The characterization of the computation progress in a single processor computer, for each of the six mentioned per- mutations.

Formula used to compute the result matrix C which is the multiplication of input matrix A and B is as follows:
$\mathbf{C_{ij} = C_{ij} + A_{ik}*B_{kj}}$.

Using above computation formula, below is the analysis of multiplication for 6 permutations of I, j and k unfolding:

(i,j,k):

- k increases first from 0 to n-1 followed by j and eventually i increases from 0 to n-1.
- In the first iteration of j, C_{00} is calculated. In the second iteration of j, C_{01} is calculated and so on.
- Hence, in the first iteration of i, we will get C_{00}, C_{01}, C_{02}.. and i.e. **first row of result matrix C**.
- In the **second iteration of i**, **2nd row of result matrix C** is calculated. In the **nth iteration of** i, **the nth row of result matrix C is calculated.**

(i,k,j):

- j increases first from 0 to n-1 followed by k and eventually i increases from 0 to n-1.
- In the first iteration of k, partial values of C_{00}, C_{01} and C_{02} are calculated. In the second iteration of k, the remaining partial values of C_{00}, C_{01} and C_{02} are calculated.
- Hence, in the first iteration of i, we will get the full values of C_{00}, C_{01}, C_{02} i.e. **first row of result matrix C.**
- In the **second iteration of i**, **2nd row of result matrix C** is calculated. In the **nth iteration of** i, **the nth row of result matrix C is calculated.**

(j,i,k):

- k increases first from 0 to n-1 followed by i and eventually j increases from 0 to n-1.
- In the first iteration of i, C_{00} is calculated. In the second iteration of i, C_{10} is calculated and so on.
- Hence, in the first iteration of j, we will get C_{00}, C_{10}, C_{20}.. and i.e. **first column of result matrix C.**
- In the **second iteration of j**, **2nd column of result matrix C** is calculated. In the **nth iteration of** j, **the nth column of result matrix C is calculated.**

(j,k,i):

- i increases first from 0 to n-1 followed by k and eventually j increases from 0 to n-1.
- In the first iteration of k, **partial values** of C_{00}, C_{10} and C_{20} are calculated. In the second iteration of k, the **remaining partial values** of C_{00}, C_{10} and C_{20} are calculated.
- Hence, in the first iteration of j, we will get the full values of C_{00}, C_{10}, C_{20} i.e. the **first column of the result matrix C.**
- In the **second iteration of j**, **2nd column of result matrix C** is calculated. In the **nth iteration of** j, **the nth column of result matrix C is calculated.**

(k,i,j):

- j increases first from 0 to n-1 followed by i and eventually k increases from 0 to n-1.
- In the first iteration of i, **partial values** of C_{00}, C_{01} and C_{02} are calculated. In the second iteration of i, the **partial values** of C_{10}, C_{11} and C_{12} are calculated. Hence, in each iteration of i, we calculate the **partial value** of **each row** of the result matrix C.
- Hence, in the **first iteration of k**, we will get the **partial values for every rows of result matrix C.**
- In the **next iteration of k**, we will get the **remaining partial values for every rows of result matrix C**.

- Eventually, in the final iteration of k, we will get the complete value for **every rows of result matrix C.**

(k,j,i):

- i increases first from 0 to n-1 followed by j and eventually k increases from 0 to n-1.
- In the first iteration of j, **partial values** of C_{00}, C_{10} and C_{20} are calculated. In the second iteration of j, the **partial values** of C_{01}, C_{11} and C_{21} are calculated. Hence, in each iteration of j, we calculate the **partial value** of **each column** of the result matrix C.
- Hence, in the **first iteration of k**, we will get the **partial values for every columns of result matrix C.**
- In the **next iteration of k**, we will get the **remaining partial values for every columns of result matrix C.**
- Eventually, in the final iteration of k, we will get the complete value for **every columns of result matrix C.**

Analysis for N Processor ring System

(i,j,k)			
P1	**P2**	**...**	**PN**
A00	A10		An-10
A01	A11		An-11
A0n-1	A1n-1		An-1n-1
B00	B01		B0n-10
B10	B11		B1n-11
Bn-10	Bn-11		Bn-1n-1
C00	C10		CN-10
C01	C11		CN-11
C0N-1	C1N-1		CN-1N-1

(i,k,j)			
P1	**P2**	**...**	**PN**
A00	A10		An-10
A01	A11		An-11
A0n-1	A1n-1		An-1n-1
B00	B10		BN-10
B01	B11		BN-11
B0N-1	B1N-1		BN-1N-1
C00	C10		CN-10
C01	C11		CN-11
C0N-1	C1N-1		CN-1N-1

(j,k,i)			
P1	**P2**	**...**	**PN**
A00	A01		A0n-10
A10	A11		A1n-11
An-10	An-11		An-1n-1
B00	B01		B0n-10
B10	B11		B1n-11
Bn-10	Bn-11		Bn-1n-1
C00	C01		C0n-10
C10	C11		C1n-11
Cn-10	Cn-11		Cn-1n-1

(j,i,k)			
P1	**P2**	**...**	**PN**
A00	A10		An-10
A01	A11		An-11
A0n-1	A1n-1		An-1n-1
B00	B01		B0n-10
B10	B11		B1n-11
Bn-10	Bn-11		Bn-1n-1
C00	C01		C0n-10
C10	C11		C1n-11
Cn-10	Cn-11		Cn-1n-1

(k,i,j)			
P1	**P2**	**...**	**PN**
A00	A01		A0n-10
A10	A11		A1n-11
An-10	An-11		An-1n-1
B00	B10		BN-10
B01	B11		BN-11
B0N-1	B1N-1		BN-1N-1
C Matrix full	C Matrix full		C Matrix full
C Matrix full	C Matrix full		C Matrix full
C Matrix full	C Matrix full		C Matrix full

(k,j,i)			
P1	**P2**	**...**	**PN**
A00	A01		A0n-10
A10	A11		A1n-11
An-10	An-11		An-1n-1
B00	B10		BN-10
B01	B11		BN-11
B0N-1	B1N-1		BN-1N-1
C Matrix full	C Matrix full		C Matrix full
C Matrix full	C Matrix full		C Matrix full
C Matrix full	C Matrix full		C Matrix full

(i,j,k):

- Processor Pi will store ith row of matrix A and ith column of matrix B initially.
- In first cycle processor P1 will calculate C00, P2 will calculate C10 and so on.
- After each cycle, a processor can send it's B values to the processor on the left and do the further calculation. Hence, now P1 will calculate C01, P2 will calculate C11 and so on.
- Eventually after n cycles, P1 would have calculated 1^{st} row of result matrix C, P2 would have calculated 2^{nd} row of matrix C and so on.

(i,k,j):

- Processor Pi will store ith row of matrix A and ith row of matrix B initially.
- In the first cycle one processor will calculate the partial value for ith row of matrix C.

- After each cycle, a processor can send it's B values to the next processor and in the next cycle, the processor will calculate the remaining values of ith row of matrix C.
- After n cycles, each processor will complete the full values of ith row of matrix C.

(j,k,i):

- Processor Pj will store jth column of matrix A and jth column of matrix B initially.
- In the first cycle one processor will calculate the partial value for jth column of matrix C.
- After each cycle, a processor can send it's A values to the next processor and in the next cycle, the processor will calculate the remaining values of jth column of matrix C.
- Eventually, after n cycles, each processor will complete the full values of jth column of matrix C.

(j,i,k):

- Processor Pj will store jth row of matrix A and jth column of matrix B initially.
- In the first cycle one processor will calculate the first element for jth column of matrix C.
- After each cycle, a processor can send it's A values to the next processor and in the next cycle, the processor will calculate the second element of jth column of matrix C.
- Eventually, after n cycles, processor j will compute all the last element of jth column of matrix C i.e. after n cycles every processor will finish the computation of every columns of matrix C.

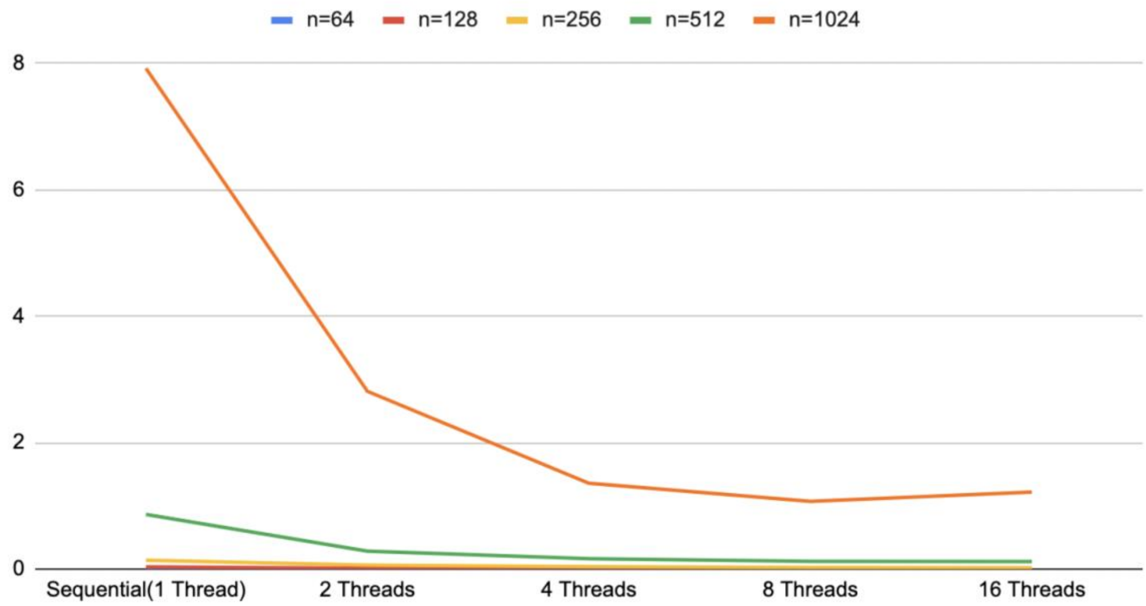
(k,i,j):

- Processor Pk will store kth column of matrix A and kth row of matrix B initially.
- In first cycle, the first processor will calculate the partial values of first row. At the same time the second processor will calculate the partial values of second row and so on.
- Eventually after n cycles, all processor will calculate complete values of all rows.

(k,j,i):

- Processor Pk will store kth column of matrix A and kth row of matrix B initially.
- In first cycle, the first processor will calculate the partial values of first column. At the same time the second processor will calculate the partial values of second column and so on.
- Eventually after n cycles, all processor will calculate complete values of all columns.

Matrix multiplication execution time analysis for multithreading:



From the above chart we can see that the execution time decreases as the number of threads increases until the number of threads is equal to the cores in the system. Here we have 8 cores in the system. So after crossing 8 threads execution time does not change or almost remains the same. The system on which we tested the code is MAC M1 Chip. Further in the above graph, we have not added the data of execution times for the size of matrixes(n=2048) in order to make above graph better visualize. Added the data for n=2048 in the below table.

Matrix Size(n)	Sequential(1 Thread)	2 Threads	4 Threads	8 Threads	16 Threads
64	0.00520800	0.00172000	0.00097700	0.00065500	0.00093300
128	0.03234000	0.01179700	0.00605600	0.00338300	0.00425300
256	0.13733700	0.05997200	0.03643500	0.02318100	0.01866400
512	0.86352600	0.28332700	0.16186400	0.12015100	0.11632900
1024	7.92134900	2.81303400	1.35747300	1.07100100	1.21837300
2048	133.32884200	45.3173030	22.78311600	20.44888100	20.62383000