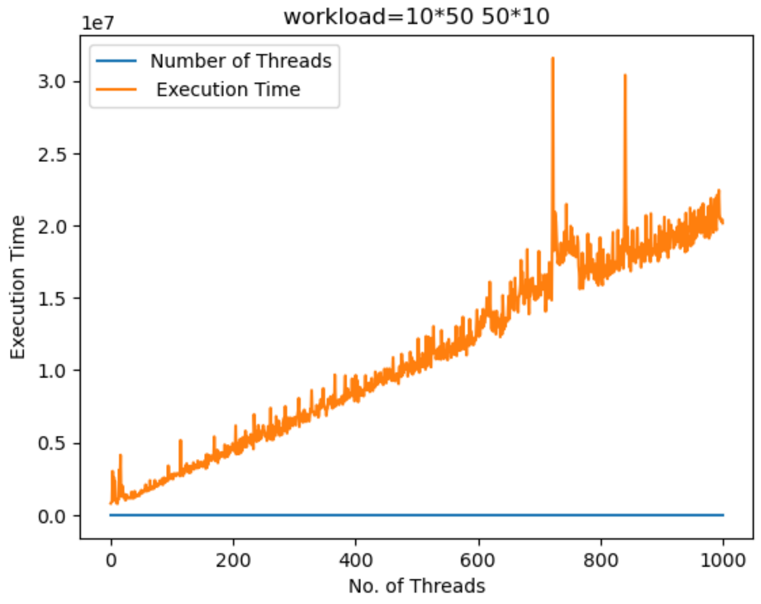
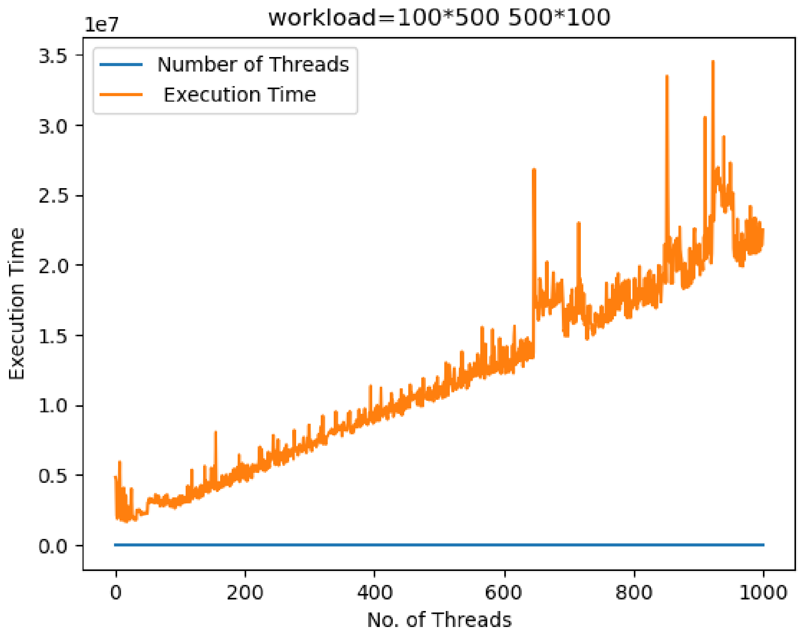
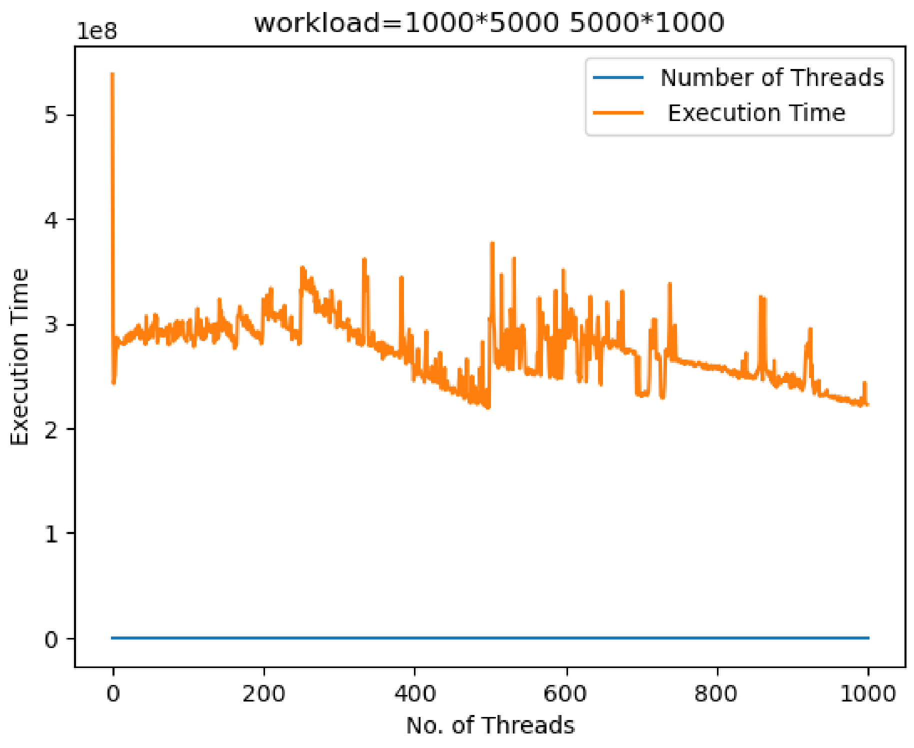
OS Assignment 2

P1

P1 is responsible for reading the two input matrices using threads and writing them to the shared memory.







OBSERVATIONS

1. The execution time for fairly small inputs increases as we add more threads. This is because a low number of threads can read the input sufficiently fast, increasing the number of threads just causes the execution time to inflate because of all the extra calculations, context switches and thread creations the program has to do.

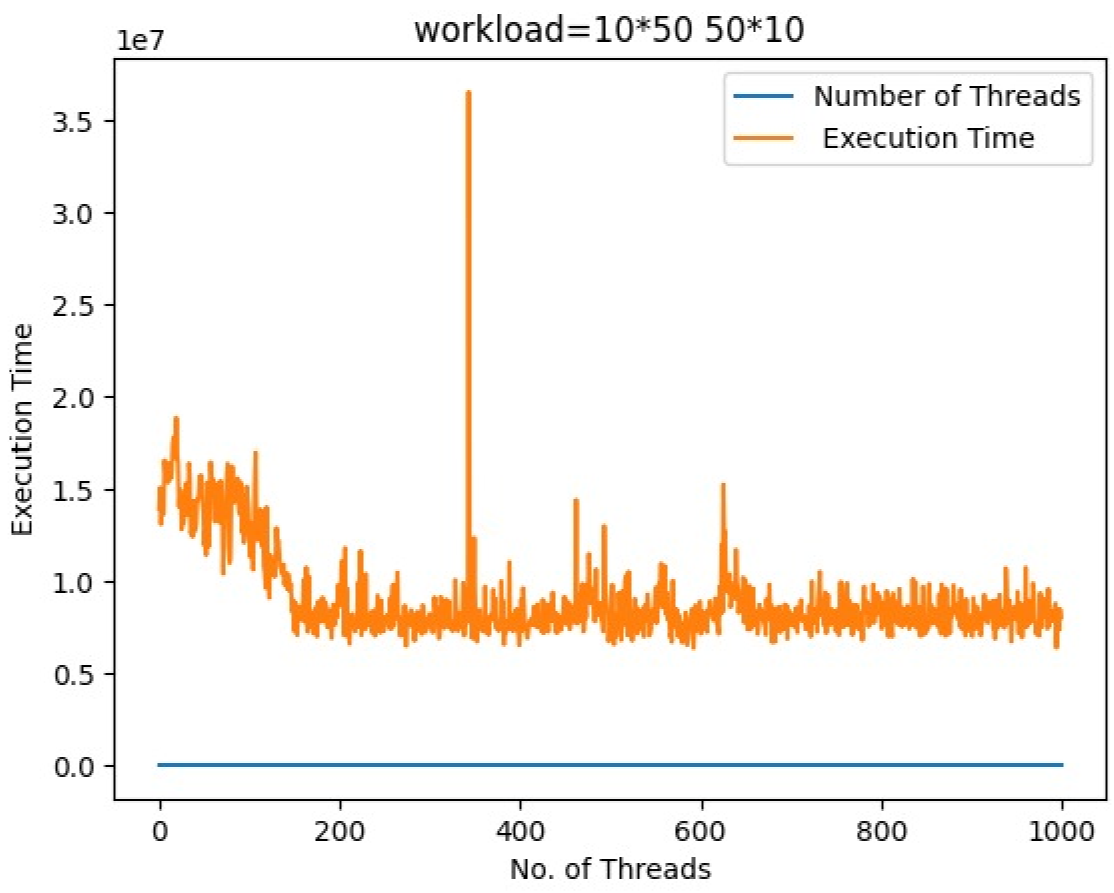
2. With a very large input (like the one taken for the third graph), a small number of threads cannot read the given input fast enough. Thus we see a decrease in the execution time as we increase the number of threads up until a point, after which increasing the number of threads does not really result in a decrease in execution time because of hardware limitations and the execution time remains more or less the same after that(slightly increasing due to overheads).

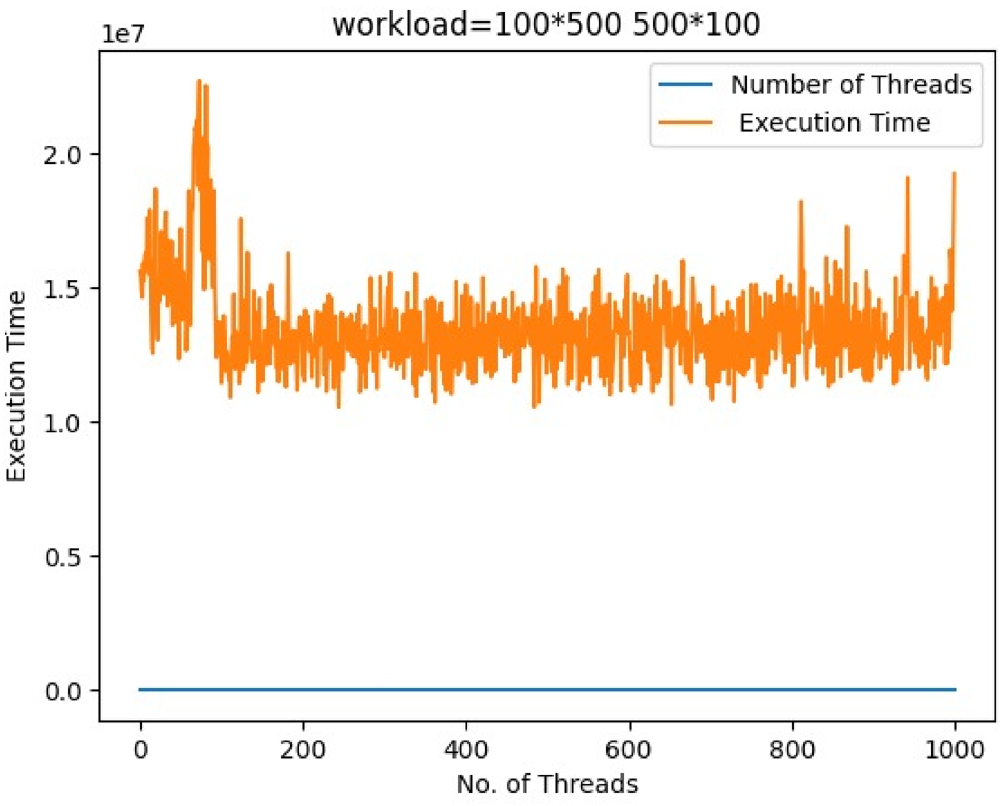
P2

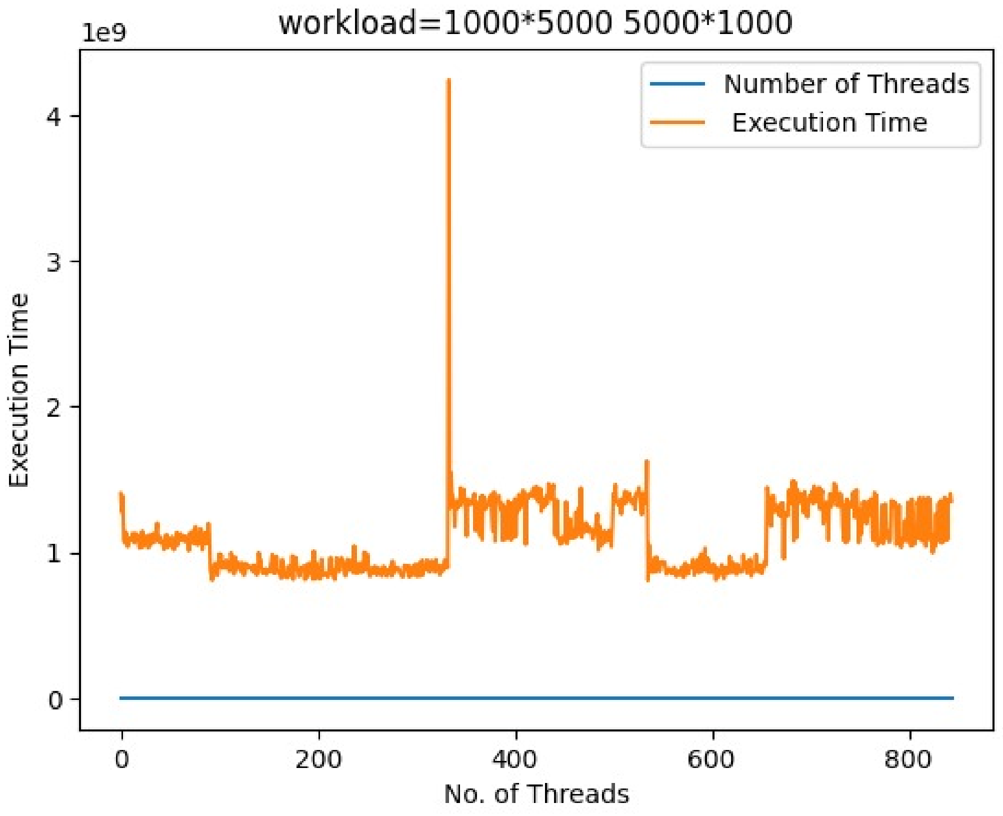
P2 accounts for reading the matrices from the shared memory, calculating the product matrix using threads and writing the result to the output file.

With multiple threads, we assume that each of them takes up an equal number of rows and the last thread takes up the remaining rows, if any.

We used flags to keep track of the rows and columns available to multiply and note the entries yet to be calculated.




workload=100*500 500*100
1e7
Number of Threads
Execution Time
Execution Time
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000
No. of Threads


workload=1000*5000 5000*1000
1e9
Number of Threads
Execution Time
Execution Time
4
3
2
1
0
0
200
400
600
800
No. of Threads

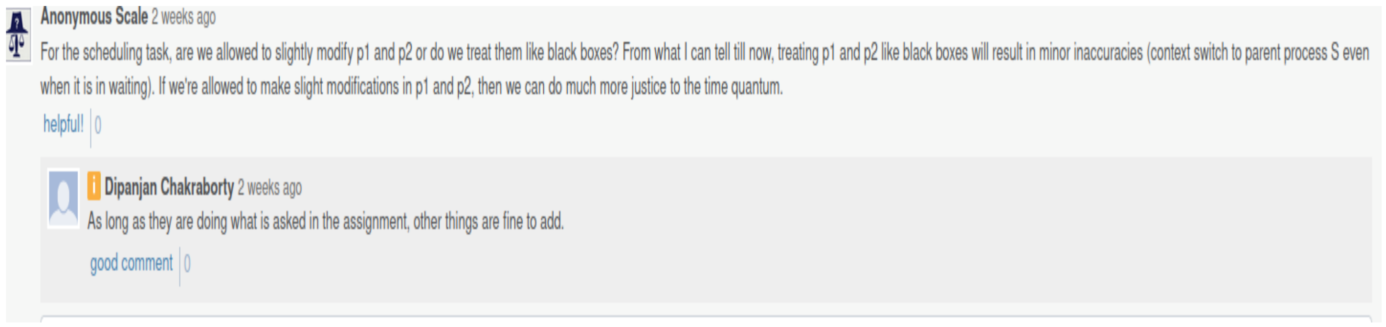
OBSERVATIONS

1. Execution time decreases with the increase in the number of threads up until a point, after which it remains practically constant.

2. The graphs demonstrate that creating more threads beyond a point does not yield a significant change in execution time. There's only additional overhead that can occur owing to context switches.

SCHEDULER

We create user defined signal handlers inside P1 and P2 to increase precision of context switches between processes. The default signal handlers were created to work well with the standard OS Scheduler, so creating new signal handlers for a new scheduler is justified.



Anonymous Scale 2 weeks ago
For the scheduling task, are we allowed to slightly modify p1 and p2 or do we treat them like black boxes? From what I can tell till now, treating p1 and p2 like black boxes will result in minor inaccuracies (context switch to parent process S even when it is in waiting). If we're allowed to make slight modifications in p1 and p2, then we can do much more justice to the time quantum.
helpful! | 0

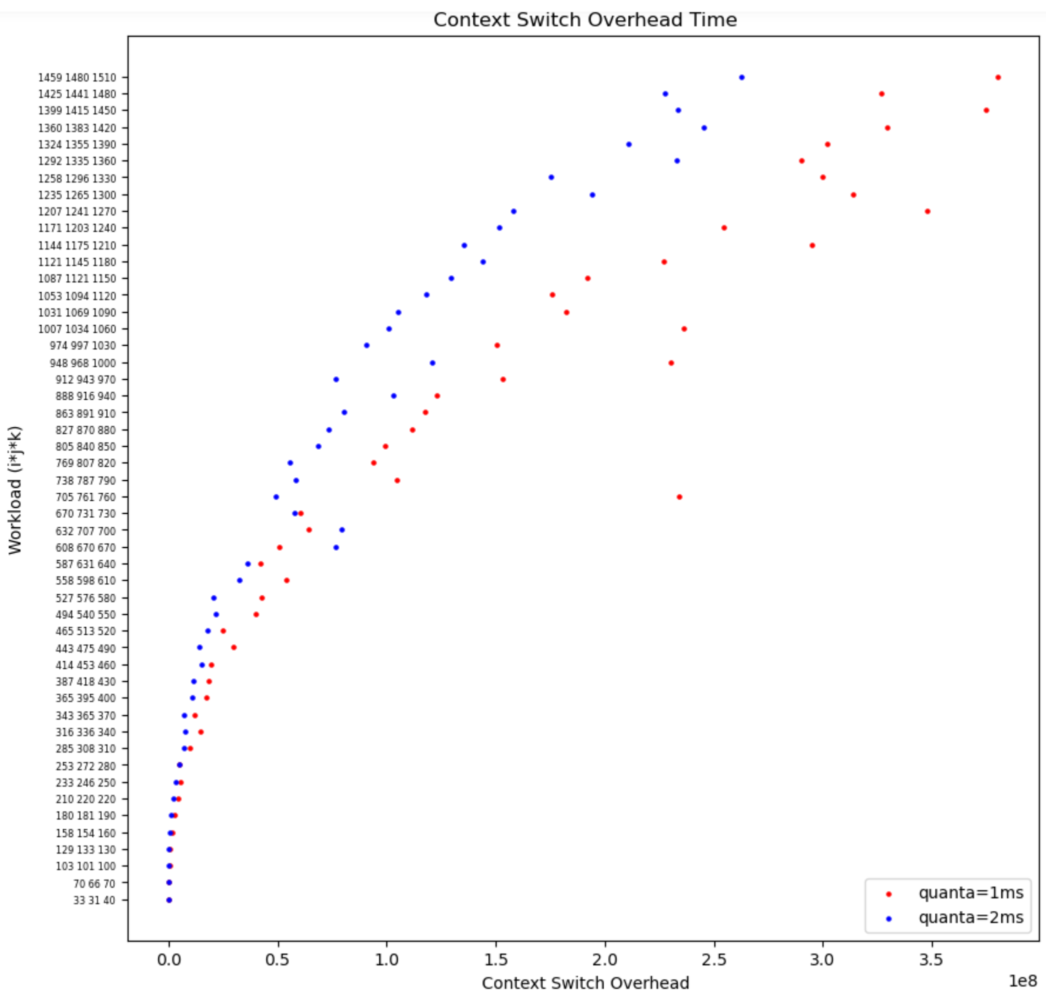
Dipanjan Chakraborty 2 weeks ago
As long as they are doing what is asked in the assignment, other things are fine to add.
good comment | 0

We employ round-robin scheduling to equally divide CPU time between P1 and P2. The results below are for two different quanta, 1ms and 2ms.

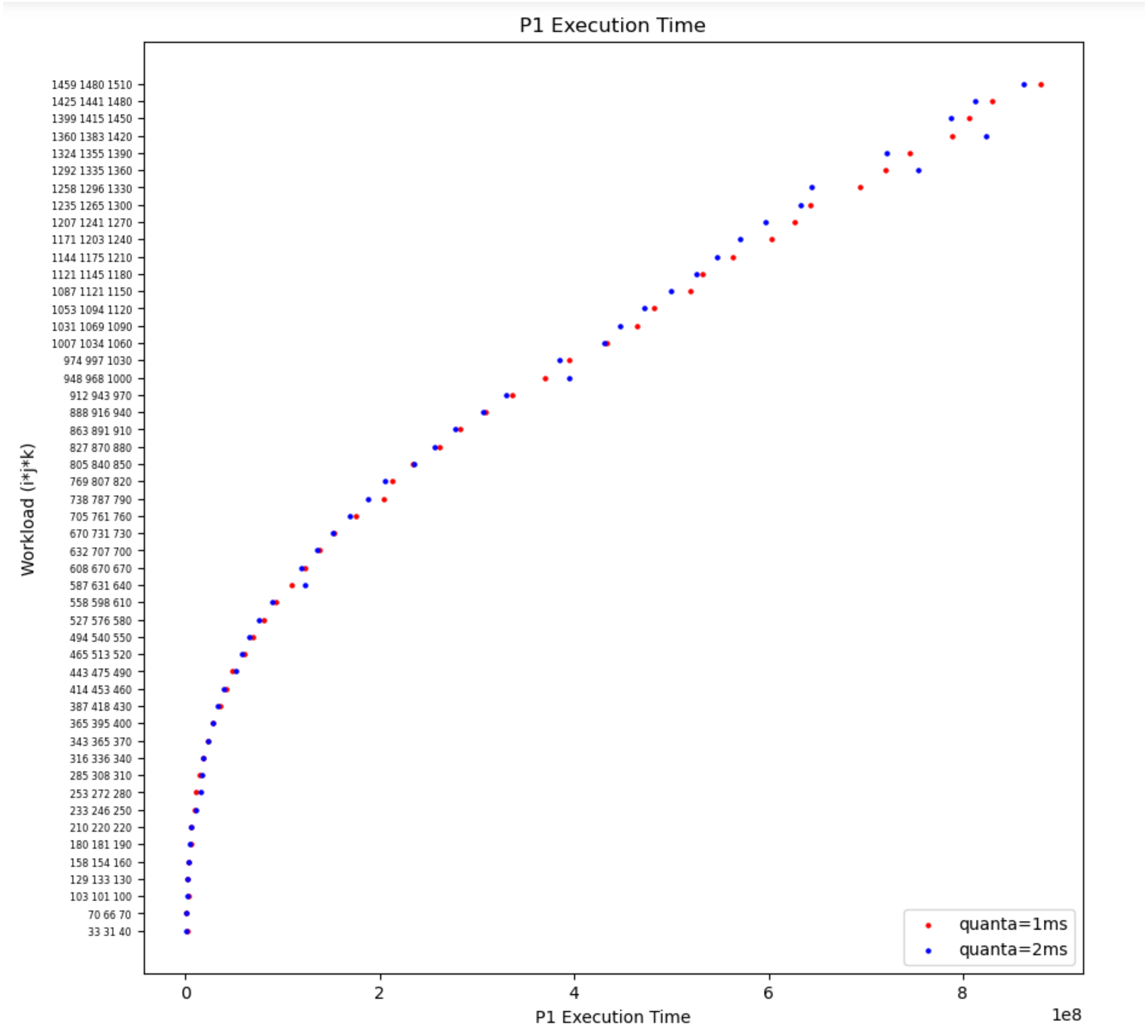
OBSERVATIONS

P1

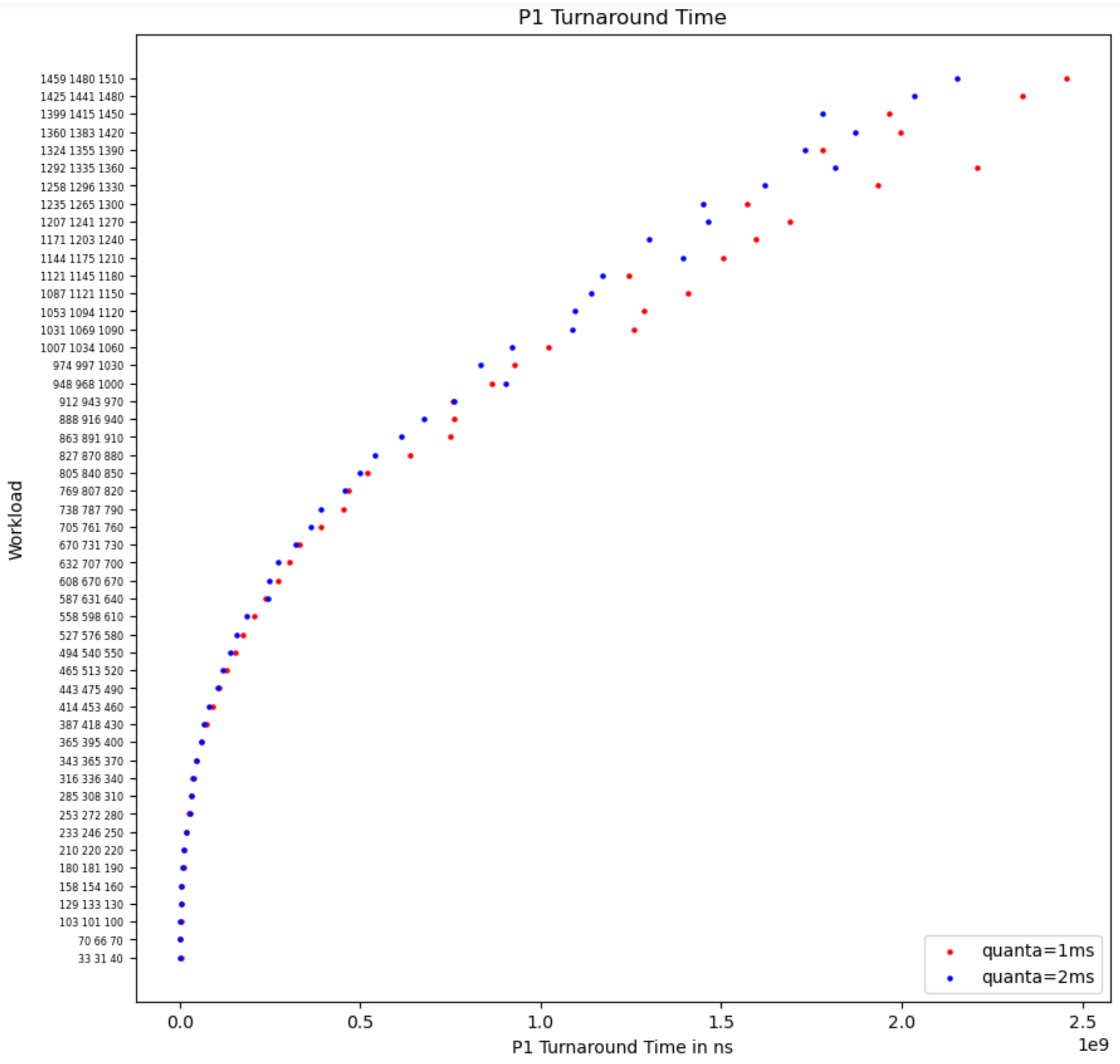
- When decreasing time quanta for the round robin scheduling, it is evident from the graph that the time taken for context switches increases, which is expected.



- It can be seen that there is not much difference in the execution time for P1 even though the time quanta changed from 1ms to 2ms. This can be explained on the basis that the amount of work that P1 has to do remains the same even though time quanta has changed and hence we get the graph below.

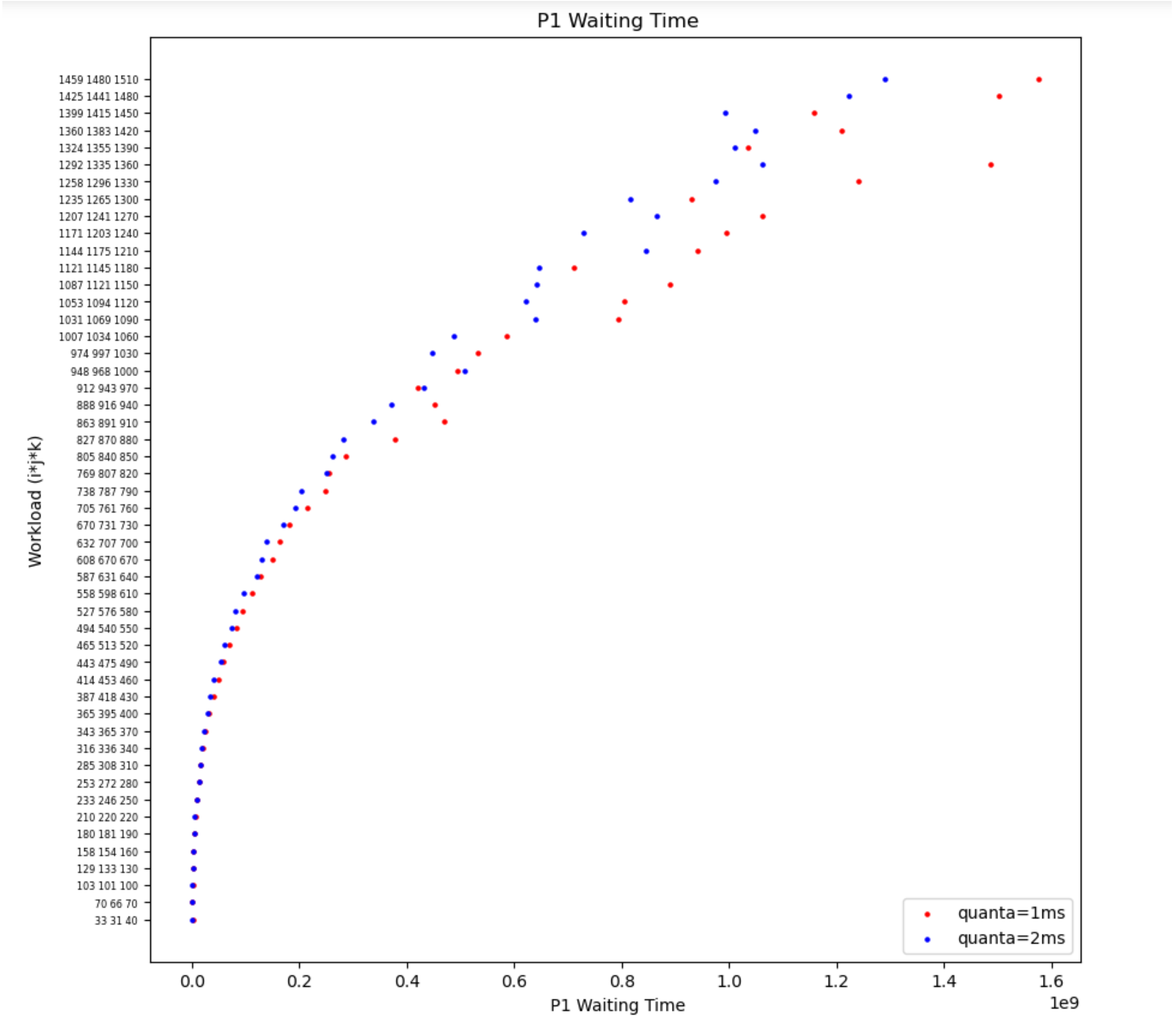


- The turnaround time for P1 increases for smaller time quanta as the context switch time increases when time quanta is smaller. This behavior can be confirmed from the graph below.



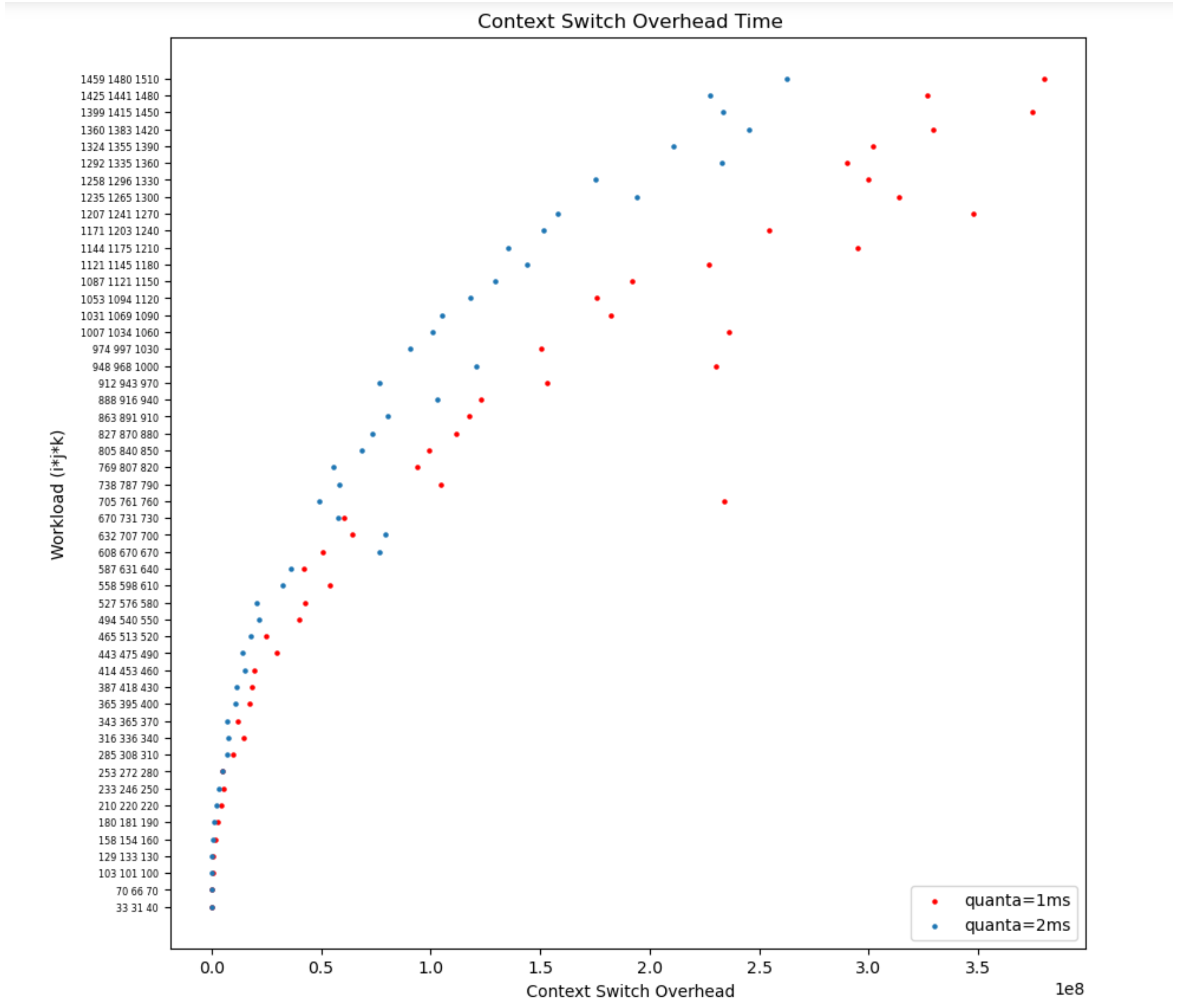
- Waiting time can be defined as Turnaround time - Execution time. Now from our above inferences we know that the Execution time is more or less similar when the time quanta is 1ms and 2ms so we must take into consideration the Turnaround time. Turnaround time is lesser when the time quanta is 2ms and hence we can conclude that the waiting time is lesser when the time quanta is 2ms compared to when time quanta is 1ms.

This is also evident from the graph below.

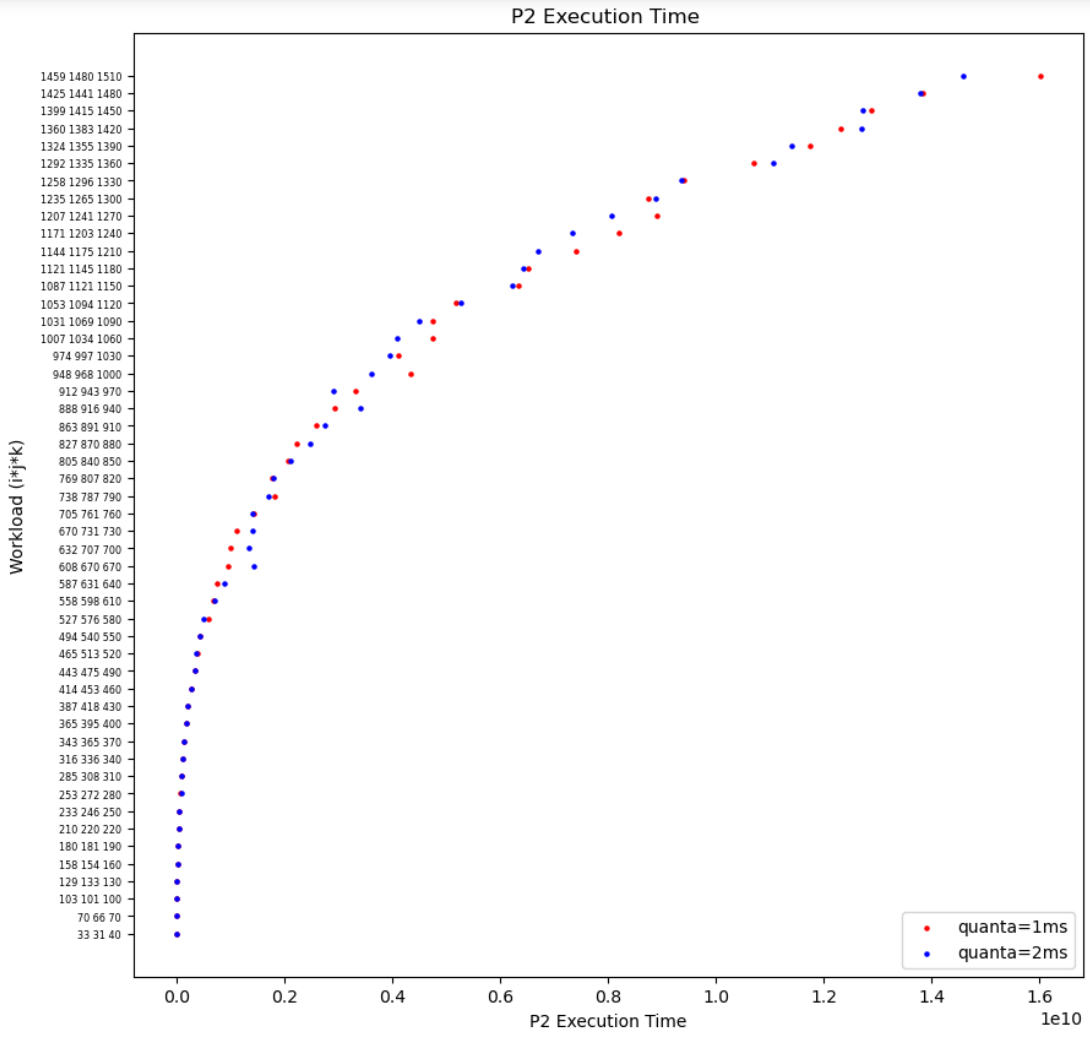


P2

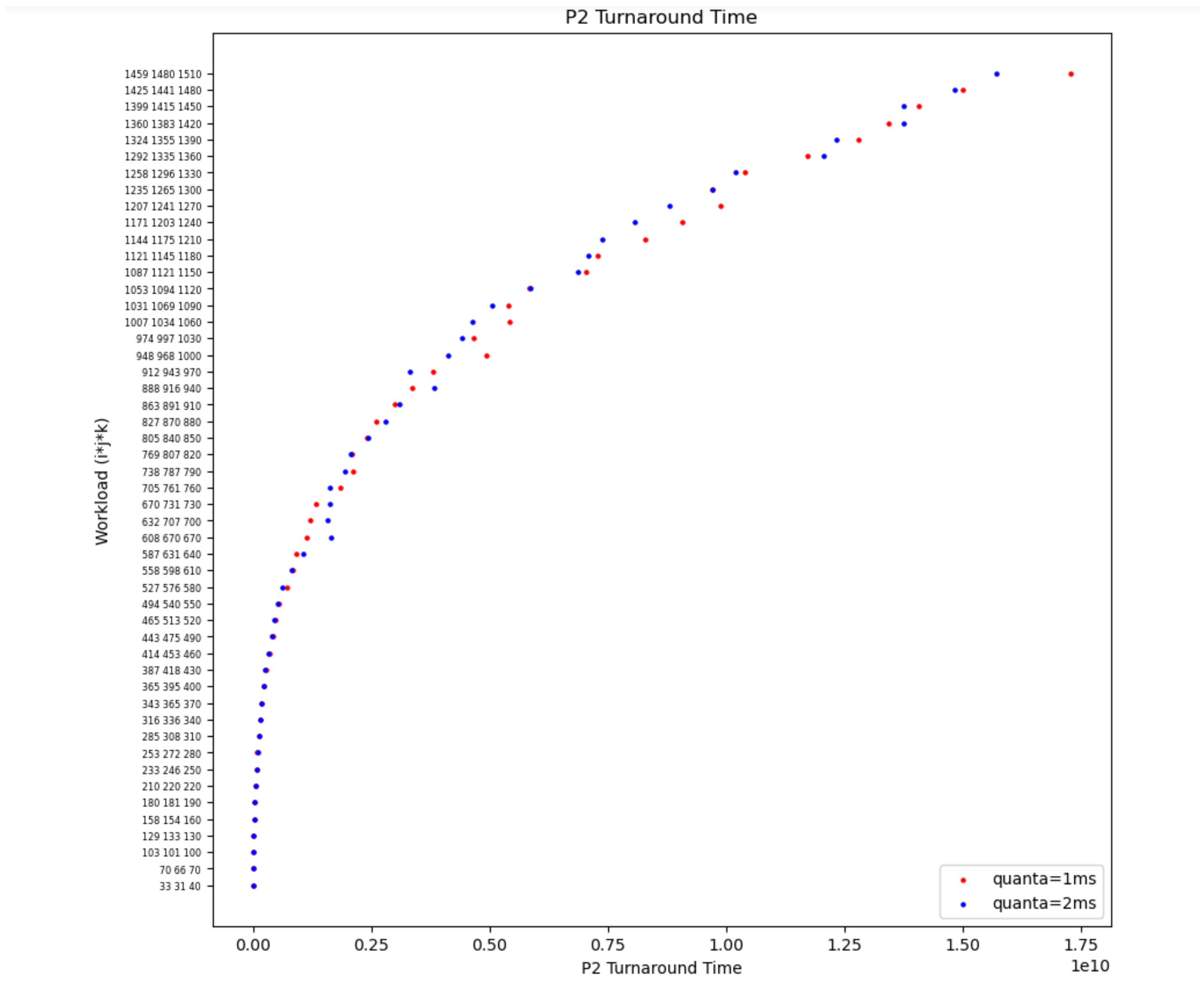
- When decreasing time quanta for the round robin scheduling, it is evident from the graph that the time taken for context switches increases, which is expected. We got the same result for P1.



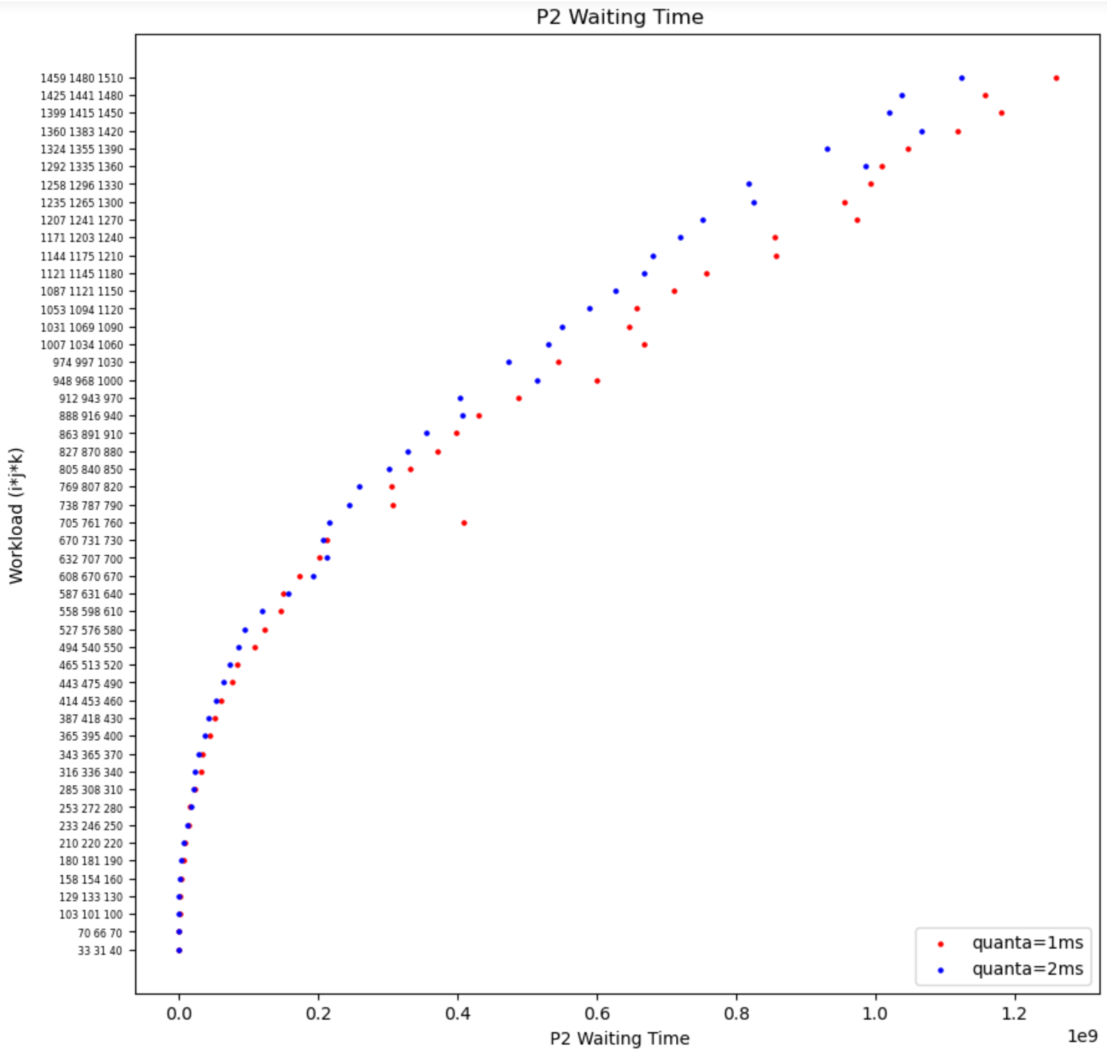
- It can be seen that there is not much difference in the execution time for P2 even though the time quanta changed from 1ms to 2ms. This can be explained on the basis that the amount of work that P2 has to do remains the same even though time quanta has changed and hence we get the graph below. This is very similar to the reasoning which was given for P1



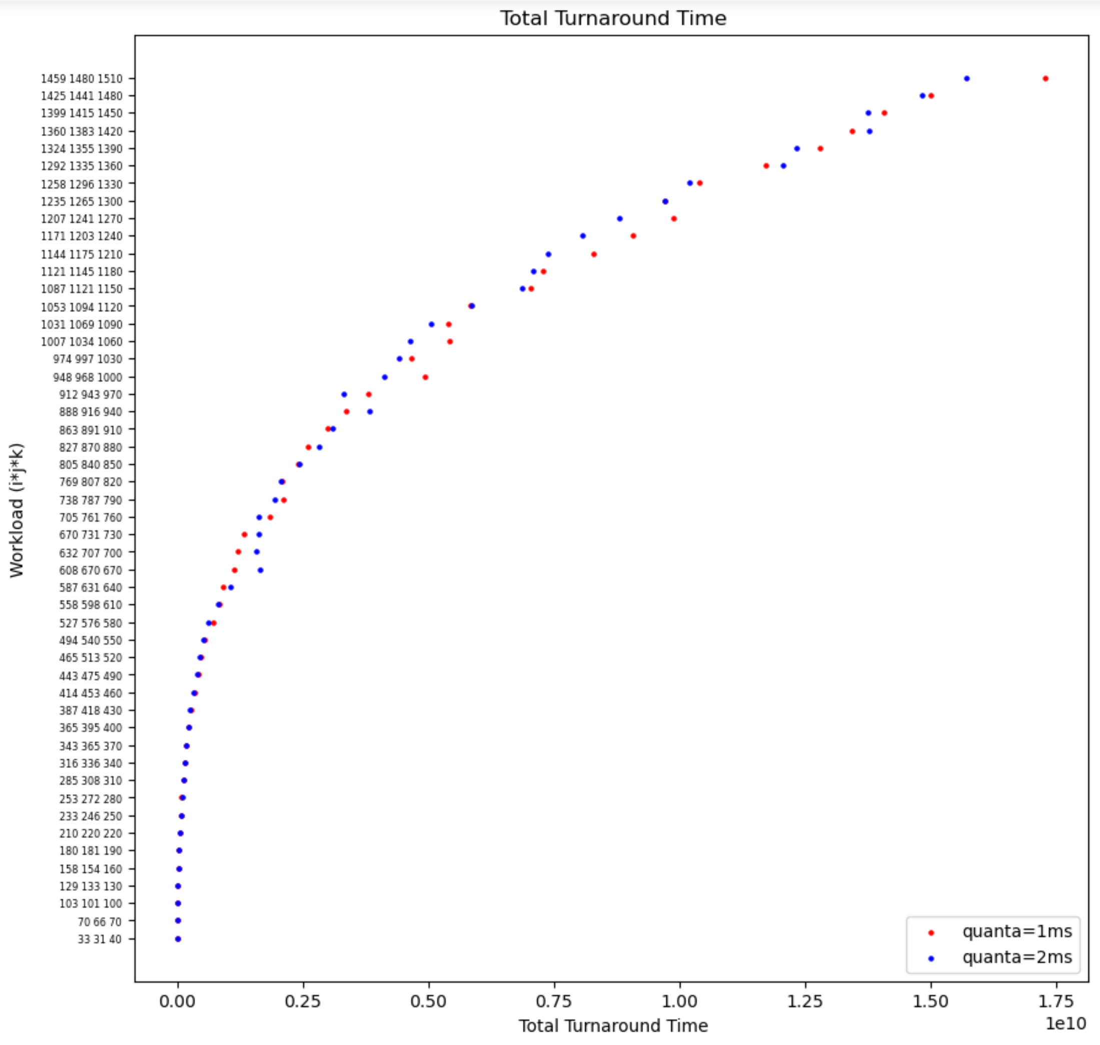
- The turnaround time for P2 increases for smaller time quanta as the context switch time increases when time quanta is smaller. This behavior can be confirmed from the graph below. Compared to P1, this difference is negligible since Turnaround time for P2 is in order of seconds, and context switches are very small.



- Waiting time can be defined as Turnaround time - Execution time. Now from our above inferences we know that the Execution time is more or less similar when the time quanta is 1ms and 2ms so we must take into consideration the Turnaround time. Turnaround time is lesser when the time quanta is 2ms and hence we can conclude that the waiting time is lesser when the time quanta is 2ms compared to when time quanta is 1ms. This is also evident from the graph below.


P2 Waiting Time
Workload (i*j*k)
1459 1480 1510
1425 1441 1480
1399 1415 1450
1360 1383 1420
1324 1355 1390
1292 1335 1360
1258 1296 1330
1235 1265 1300
1207 1241 1270
1171 1203 1240
1144 1175 1210
1121 1145 1180
1087 1121 1150
1053 1094 1120
1031 1069 1090
1007 1034 1060
974 997 1030
948 968 1000
912 943 970
888 916 940
863 891 910
827 870 880
805 840 850
769 807 820
738 787 790
705 761 760
670 731 730
632 707 700
608 670 670
587 631 640
558 598 610
527 576 580
494 540 550
465 513 520
443 475 490
414 453 460
387 418 430
365 395 400
343 365 370
316 336 340
285 308 310
253 272 280
233 246 250
210 220 220
180 181 190
158 154 160
129 133 130
103 101 100
70 66 70
33 31 40
quanta=1ms
quanta=2ms
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e9
P2 Waiting Time

- The total turnaround time when q=1ms is more as compared to q=2ms because of the context switches. But since the turnaround time is in order of seconds, this difference is very small.


Total Turnaround Time
Workload (i*j*k)
1e10
Total Turnaround Time
quanta=1ms
quanta=2ms