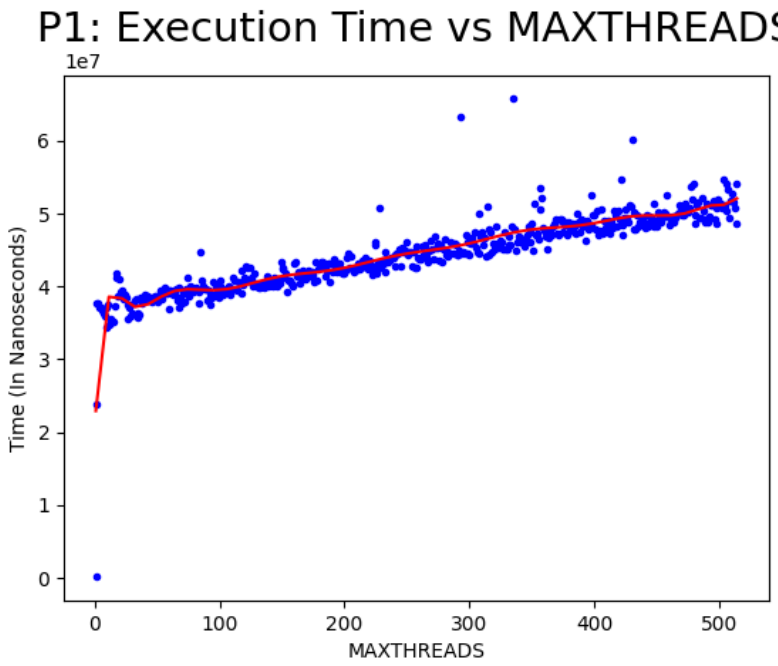
OPERATING SYSTEM
ASSIGNMENT 2
REPORT

Group Members:

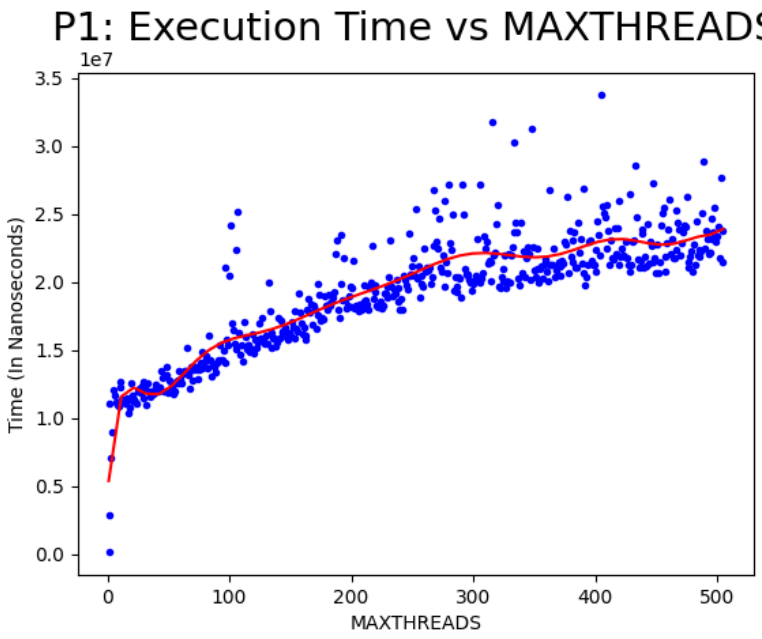
1. Khooshrin Aspi Pithawalla - 2020A7PS2067H
2. Ankesh Pandey - 2020A7PS0104H
3. Kavyanjali Agnihotri - 2020A7PS0185H
4. Tushar Brijesh Chenan - 2020A7PS0253H
5. Khushi Biyani - 2020A7PS0194H
6. Manik Arneja - 2020A7PS0119H

P1: Thread Count VS Time

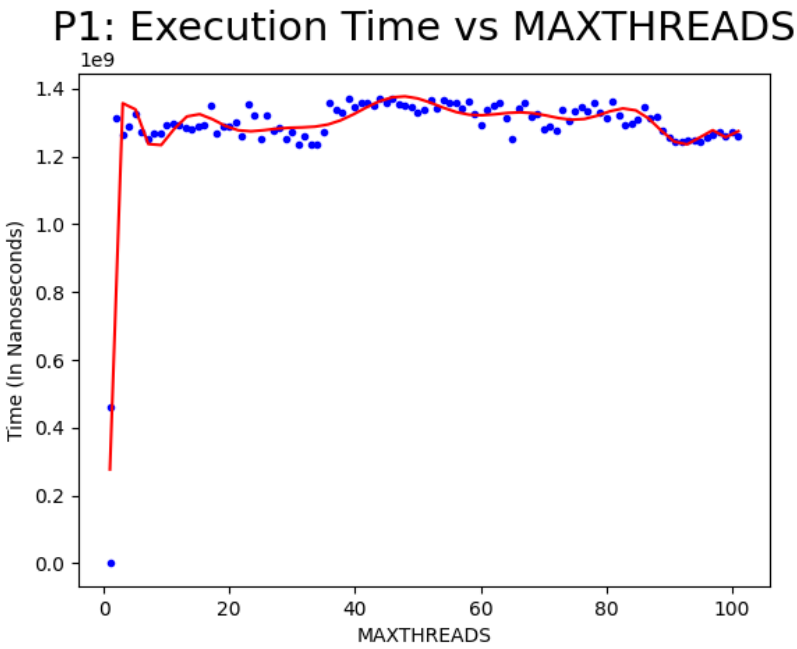
Case 1: Square Matrix Multiplication ($500 \times 500 \times 500$)



Case 2: Row by Column Multiplication: ($1000 \times 1 \times 1000$)



Case 3: Skewed Matrix Multiplication: $(100000 \times 67 \times 32)$

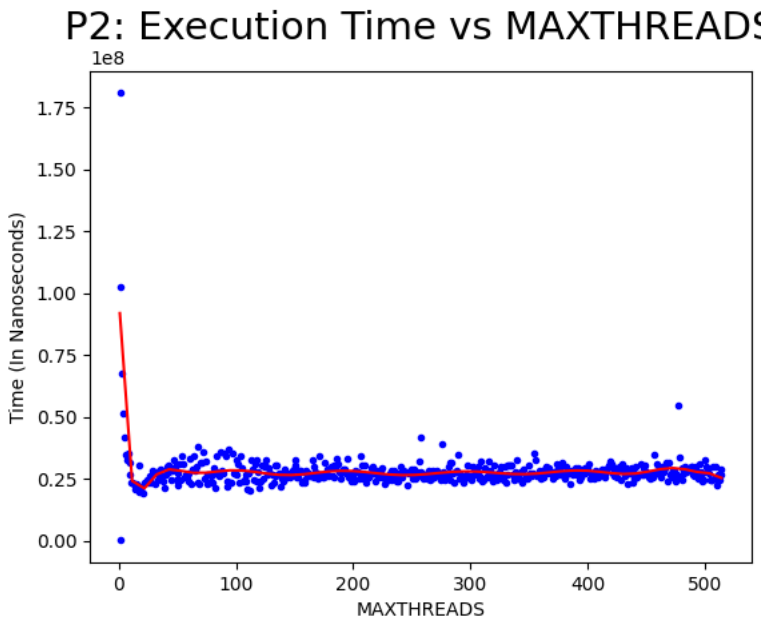


Analysis

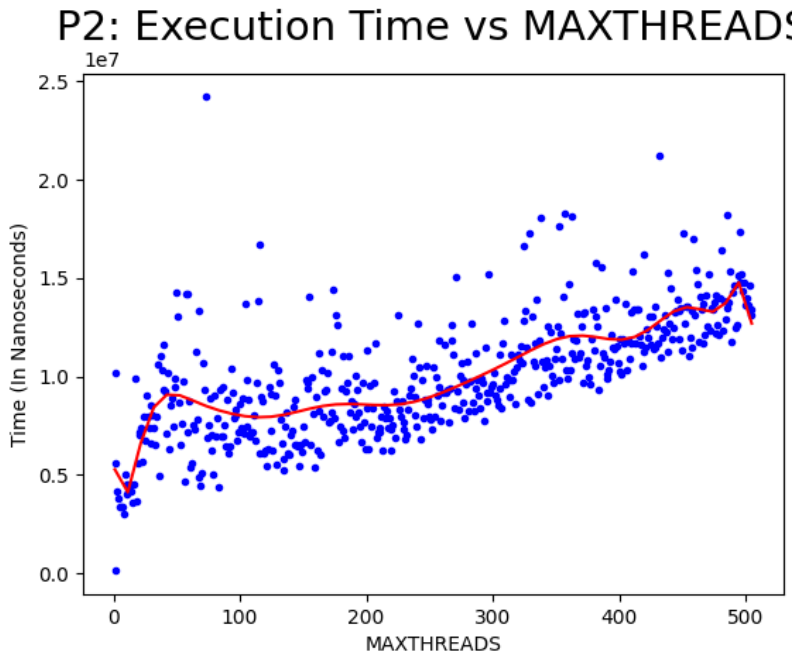
- It can be seen that there is a sharp increase in time taken when the number of threads goes up from 1 because of the context-switching overhead.
- This is also complemented by the fact that file I/O is constrained by the disk reading hardware which allows you to read only one sector at a time. Therefore, even with multiple threads, there is no actual speedup we achieve.
- The skewed matrix shows an uneven rise and fall in time because of the small dimensions (j and k) causing the processes to terminate before even all threads are created.
- Therefore, considering that an arbitrary large number of threads will cause the execution time of the program to increase (context switching dominates speedup from multi-threading), it makes sense to limit the number of threads to a small number.

P2: Thread Count VS Time

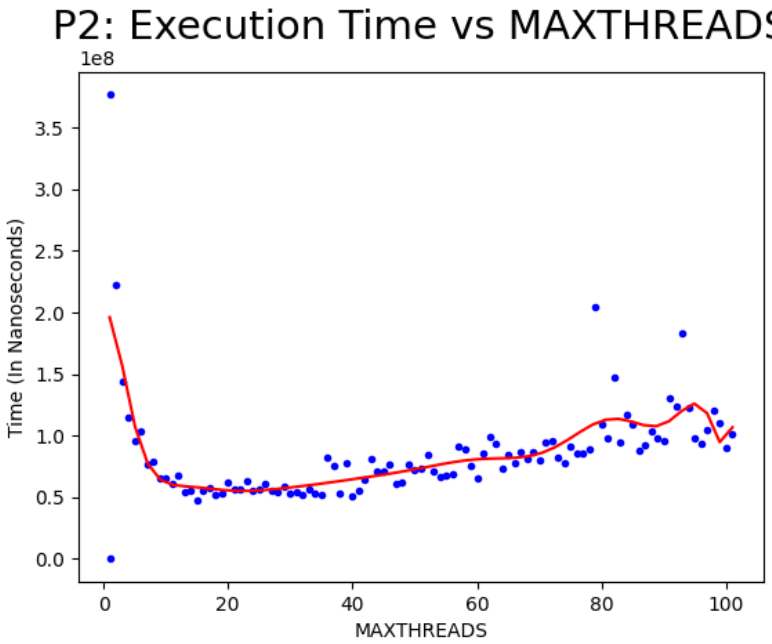
Case 1: Square Matrix Multiplication ($500 \times 500 \times 500$)



Case 2: Row by Column Multiplication: ($1000 \times 1 \times 1000$)



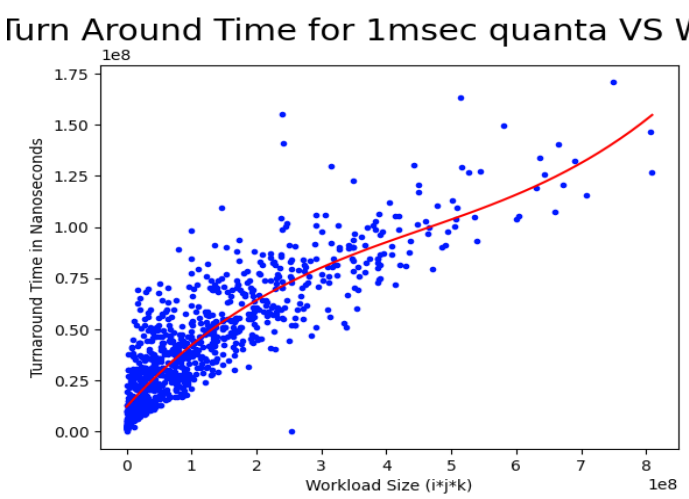
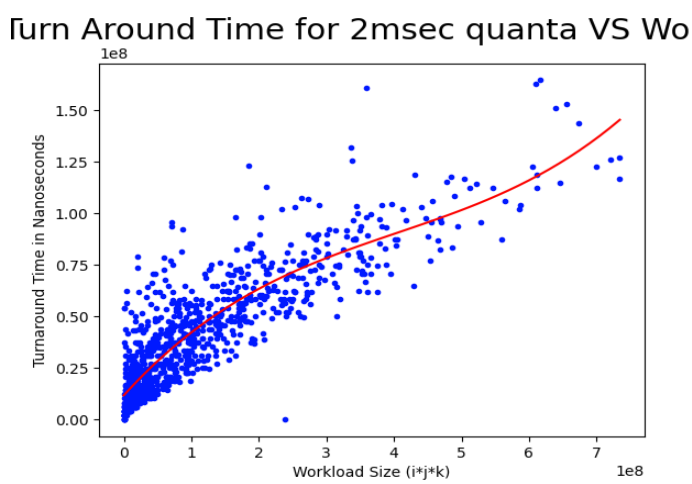
Case 3: Skewed Matrix Multiplication: $(100000 \times 67 \times 32)$



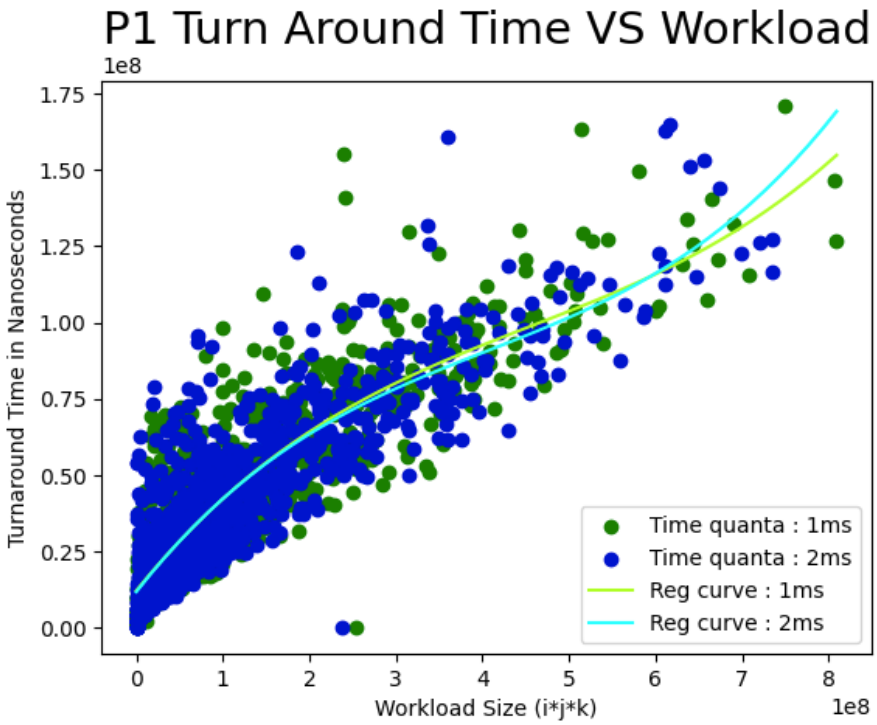
Analysis

- In all the plots, there is a small dip at the start, indicating we are achieving parallelism through use of multiple threads.
- For all matrices, the execution time becomes constant or increases after the dip, indicating that there is no clear benefit of increasing the number of threads after a certain point and in fact adds unnecessary overhead through context switching.

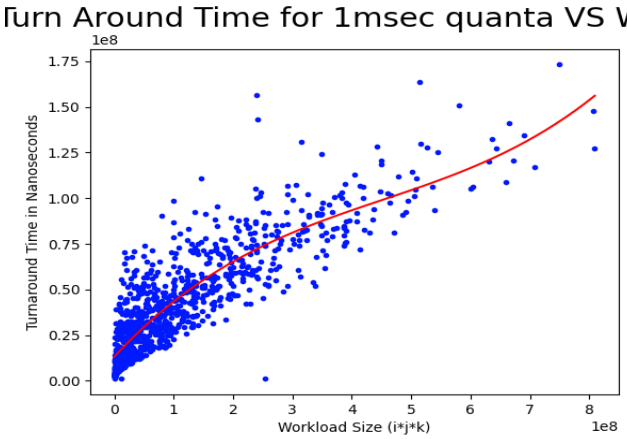
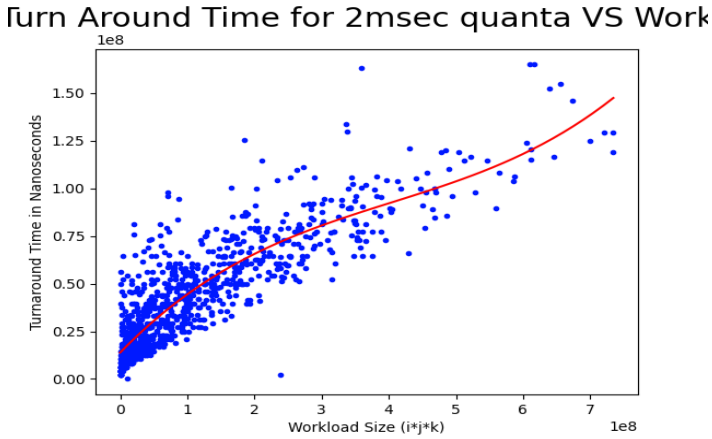
P1 : Plots for Turn Around Time VS Workload

Round Robin with Time Quanta = 1 msec	Round Robin with Time Quanta = 2 msec
	

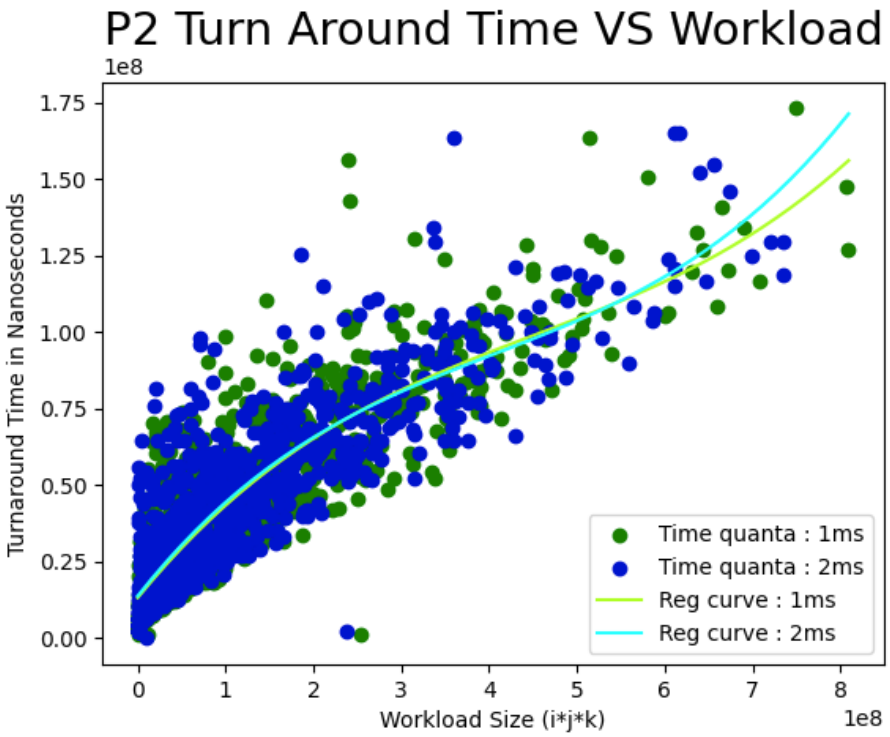
Comparative Plot for Turn Around Time VS Workload in P1 for both time quanta in Round Robin :



Plots for Turn Around Time VS Workload in P2

Round Robin with Time Quanta = 1 msec	Round Robin with Time Quanta = 2 msec
	

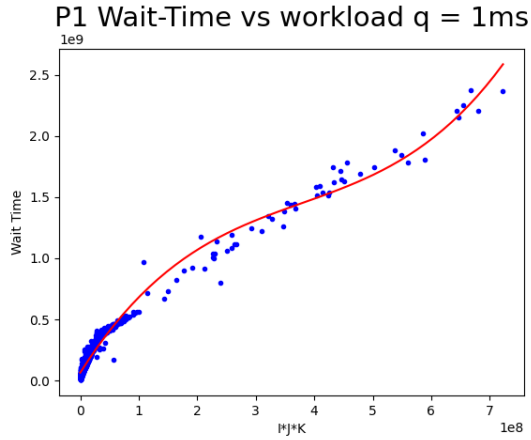
Comparative Plot for Turn Around Time VS Workload in P2 for both time quanta in Round Robin :

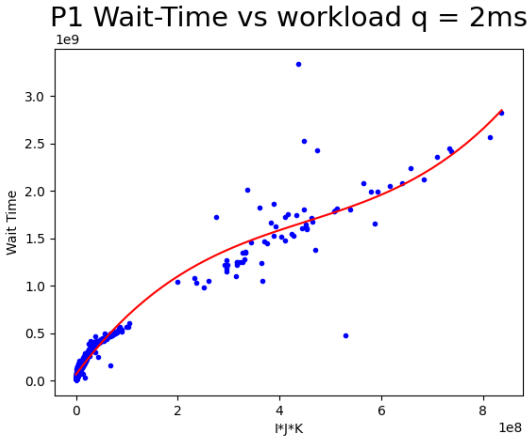


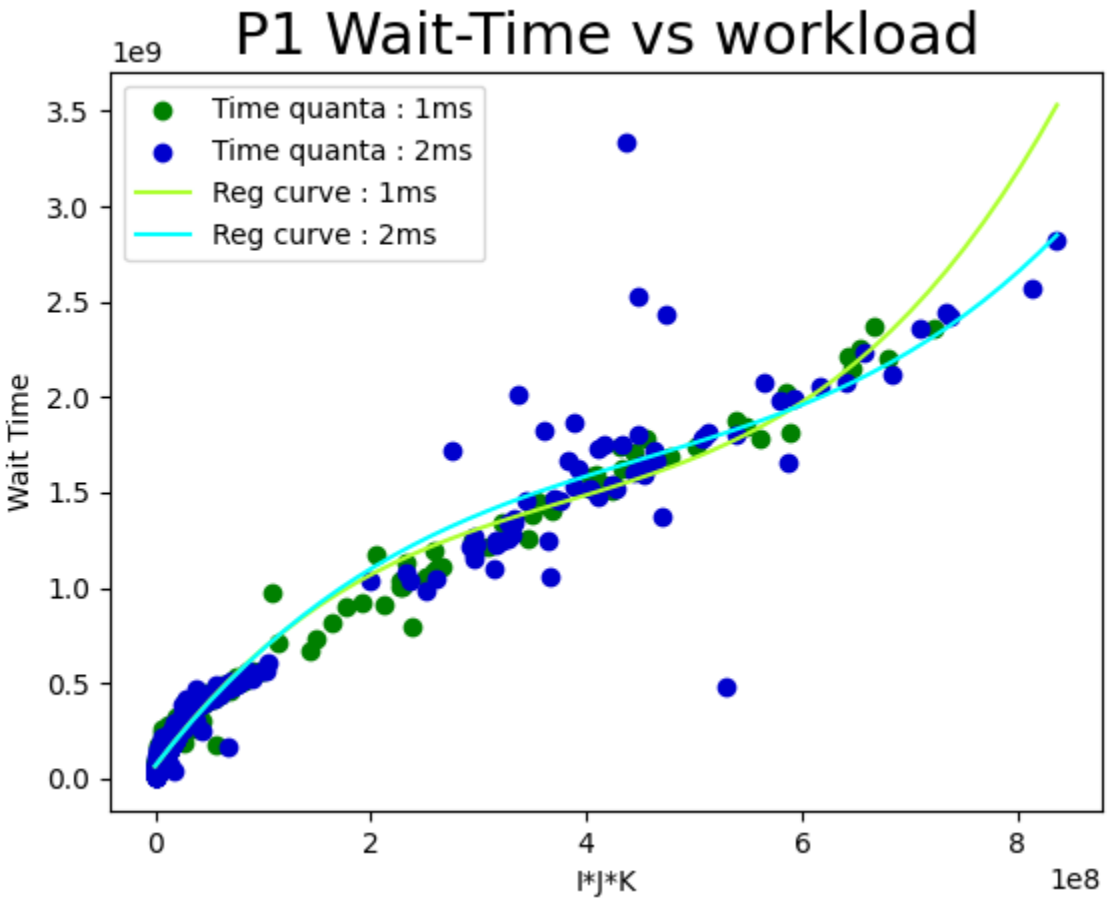
Analysis

- Turnaround Time is defined as the total time that is taken for the process to complete its execution since it enters into the ready queue.
- As we can see from the above diagrams, as the workload increases, the turnaround time increases in both processes for both the Round Robin time quanta. This is because with the increasing workload, P1 has to read a larger number of matrix elements and P2 has to compute product values for more number of elements, thereby consuming more time leading to an overall increase in turnaround time in both cases with increase in workload.
- However, as a comparison between the turnaround time for Round Robin with time quanta = 1 msec and for Round Robin with time quanta = 2 msec, we can observe that the turnaround time increases with decreasing Round Robin time quanta due to the increase in the number of context switches.
- Comparing the turnaround time for P1 and P2, we can see that the turnaround time for P1 is lesser than the turnaround time for P2 as P2 performs more computations (matrix multiplication and file write) during its execution while P1 performs less computations(file read).

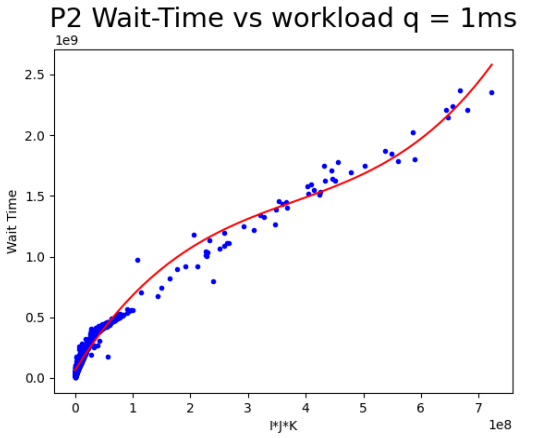
P1: Waiting Time vs Workload

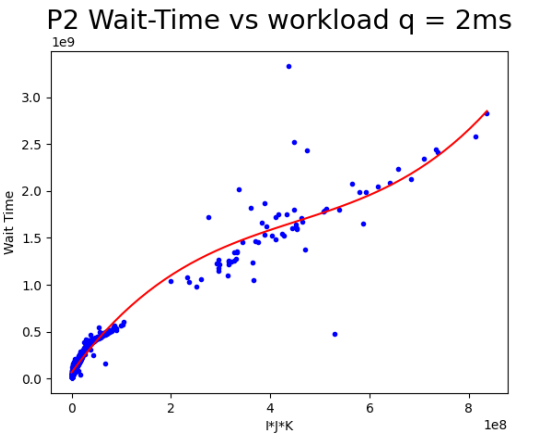


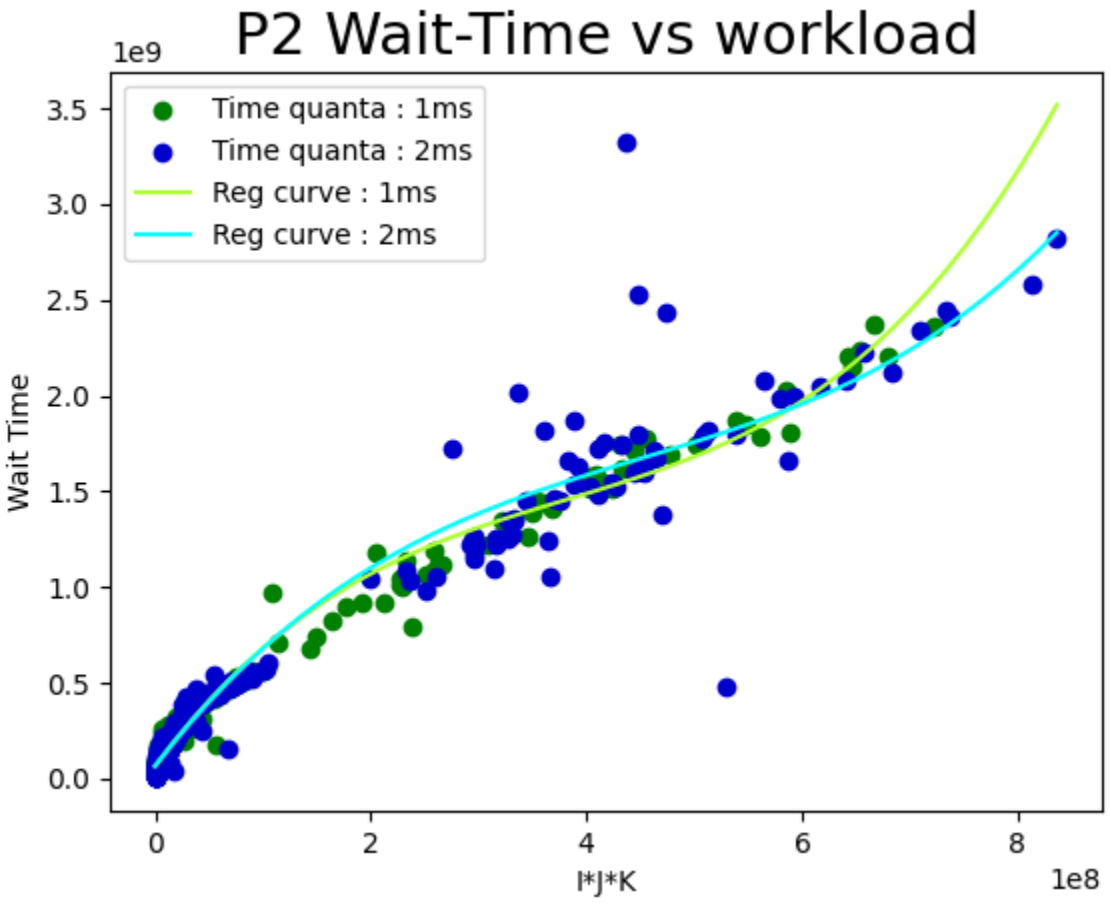




P2: Waiting Time vs Workload







Analysis of workload and waiting time

- As we can clearly see with increase in workload we require more time quanta which results in more waiting time for completion hence we get an increasing trend of waiting time as the workload increases.
- With increasing workload we know that the total time taken by P1 and P2 increases accordingly, therefore more time quanta required for both P1 and P2 to get executed completely.
- In a round robin scheduler when one process is executed all others are in waiting state. So in our case we get the increasing trend of waiting time vs workload.
- Since we have 2 processes P1 and P2 being scheduled in a round robin manner and at a given time quanta, one is running and other is in waiting state, and similarly vice versa thus we get similar wait time across P1 and P2 for both the time quantum (namely q=1ms, q-2ms).

Context Switching Time

Time Quanta : 2ms

contextSwitch2ms.csv

```
Matrix Sizes 500 X 500 X 500
Total Context Switch Time: 1814741 ns
Total Context Switch Count: 229
Average Context Switch Time: 7924.633188 ns

Matrix Sizes 100000 X 67 X 32
Total Context Switch Time: 16771401 ns
Total Context Switch Count: 1938
Average Context Switch Time: 8653.973684 ns

Matrix Sizes 500 X 1 X 67
Total Context Switch Time: 70859 ns
Total Context Switch Count: 9
Average Context Switch Time: 7873.222222 ns

Matrix Sizes 400 X 400 X 400
Total Context Switch Time: 1007276 ns
Total Context Switch Count: 134
Average Context Switch Time: 7516.985075 ns

Matrix Sizes 100 X 100 X 100
Total Context Switch Time: 63609 ns
Total Context Switch Count: 8
Average Context Switch Time: 7951.125000 ns
```

Time Quanta : 1ms

contextSwitch1ms.csv

```
Matrix Sizes 500 X 500 X 500
Total Context Switch Time: 3517334 ns
Total Context Switch Count: 470
Average Context Switch Time: 7483.689362 ns

Matrix Sizes 100000 X 67 X 32
Total Context Switch Time: 32151928 ns
Total Context Switch Count: 3899
Average Context Switch Time: 8246.198512 ns

Matrix Sizes 500 X 1 X 67
Total Context Switch Time: 125584 ns
Total Context Switch Count: 16
Average Context Switch Time: 7849.000000 ns

Matrix Sizes 400 X 400 X 400
Total Context Switch Time: 1913642 ns
Total Context Switch Count: 274
Average Context Switch Time: 6984.094891 ns

Matrix Sizes 100 X 100 X 100
Total Context Switch Time: 113713 ns
Total Context Switch Count: 14
Average Context Switch Time: 8122.357143 ns
```

Analysis

- It is observed that average context switching time doesn't depend on the size of the matrix (that is, I , J and K).
- We get nearly similar values of average context switch time over the course of a skewed matrix, and a perfectly square matrix.
- This is evident with the fact that context switching largely depends on scheduler running time which depends on processor's memory speed and interrupt handling which does not depend on the size of time quanta. Slight variations are seen because of the background processes running on the system.
- Variations are minimized when analyzing matrices of similar workload.