PDC HW4 Report

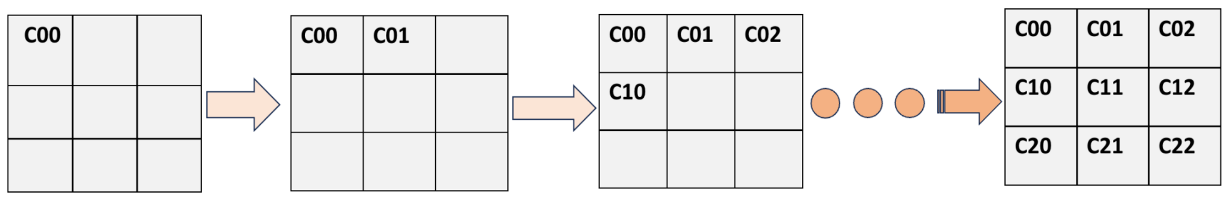
Group No: 58

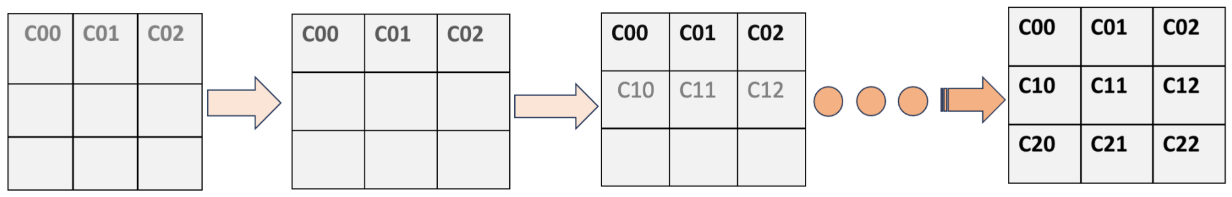
1. Aswin Sampath - 53844684 - saswin@uci.edu
2. Shreya Chetan Pawaskar - 12041645 - pawaskas@uci.edu

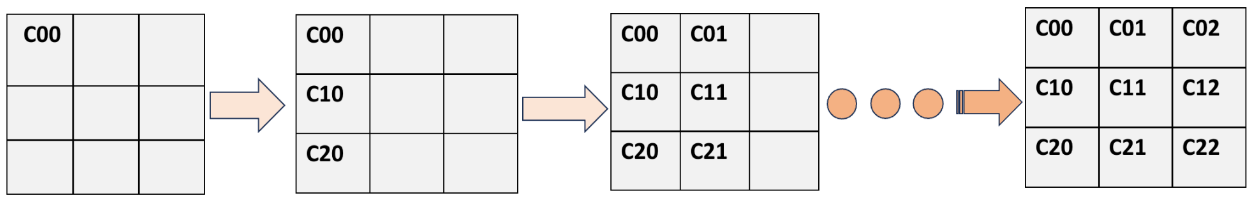
Q 1) Assuming execution of the above program in a single processor computer, characterize the manner in which the computation progresses for each one of the 6 possible nested loop cases. Provide data to memory mapping. Show how the main arithmetic expression progresses, accumulating its value as the indices advance, with the innermost loop being the one that advances fastest.

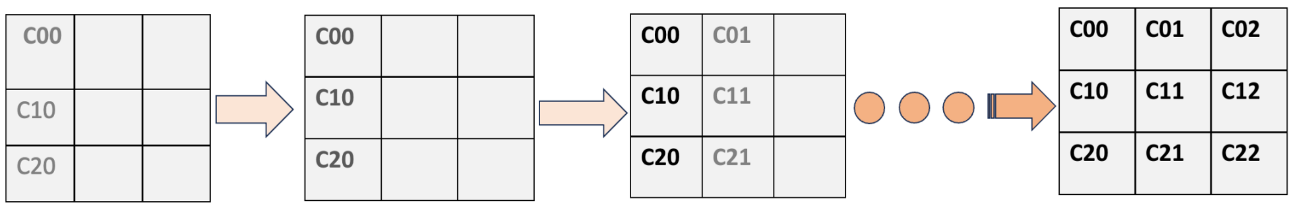
Let's consider input matrices A and B, each with size N and total elements (N x N) , and let the output matrix be C with size N and total elements (NxN).

Let Cij be the element that is present in i^{th} row and j^{th} column in C.

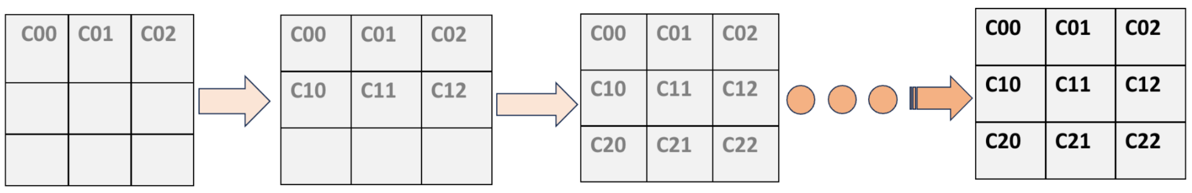
❖ Combination : (i,j,k)
  - ➢ **Each element** of the resultant matrix is **fully** calculated **row wise**..
  - ➢ We compute one full row at a time, and then move on to the next row to complete the entire result matrix. Here's a simplified diagram for a 3x3 matrix.
  - ➢ 

❖ Combination : (i,k,j)
  - ➢ **Each element** of the resultant matrix is **partially** calculated for the entire **row**.
  - ➢ For instance, we first calculate a portion of C00, followed by C01, and then C02, until we've partially computed the entire row. After this partial row computation, we repeat the process to obtain the final values for that row: C00, C01, and C02. This pattern continues for the entire row, and once completed, we move on to the next row in a similar fashion.
  - ➢ 

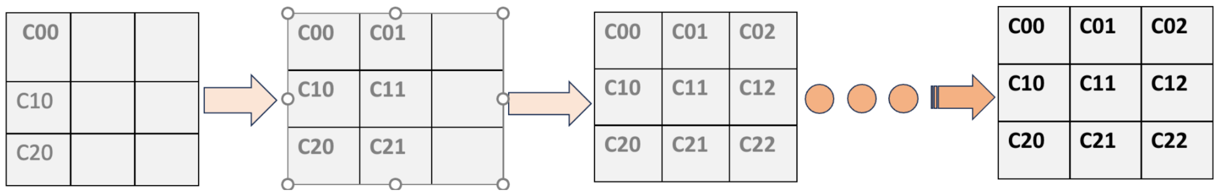
❖ Combination : (j,i,k)
  - ➢ **Each element** in the resulting matrix is calculated **fully column-wise**, one column at a time, until the entire matrix is computed.
  - ➢ We calculate C00, C10, C20 fully and move on to the next column C01,C11,C21 and subsequently evaluate the entire matrix.
  - ➢ 

❖ Combination: (j,k,i)
  - ➢ **Each Element** is computed in a **partial column-wise** fashion, progressing as C00, C10, C20, and then repeating for each column before moving on to the next one, such as C01, C11, C21, and so forth.
  - ➢ 

❖ Combination: (k,i,j)
  - ➢ The **entire** resultant matrix(C) is calculated **partially** in **row wise** direction.

- C00, C01, C02...till CNN gets computed partially and the process repeats NxN times to complete the resultant matrix.
- 

❖ Combination: (k,j,i)

- The **entire** C matrix is calculated **partially in column wise direction**,
- C00, C01, C02...CNN gets computed partially and the process repeats NxN times to complete the matrix.
- 

Q 2) For each one of the nested loop cases, discuss the mapping of the computation on a n-processor ring if, for each of the 6 cases, the outermost loop is used to unfold and parallelize the computation.

Assumption :

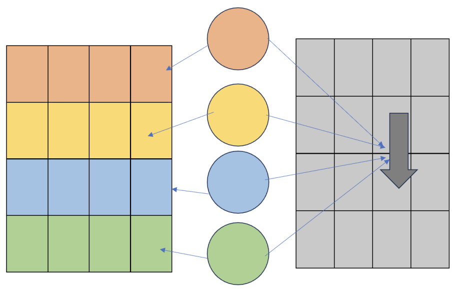
Input Matrices each of size N having N*N elements parallelizing on a N-processor ring.

Every processor has access to the resultant matrix and it updates the value residing inside memory.

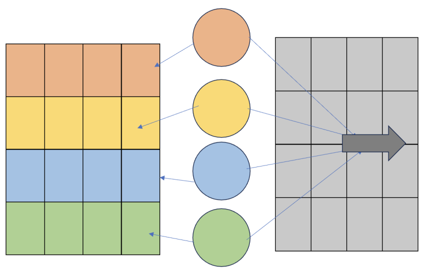
The formula to calculate the C[i][j] element is below.

$$C[i][j] += A[i][k] * B[k][j]$$

➔ Combination : (i,j,k)

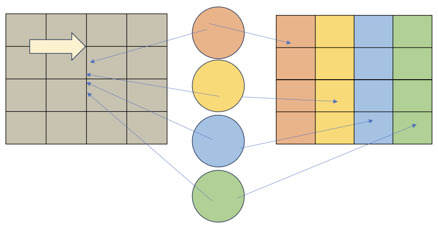
- Loop unfolding and parallelizing on **i**
- Each processor will compute **rows** of the resultant matrix C. That is , **ith processor** will compute **i**th **row** of C.
- The **i**th **processo**r will have access to **i**th **row of A** and the **entire matrix of B** and multiply **each row of A**, **column wise** with B.
- 

➔ Combination: (i,k,j)

- Loop unfolding and parallelizing on **i**
- Each processor will compute **rows** of the resultant matrix C. That is , **i**th **processor** will compute **i**th **row** of C.
- The **i**th **processo**r will have access to **i**th **row of A** and the **entire matrix of B** and multiply **each row of A**, **row wise** with B.
- 

➔ Combination (j,i,k)

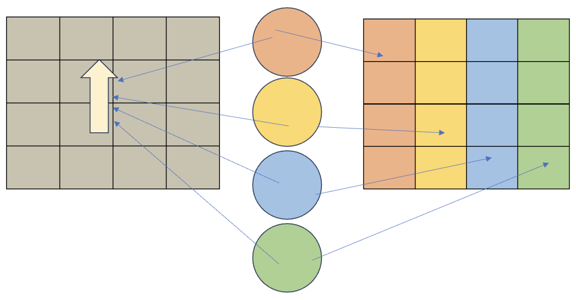
- Loop unfolding and parallelizing on **j**
- Each processor will compute individual columns of the resultant matrix C.
- That is , jth processor will compute jth col of C.
- The **j**th **processo**r will have access to the entire **matrix A** and the **j**th **column of B** and multiply **all rows of A (row wise)** with **j**th **col of B**



- ◆

➔ Combination (j,k,i)

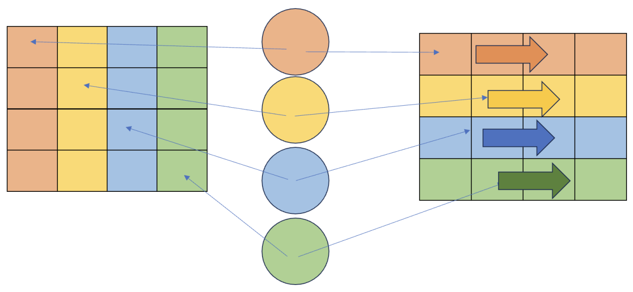
- ◆ Loop unfolding and parallelizing on **j**
- ◆ Each processor will compute individual columns of the resultant matrix C.
- ◆ That is , jth processor will compute jth col of C.
- ◆ The **jth processo**r will have access to the entire **matrix A** and the **jth column of B** and multiply **all columns of A (column wise)** with the jth **col of B.**



- ◆

➔ Combination (k,i,j)

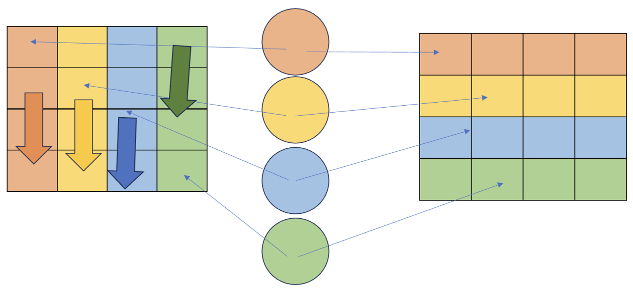
- ◆ Loop unfolding and parallelizing on k
- ◆ Each processor will compute the partial sum for each element in the resultant matrix.
- ◆ **Kth** processor will get access to **kth column of matrix A** and **kth row of matrix B**
- ◆ Each element of A's column will get multiplied with entire row in B



- ◆

➔ Combination (k,j,i)

- ◆ Loop unfolding and parallelizing on k
- ◆ Each processor will compute the partial sum for each element in the resultant matrix.
- ◆ **Kth** processor will get access to **kth column of matrix A** and **kth row of matrix B**
- ◆ Entire Col of A will get multiplied with each element of matrix B or in other words , each element of matrix B is multiplied with the entire column of matrix A.

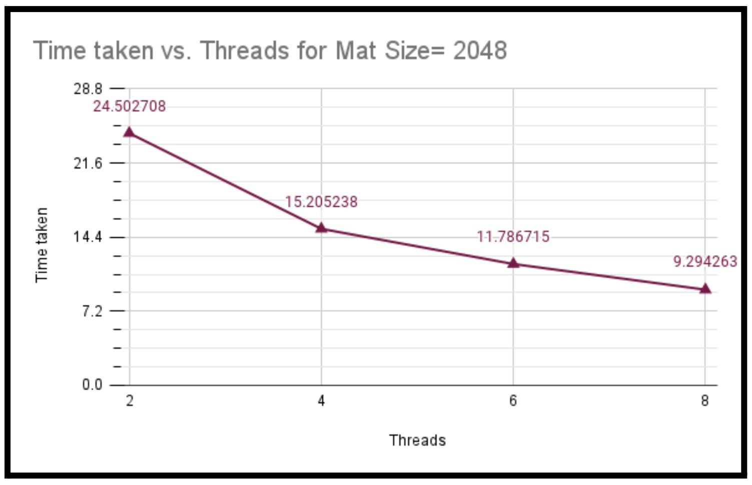


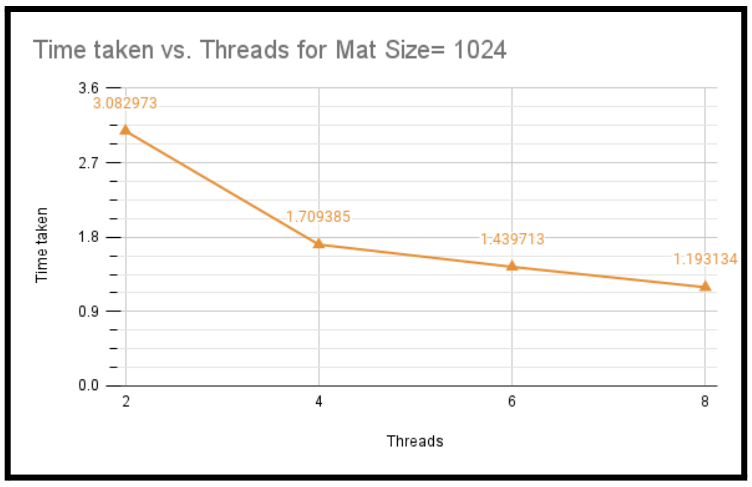
- ◆

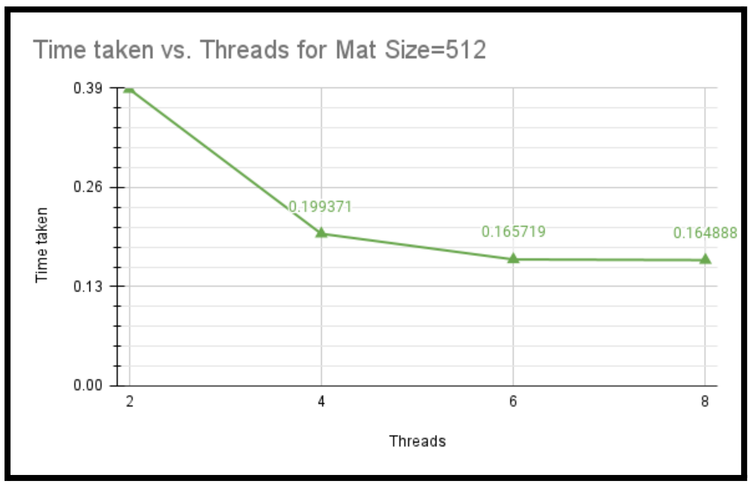
4. Discussion of the gain in performance of the multi-thread implementation as a function of the number of threads concurrently used. Add plots and figures as necessary to explain your results.

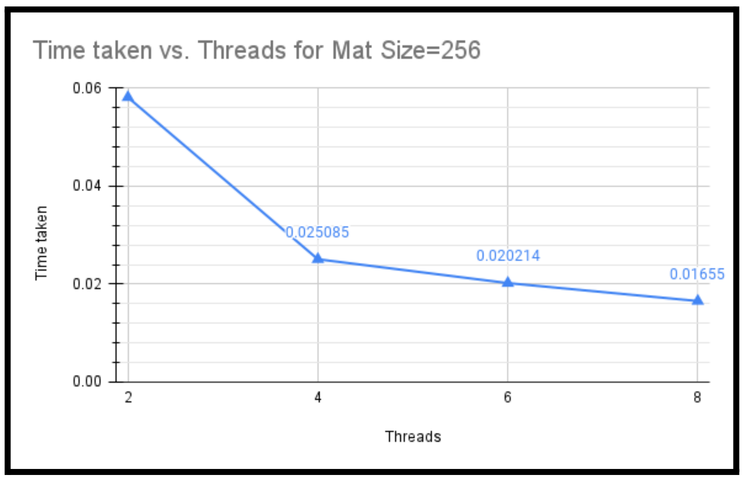
	Threads			
Matrices size	2 threads	4 threads	6 threads	8 threads
64	0.001305	0.000813	0.000958	0.000479
128	0.010528	0.004686	0.002875	0.003591
256	0.058148	0.025085	0.020214	0.01655
512	0.388909	0.199371	0.165719	0.164888

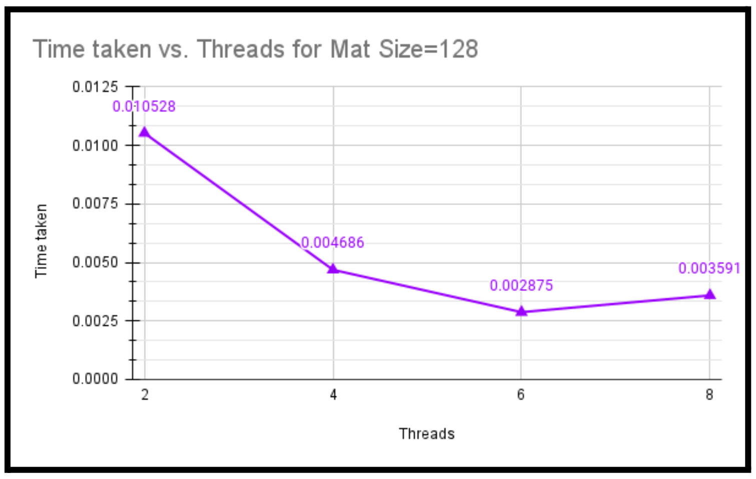
1024	3.082973	1.709385	1.439713	1.193134
2048	24.502708	15.205238	11.786715	9.294263

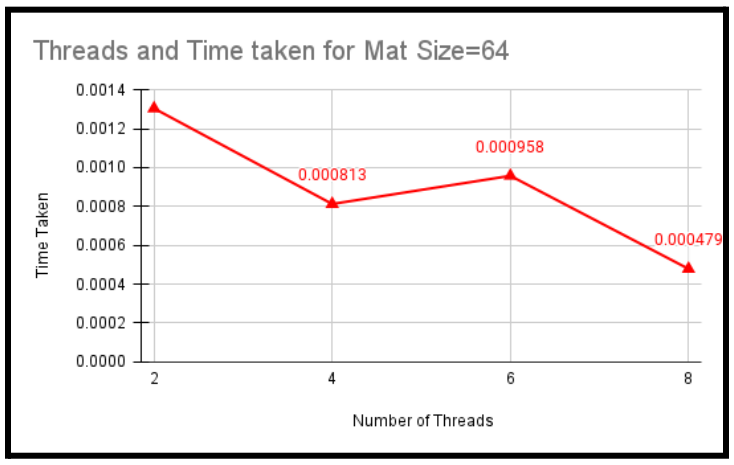

Time taken vs. Threads for Mat Size= 2048
Time taken
Threads

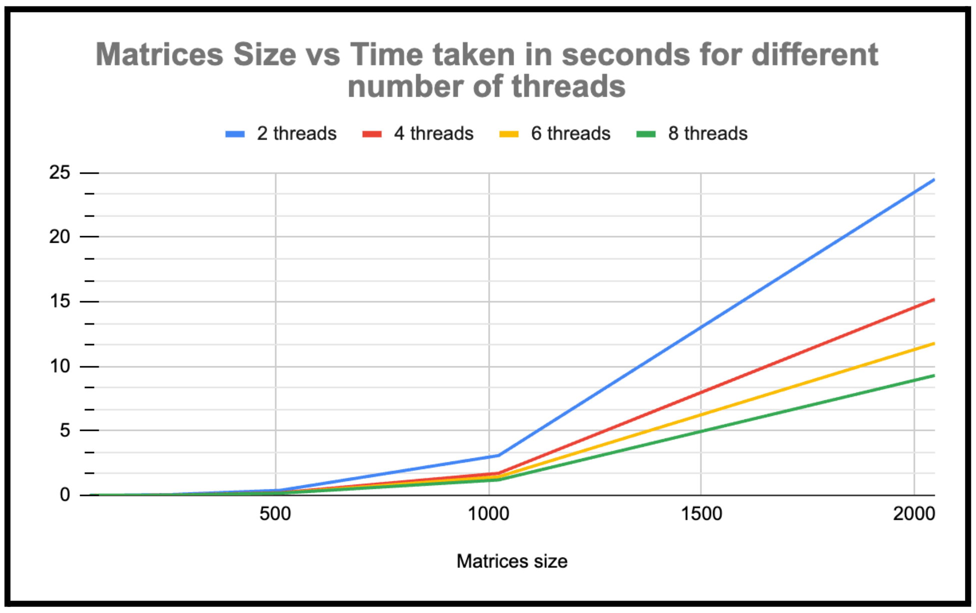

Time taken vs. Threads for Mat Size= 1024
Time taken
Threads


Time taken vs. Threads for Mat Size=512
Time taken
Threads


Time taken vs. Threads for Mat Size=256
Time taken
Threads


Time taken vs. Threads for Mat Size=128
Time taken
Threads


Threads and Time taken for Mat Size=64
Time Taken
Number of Threads


Matrices Size vs Time taken in seconds for different number of threads
2 threads
4 threads
6 threads
8 threads
Matrices size