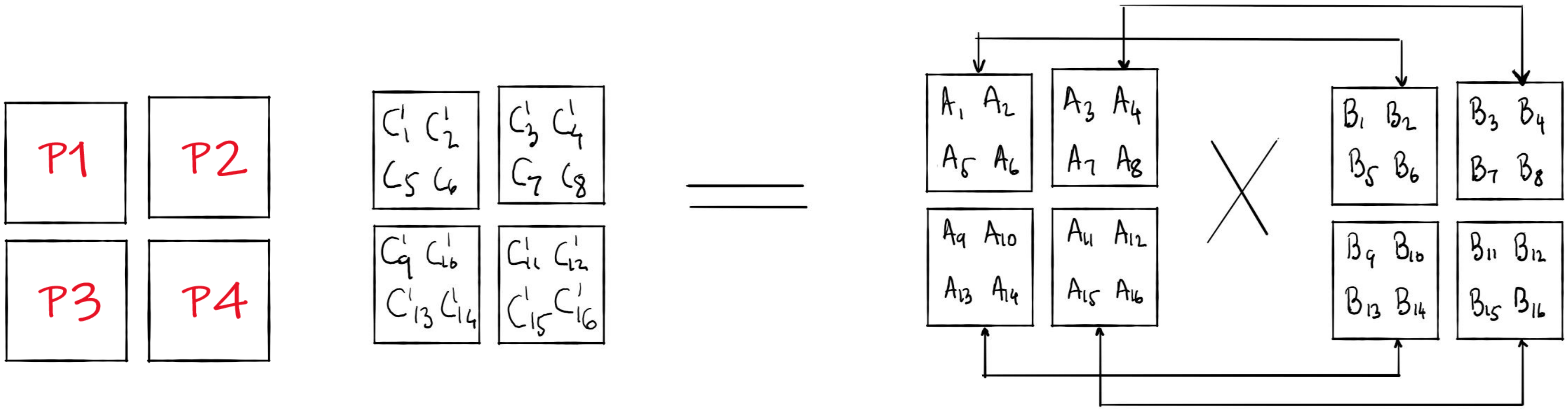
Input matrix A



Input matrix B

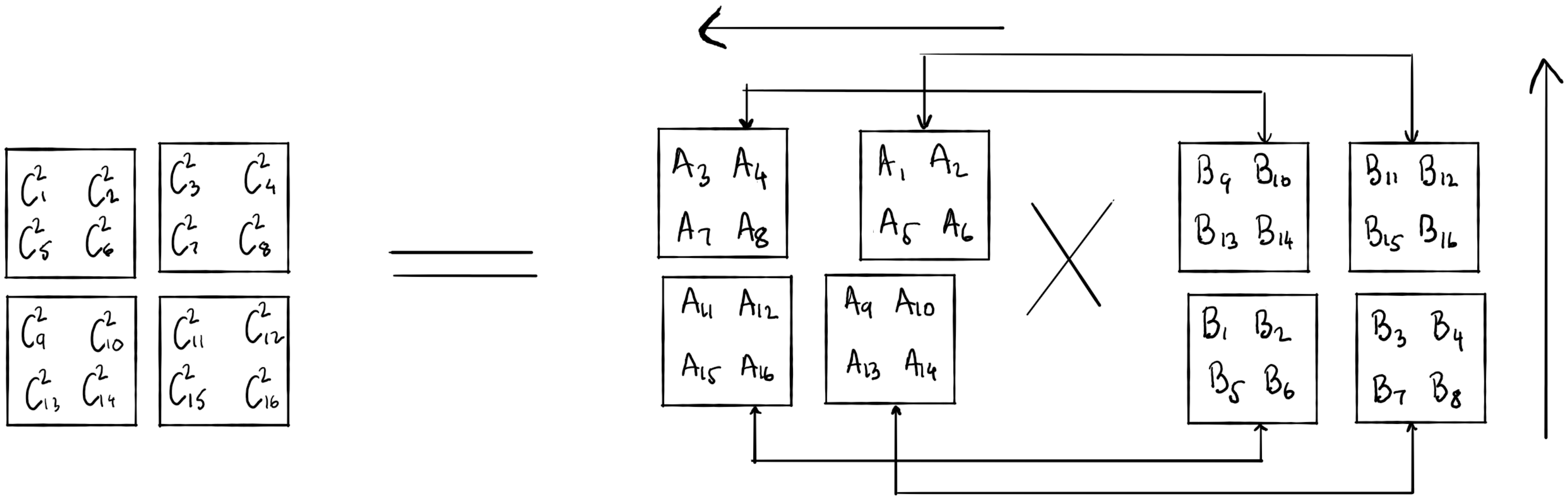
Parallel Algorithm

The processors perform the local multiplication based on the initial arrangement



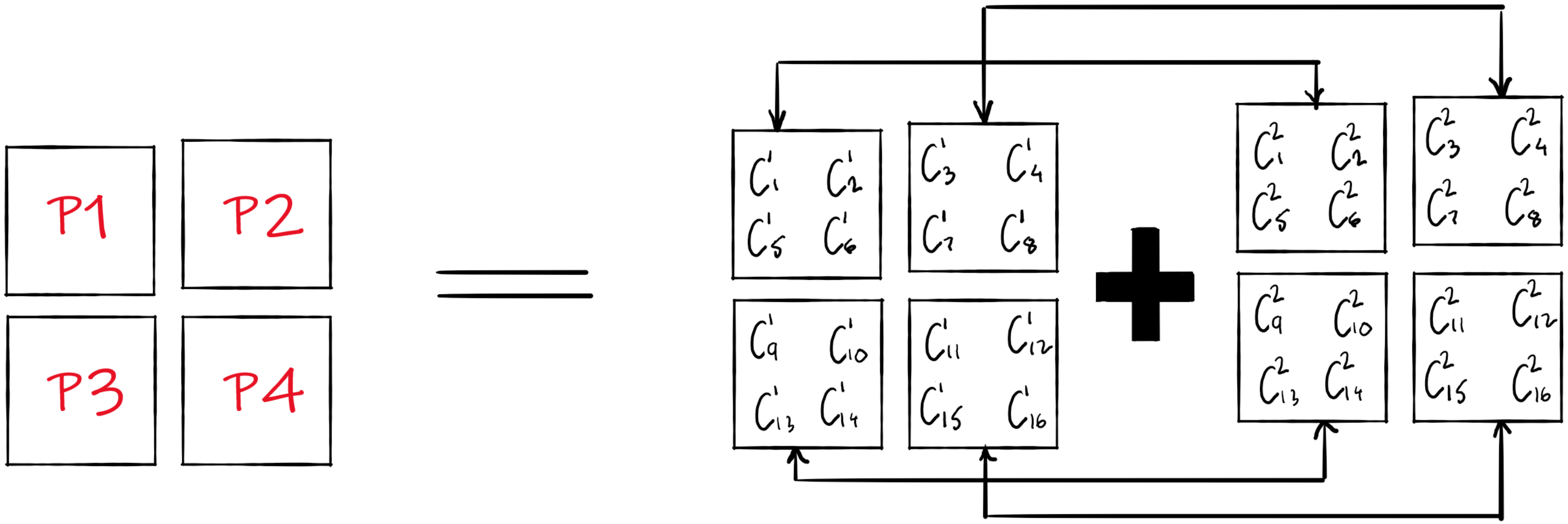
Parallel Algorithm

Shift matrix A to the left and matrix B upwards, perform the local multiplication and add it to the partial result



Parallel Algorithm

Add the partial answers



Results

Parameters used for running the parallel approach:

- Square matrices were used
- Matrix dimensions ranged from 2000 to 8000
- Number of processors used – 4,9,16,25,36,49,64

Parallel Approach

Size of Matrices

No of Processors

	2000	3000	4000	5000	6000	7000	8000
4	7.963	36.769	97.568	269.82	394.163	779.479	1221.523
9	4.913	19.459	62.256	192.071	233.235	405.353	859.578
16	2.302	12.235	24.653	75.427	113.422	226.991	264.74
25	1.426	6.147	15.657	33.853	57.588	94.843	175.485
36	3.016	10.388	29.480	54.572	59.495	101.63	158.162
49	4.167	12.547	38.996	60.971	63.689	91.627	133.747
64	5.753	16.764	46.504	67.842	69.689	89.785	115.621

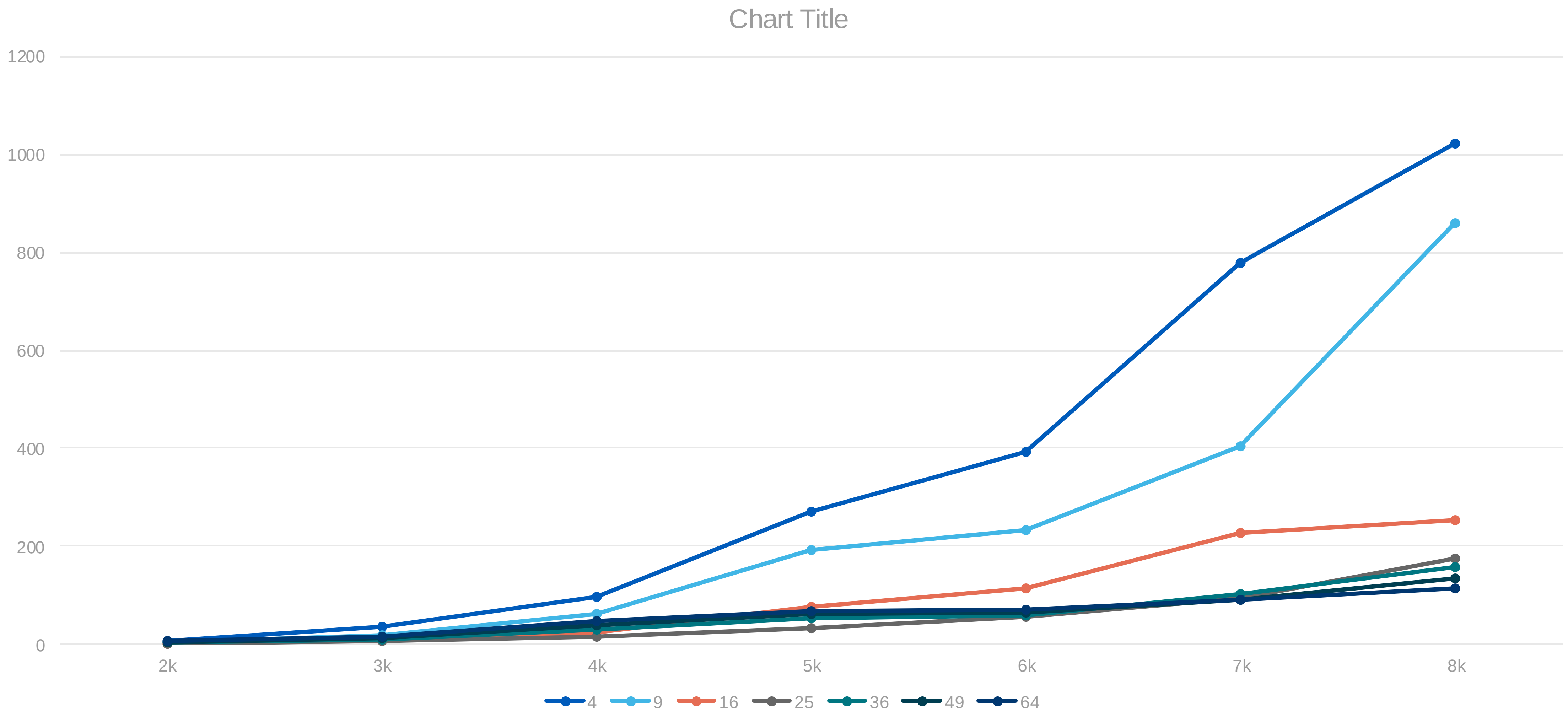


Chart Title
1200
1000
800
600
400
200
0
2k
3k
4k
5k
6k
7k
8k
4
9
16
25
36
49
64

Observations

- Computation running time decreased as a result of parallelization.
- Increase in number of processors does not necessarily result in reduction in running time due to communication overhead.
- A good balance between number of processors and runtime was observed at 25 number of processors.
- Number of processors must be perfect squares.
- Data must be equally distributed among the processors.
- Got a good idea of parallelization.

Future work

- Ran the simple block matrix multiplication in parallel. Other algorithms such as Block-striped algorithm and Fox's algorithm can be run and compared.
- Compare results with OpenMP implementation.

Thank you