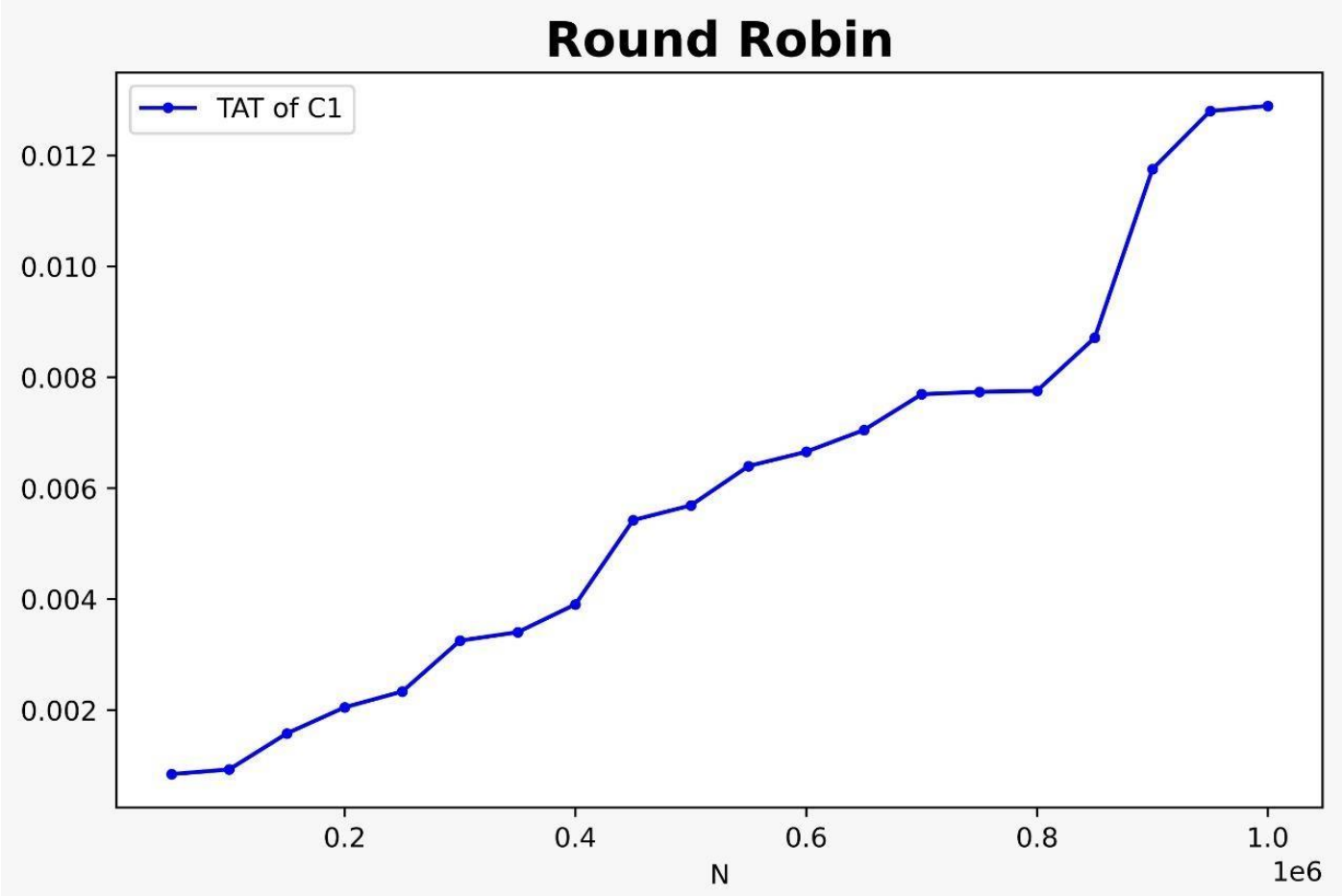
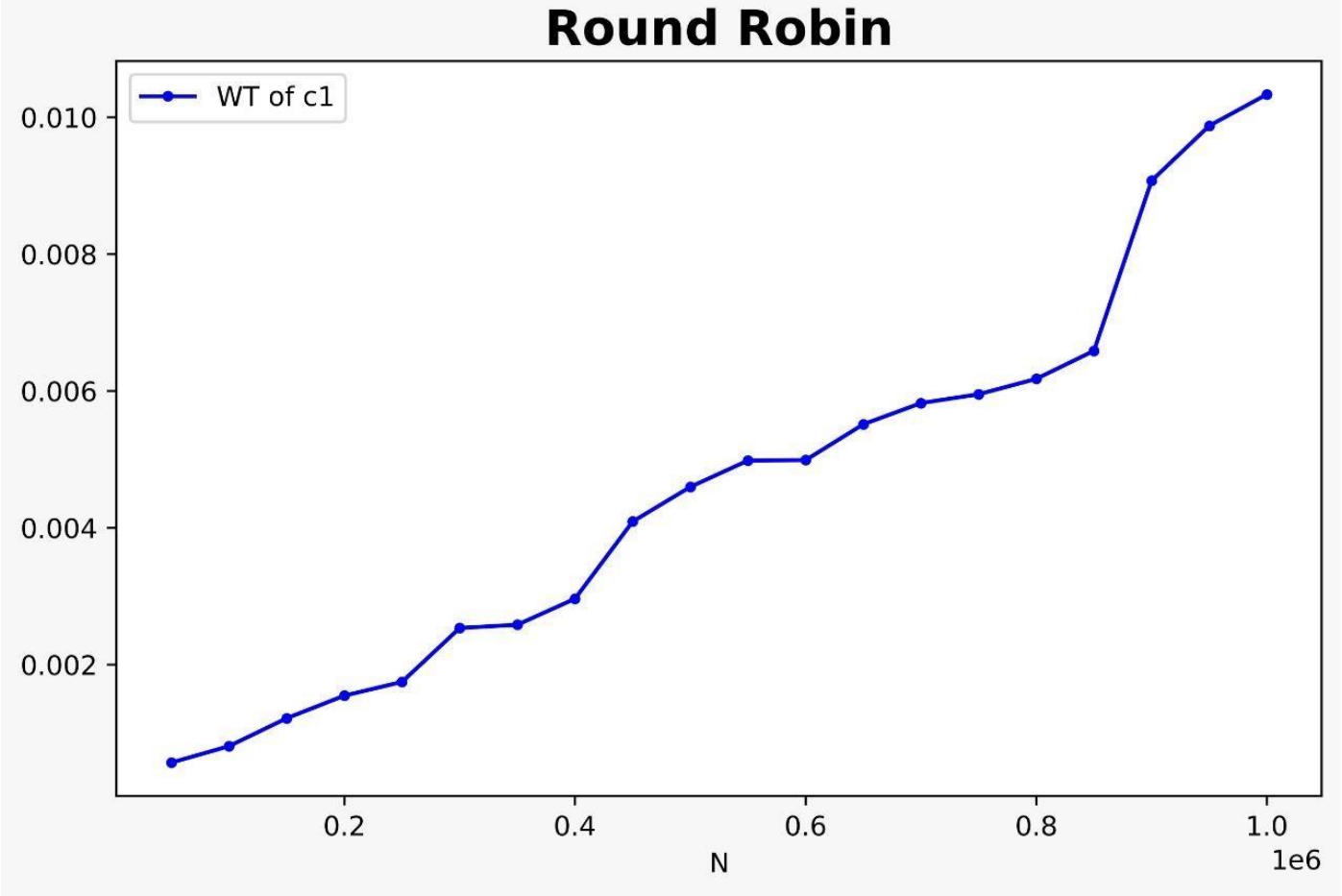
Graphs of Time vs Workload

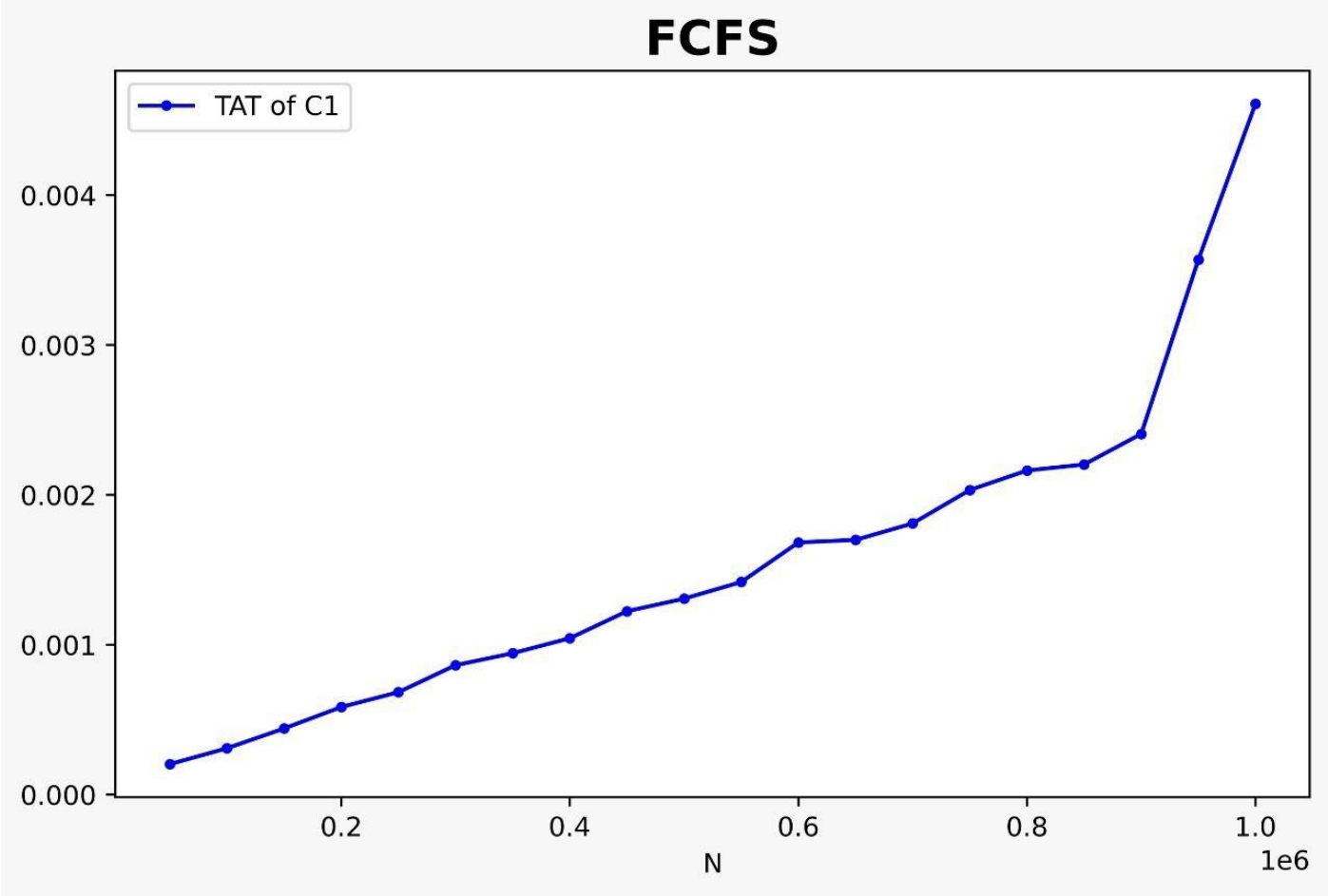
TAT, WT of C1:-

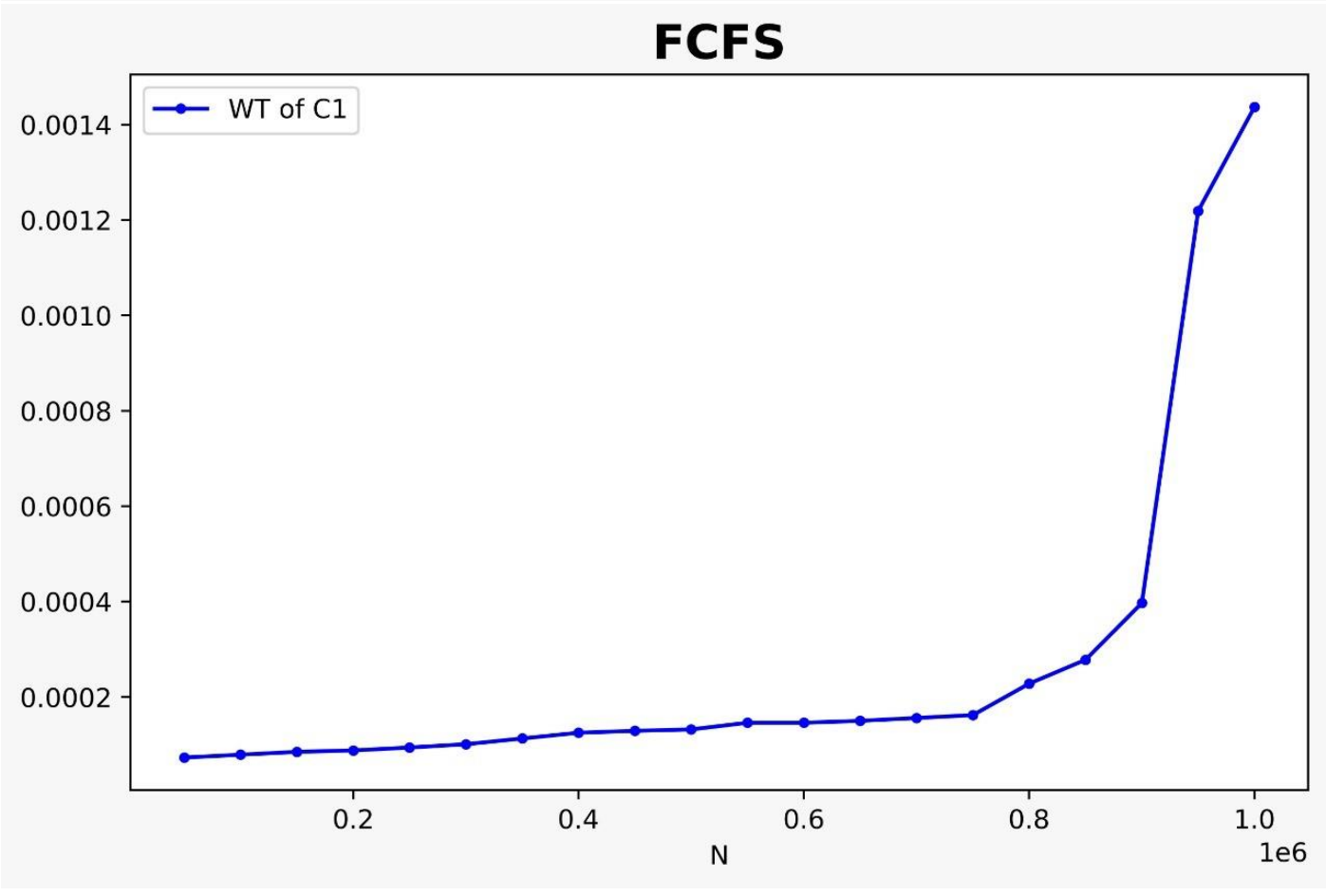
Round Robin: -





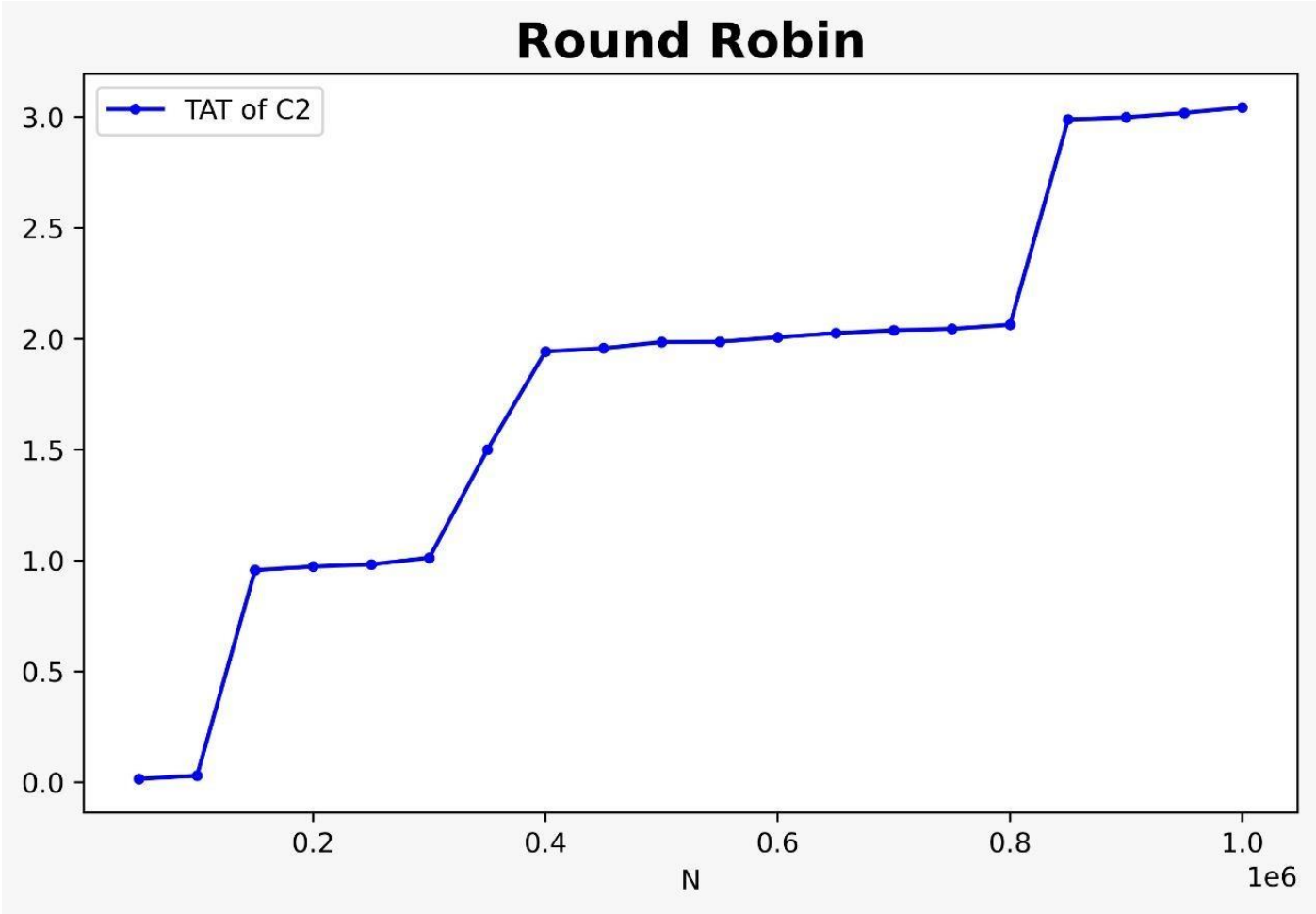
FCFS : -

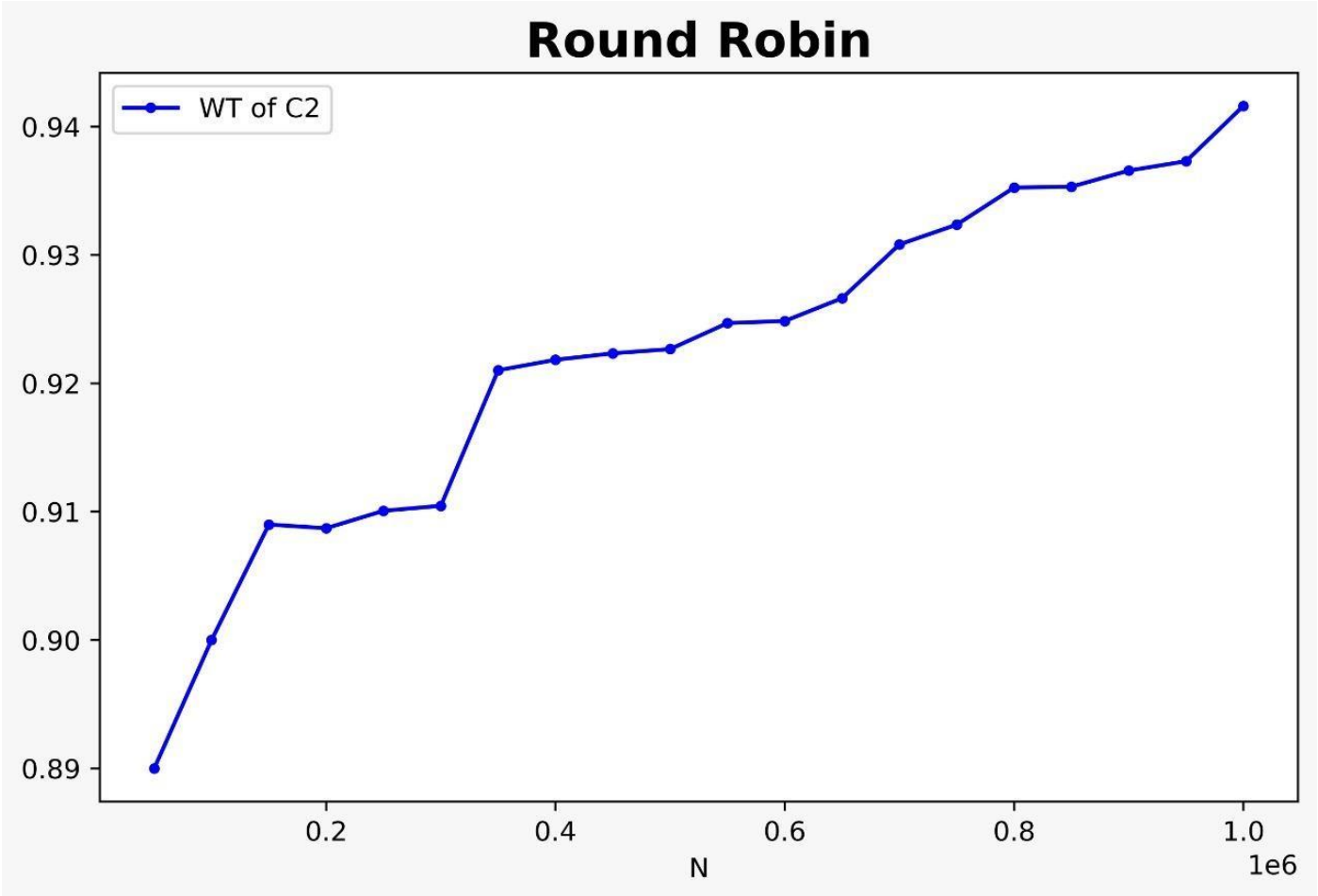

FCFS
TAT of C1
0.004
0.003
0.002
0.001
0.000
0.2
0.4
0.6
0.8
1.0
N
1e6


FCFS
WT of C1
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.2
0.4
0.6
0.8
1.0
N
1e6

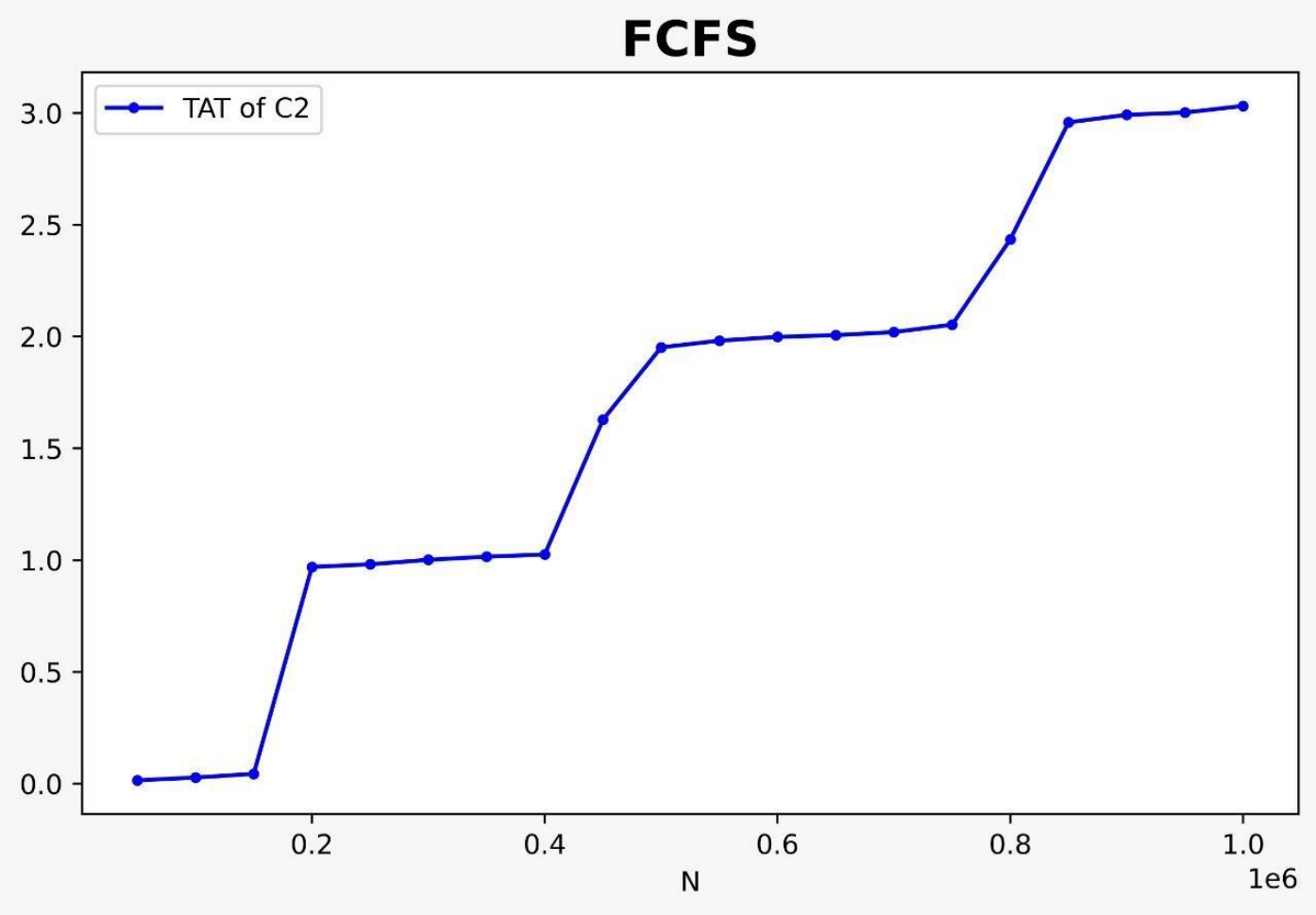
TAT, WT of C2:-

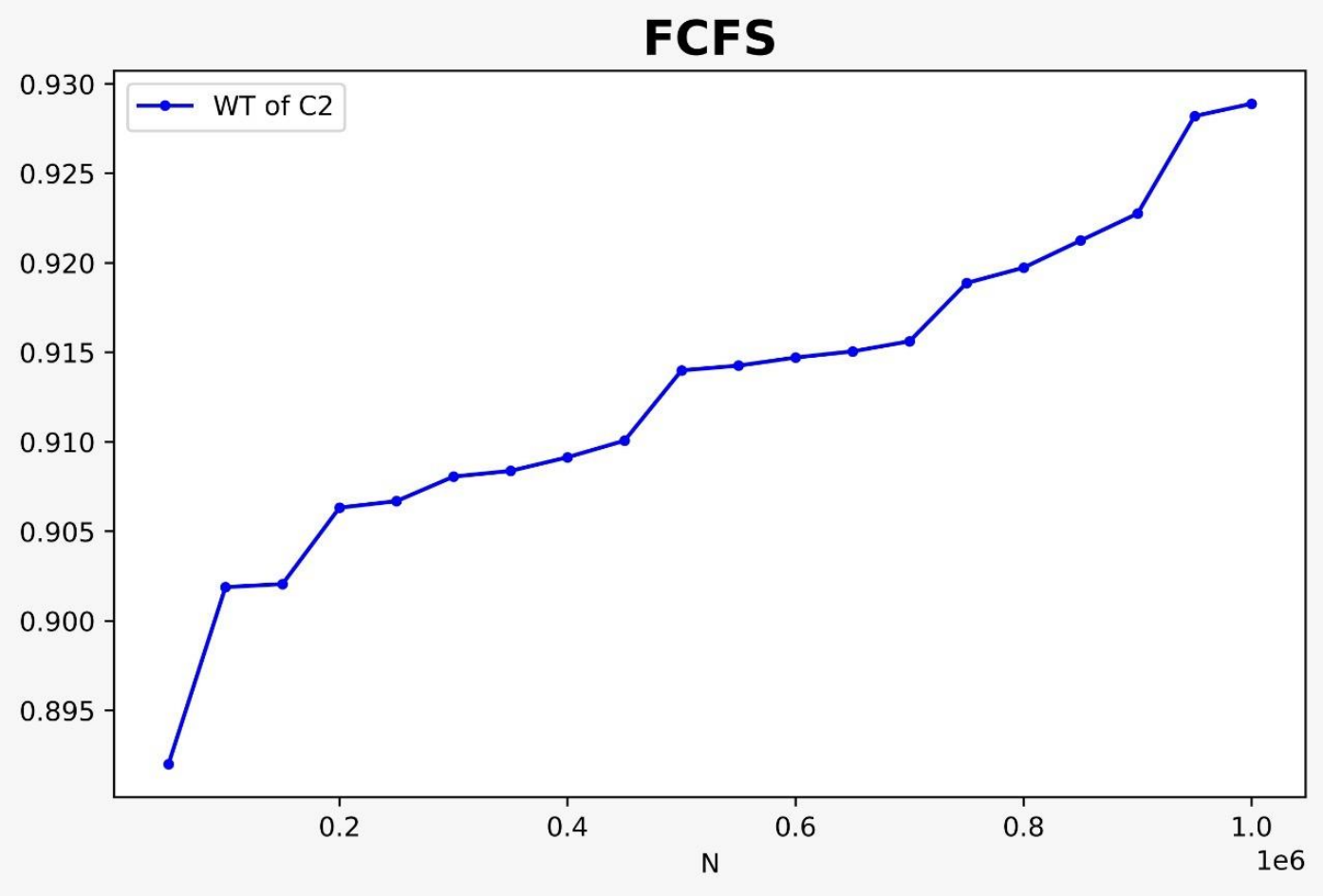
Round Robin: -





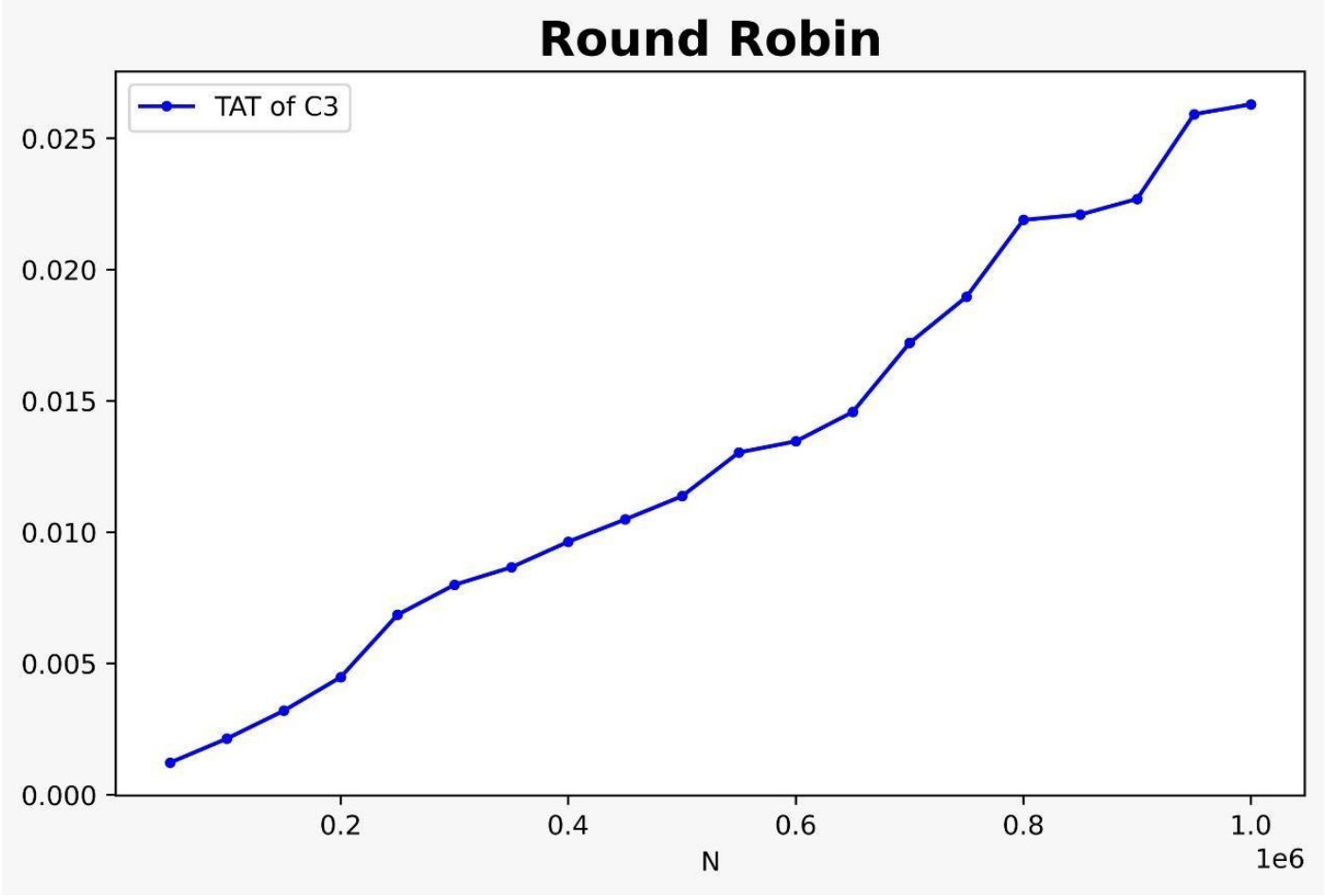
FCFS:-

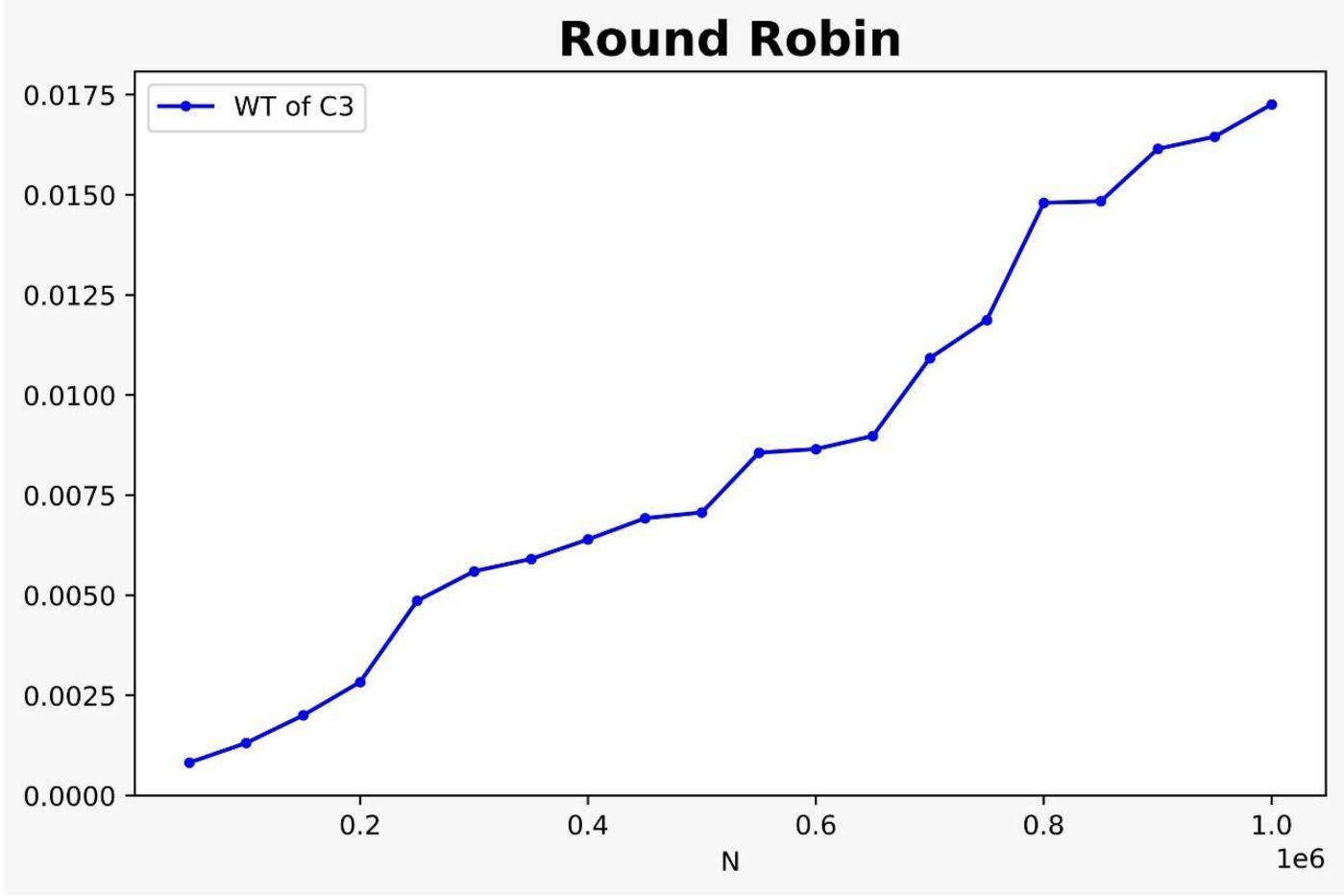

FCFS
TAT of C2
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
N
1e6


FCFS
WT of C2
0.930
0.925
0.920
0.915
0.910
0.905
0.900
0.895
0.2
0.4
0.6
0.8
1.0
N
1e6

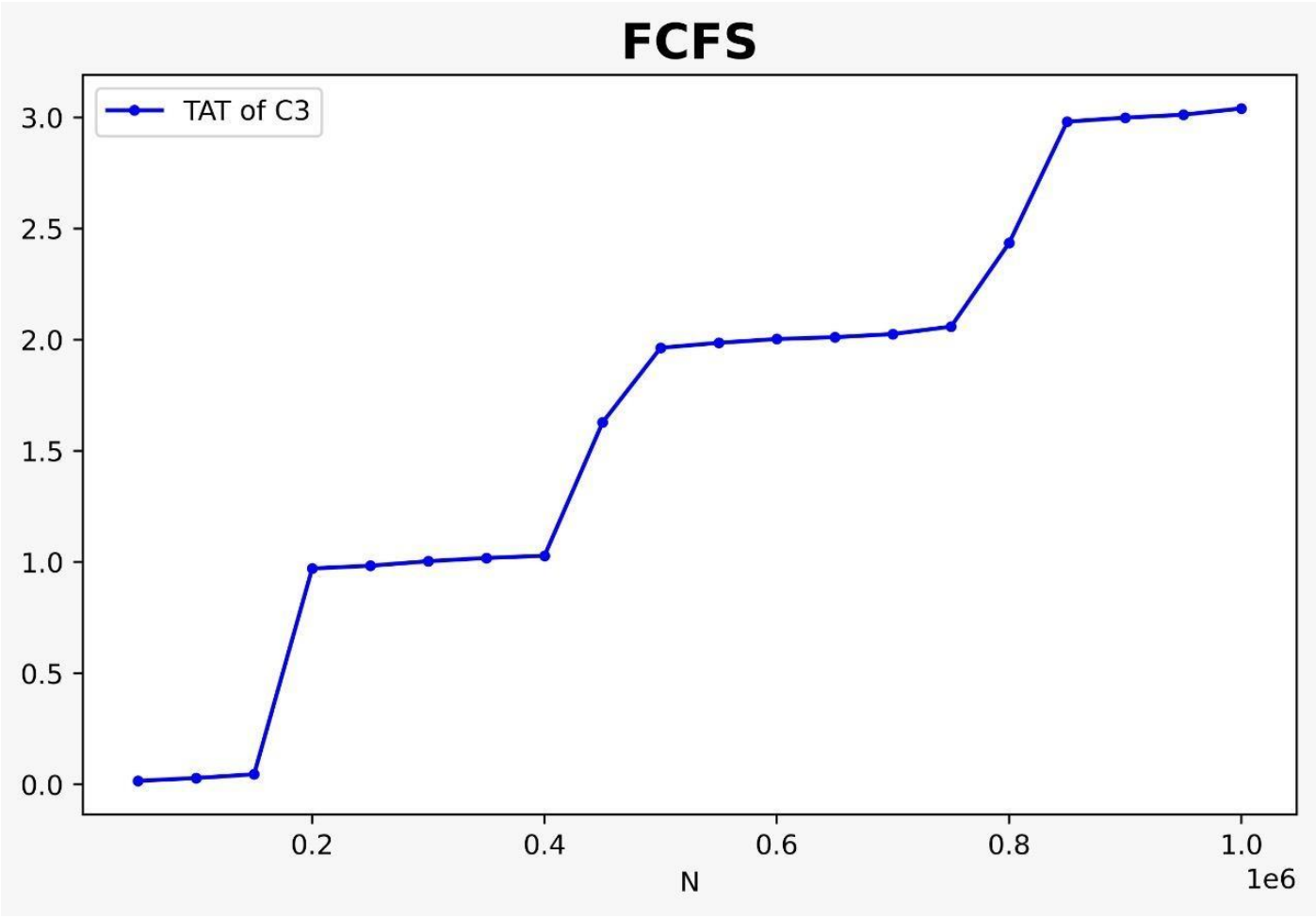
TAT, WT of C3:-

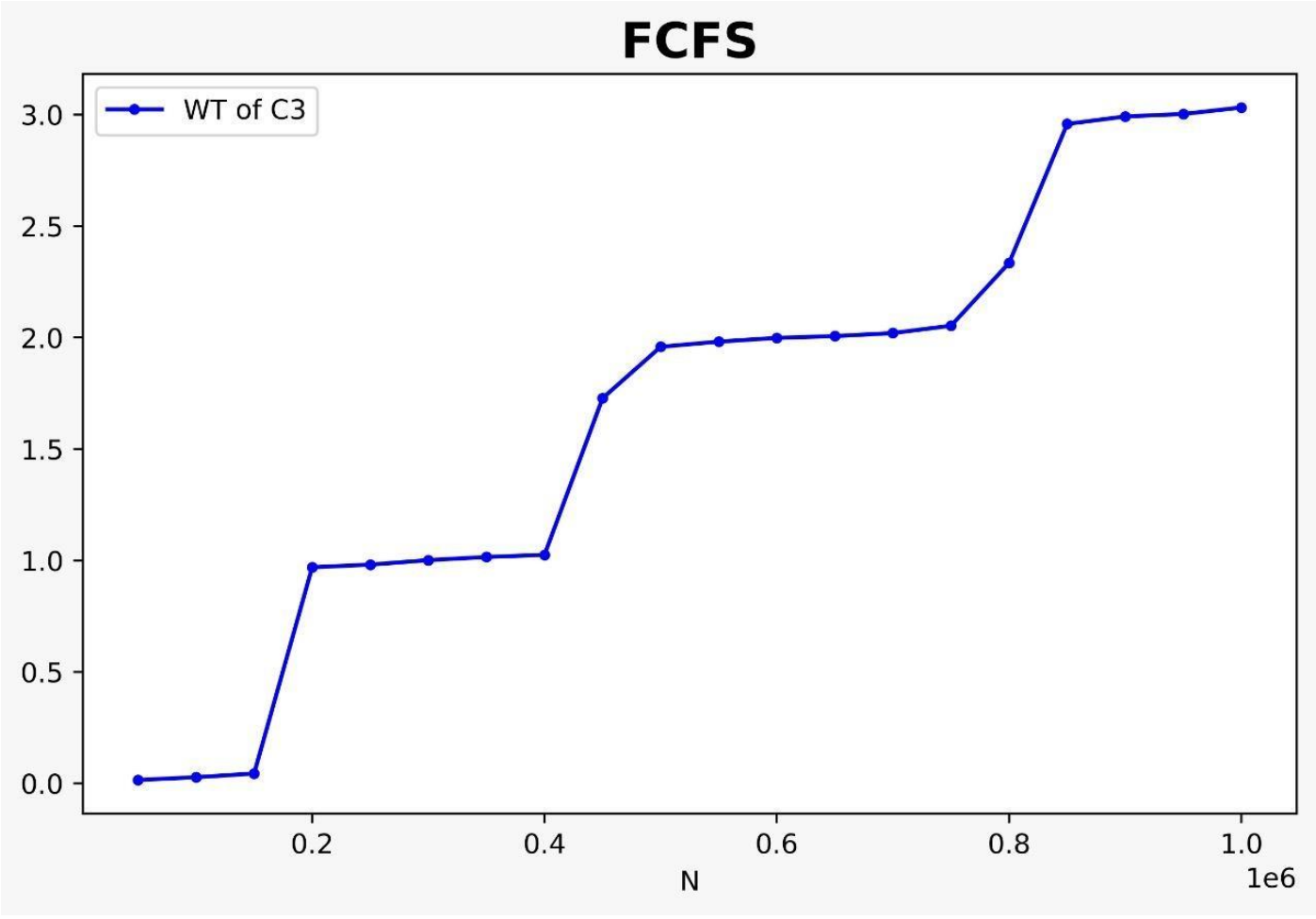
Round Robin: -





FCFS:-


FCFS
TAT of C3
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
N
1e6


FCFS
WT of C3
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
N
1e6

Performance Analysis

It is clearly evident from the graphs that both the algorithms i.e., FCFS & Round Robin are quite effective. It is also evident from the graphs that TATs and WTs of the processes are proportional to the workload. A fairly linear relationship is seen between the times and workload.

Analysis of both algorithms with respect to each process:

C1: The compute intensive process

We noticed that the waiting time of this process is lower as compared to C2 and C3 in both the algorithms because it doesn't get suspended due to lack of any I/O operations. However, TAT of C1 is lesser when scheduled with FCFS because this process is the first in the queue and gets executed to completion first among other process when scheduled using FCFS. However, while using the Round Robin algorithm, the time quantum factor interrupts the execution of the process. Hence ,it's finish time increases.

C2: The I/O intensive process

Since, most of the work done by this process is I/O, it does not differ much in TAT or WT in either algorithm. This process has the longest TAT among the three processes because it prints all the data read by the I/O on the console. print() takes longer time to complete because it is a system call. This significant difference between TAT of C2 and C1, C3 is seen clearly from the graphs above.

C3: The I/O and compute intensive process

We observe that this process executes faster than C1 but slower than C2, though it is supposed to be highly I/O and compute intensive. This process is able to complete faster than C2 because this process doesn't have any print() system call every iteration. So ,there is Kernel involved. A huge difference in the time is seen when executed using both the algorithms because of it's last position in the ready queue. The response time of this process is greatly increased when Round Robin is used as compared to when FCFS is used. This is made apparent by the construction of these algorithms: Round Robin has a focus on response time whereas FCFS wants to completely execute one process before moving on to the next.

Final comparison of the algorithms:

Round Robin algorithms are built to provide better response times, as opposed to FCFS, which all executes the processes sequentially. Hence, the Round Robin algorithms are better for almost all practical use cases of a scheduler.

However, the Round Robin algorithm has some flaws too. Due to enqueuing, dequeuing and context switching, it generates a considerable overhead when compared to FCFS, which has very minimal overhead.

It can also be seen that the execution times of the processes are lesser when using FCFS to schedule them as compared to Round Robin, as CPU cycles are not lost on checking the signal bits over and over again to decide whether the tasks should be started or stopped.

Hence, FCFS algorithms should be used when minimized overhead and execution times of processes is required. However, if we are trying to obtain the minimum response time possible and want to implement a practical system which can multitask, Round Robin algorithms should be used.