Discrete Event Simulator

Anshul Gupta (16305R001)

Features

- Multi-core server machine
- Multi-threaded server
- Round-robin scheduling
- Closed loop system

Input

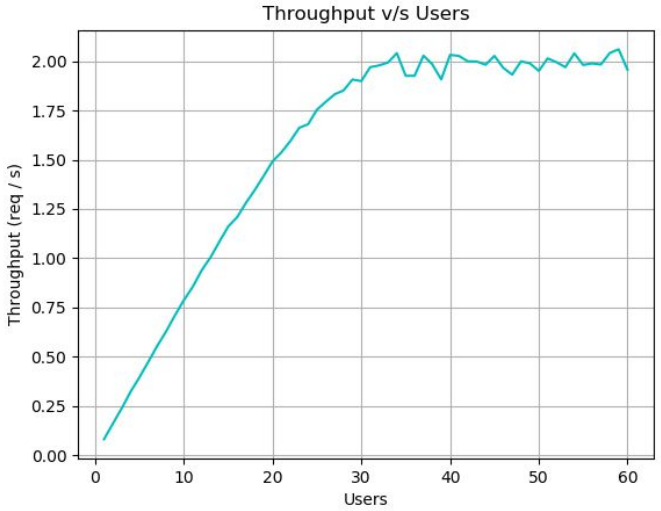
- n_users
- n_CPUs
- easeInTime
- maxIters
- bufferCapacity
- threadpoolSize
- quantum
- ctxSwitchTime
- serviceTimeMean
- requestTimeoutMin
- requestTimeoutMax
- thinkTimeMean
- thinkTimeStdv
- retryThinkTimeMean
- retryThinkTimeStdv

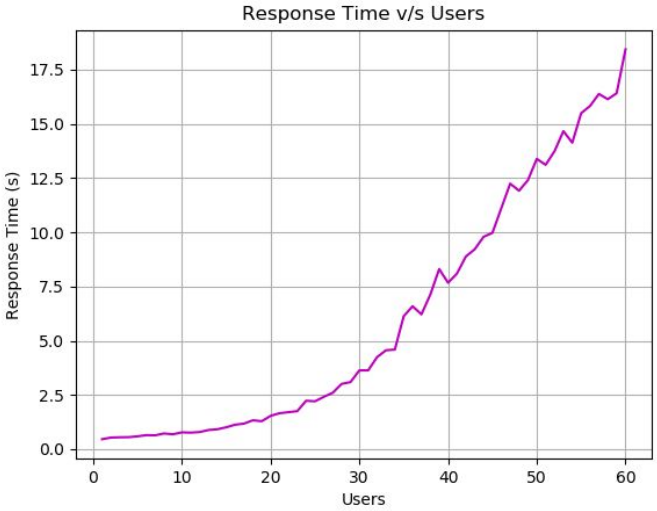
Output

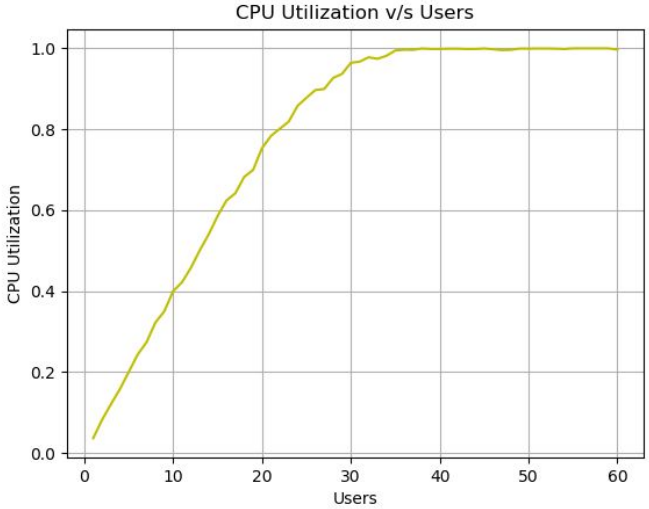
- arrivalRate
- throughput
- goodput
- badput
- responseTime
- utilization
- contextSwitchBusyTimeFraction
- requestsInSystem
- droppedFraction
- dropRate
- timedOutFraction

Correctness of the code

- Results for M/M/1 queuing system are well known
- Increasing the quantum size of round-robin will simulate FIFO scheduling
- Setting buffer size to a large number will act as infinite buffer
- $\tau = 0.5$ sec, $\mu = 2.0$ requests / sec
- Quantum size = 10000 sec
- Buffer Size = 8000000
- n_CPUs = 1
- Think Time = 12 sec
- $M^* = 25$

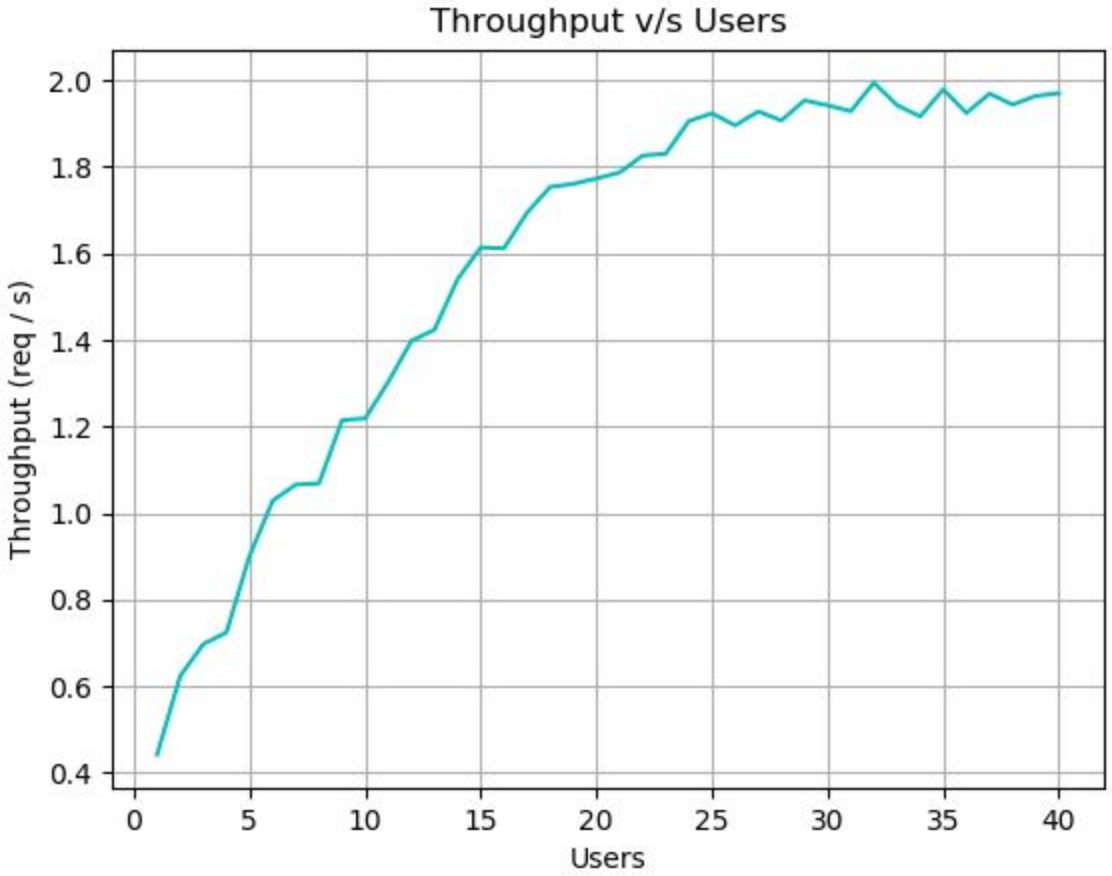

Throughput v/s Users
Throughput (req / s)
Users
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0
10
20
30
40
50
60


Response Time v/s Users
Response Time (s)
Users
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
10
20
30
40
50
60


CPU Utilization v/s Users
CPU Utilization
Users
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60

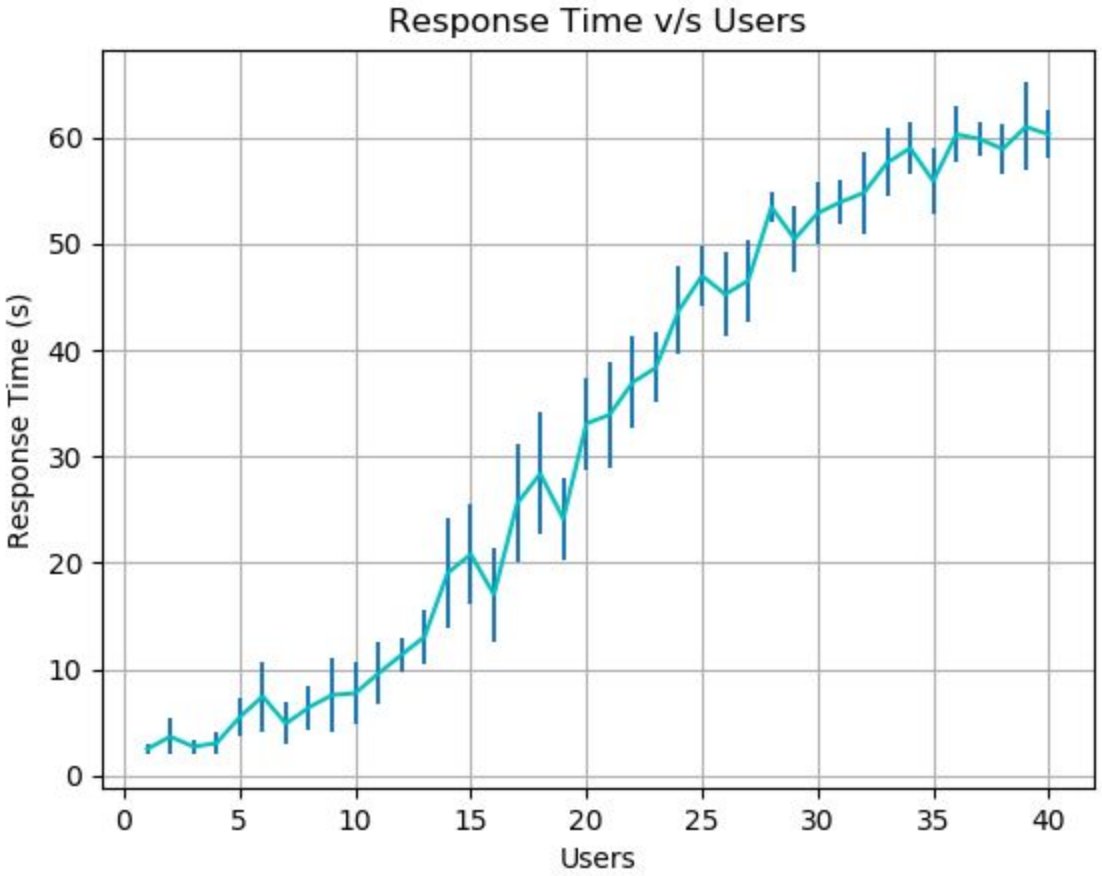
Simulation Parameters - 1 (Number of users)

- n_users = [1, 40]
- n_CPUs = 4
- easeInTime = 20.0
- maxIters = 10000
- bufferCapacity = 800
- threadpoolSize = 200
- quantum = 0.5
- ctxSwitchTime = 0.01
- serviceTimeMean = 2
- requestTimeoutMin = 5.0
- requestTimeoutMax = 15.0
- thinkTimeMean = 12.0
- thinkTimeStdv = 4.0
- retryThinkTimeMean = 12.0
- retryThinkTimeStdv = 4.0


Throughput v/s Users
Throughput (req / s)
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
5
10
15
20
25
30
35
40
Users

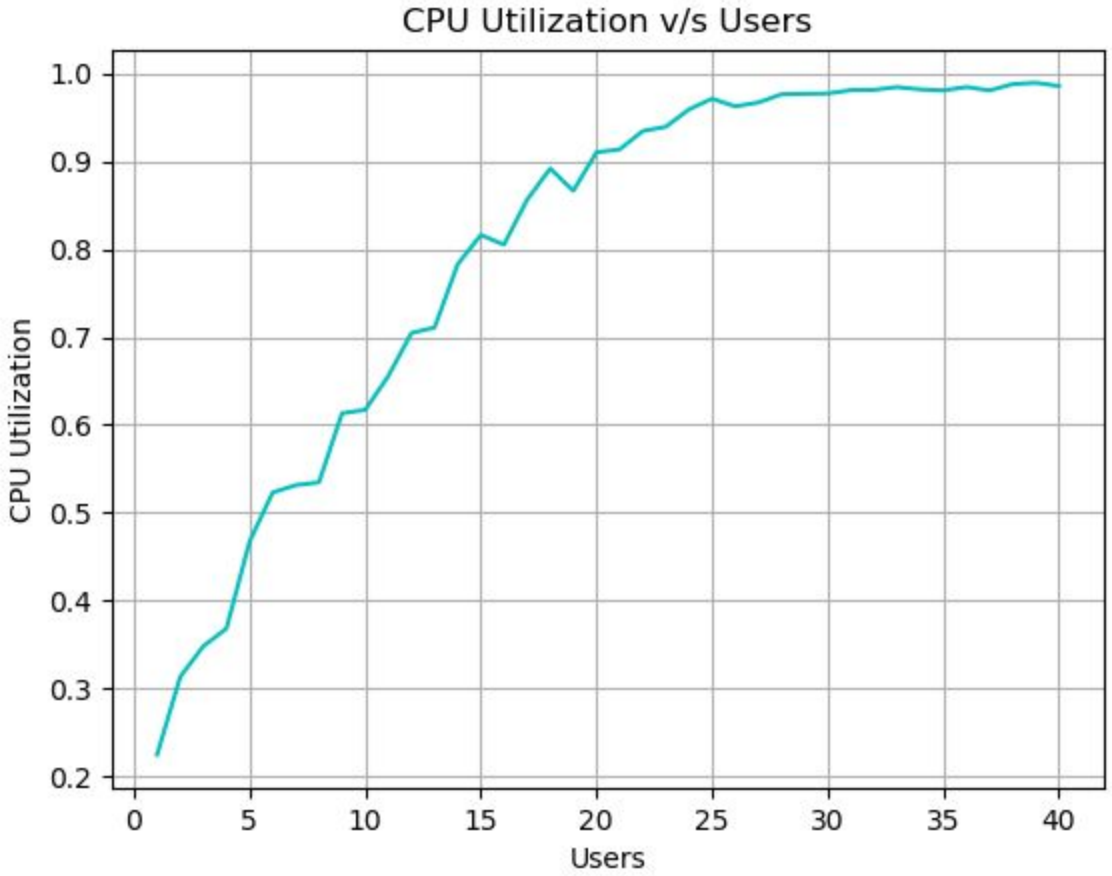
Observations

- Until saturation, throughput increases almost linearly
- At high load, it saturates at 2 requests / sec


Response Time v/s Users
Response Time (s)
0
10
20
30
40
50
60
0
5
10
15
20
25
30
35
40
Users

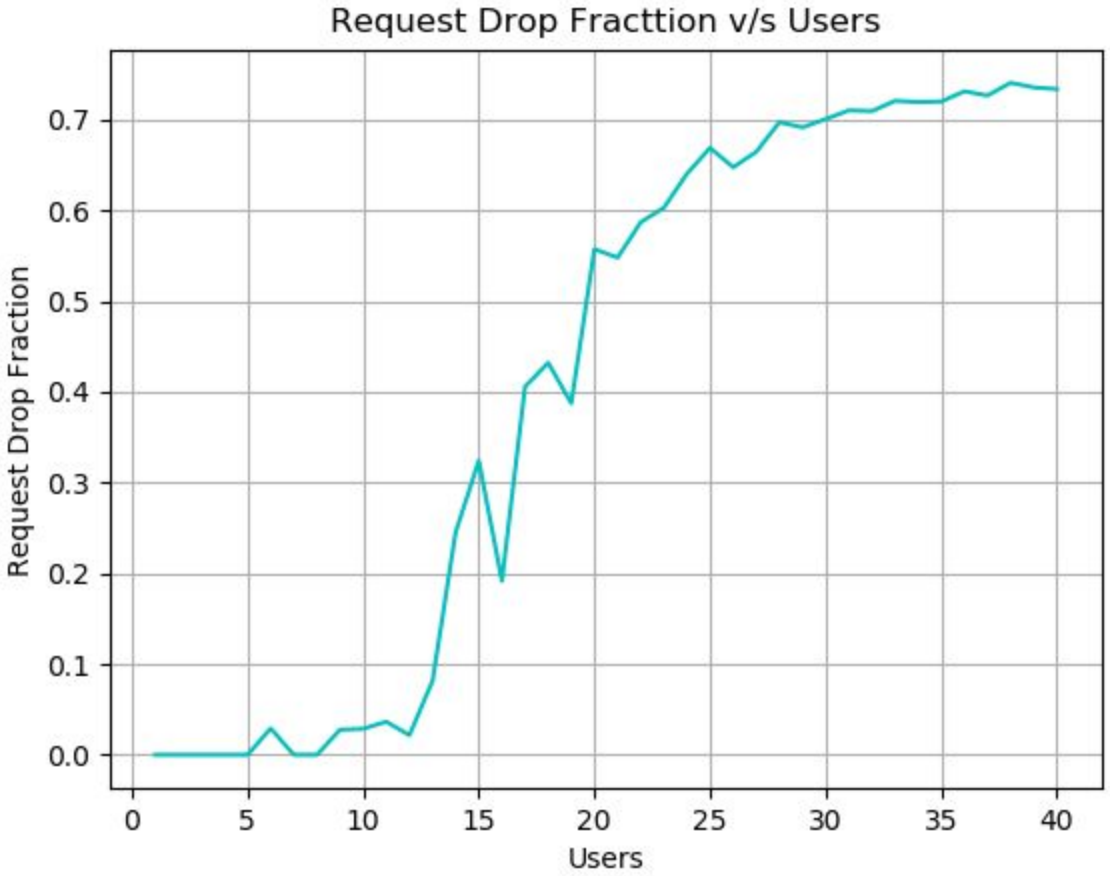
Observations

- At low load, the response time is same as the service time (2 sec)
- After saturation point, response time increases sharply


CPU Utilization v/s Users
CPU Utilization
Users
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30
35
40

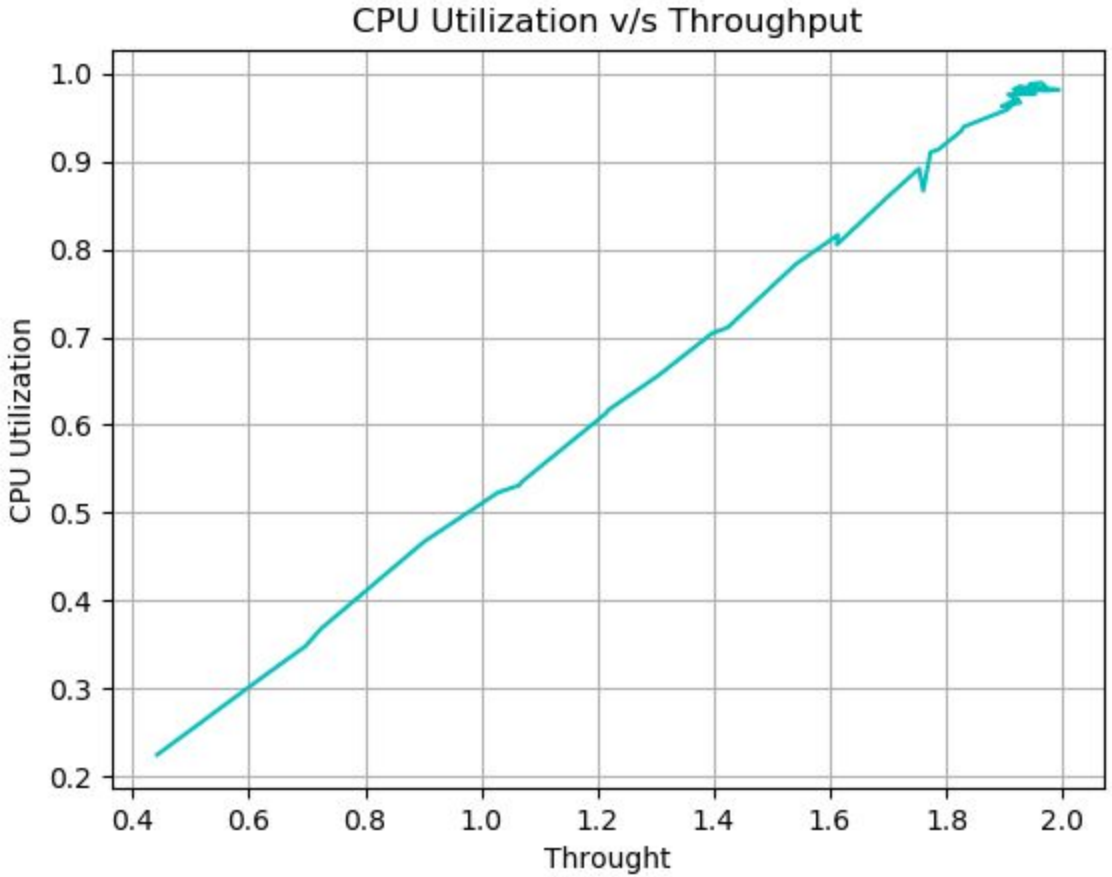
Observations

- As expected, the utilization is ~80% at M* = 25
- It saturates at around 30 - 32 users


Request Drop Fracttion v/s Users
Request Drop Fraction
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25
30
35
40
Users

Observations

- As the number of users increases, the limited buffer cannot accommodate the requests and starts dropping them

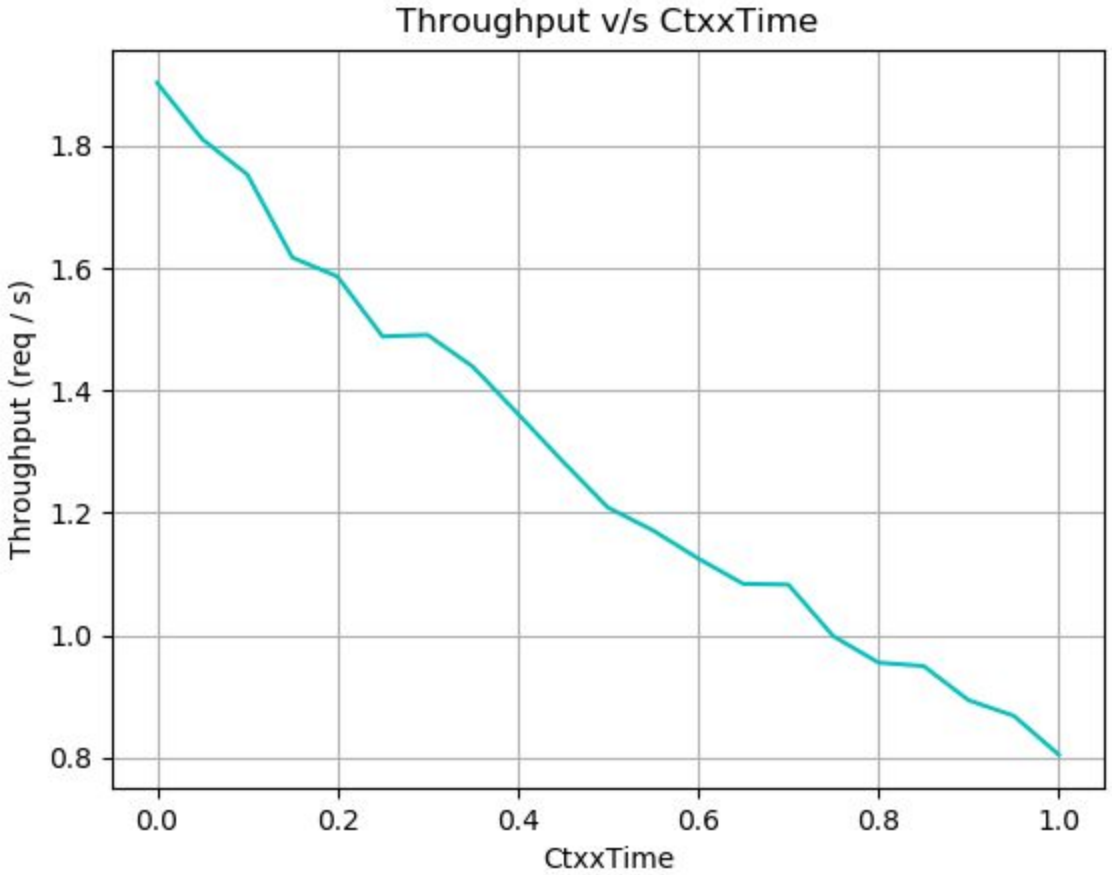

CPU Utilization v/s Throughput
CPU Utilization
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
Throught

Observations

- The linear relationship between throughput and utilization is maintained in the graph.
- Utilization is 1 when throughput saturates at 2 requests / sec

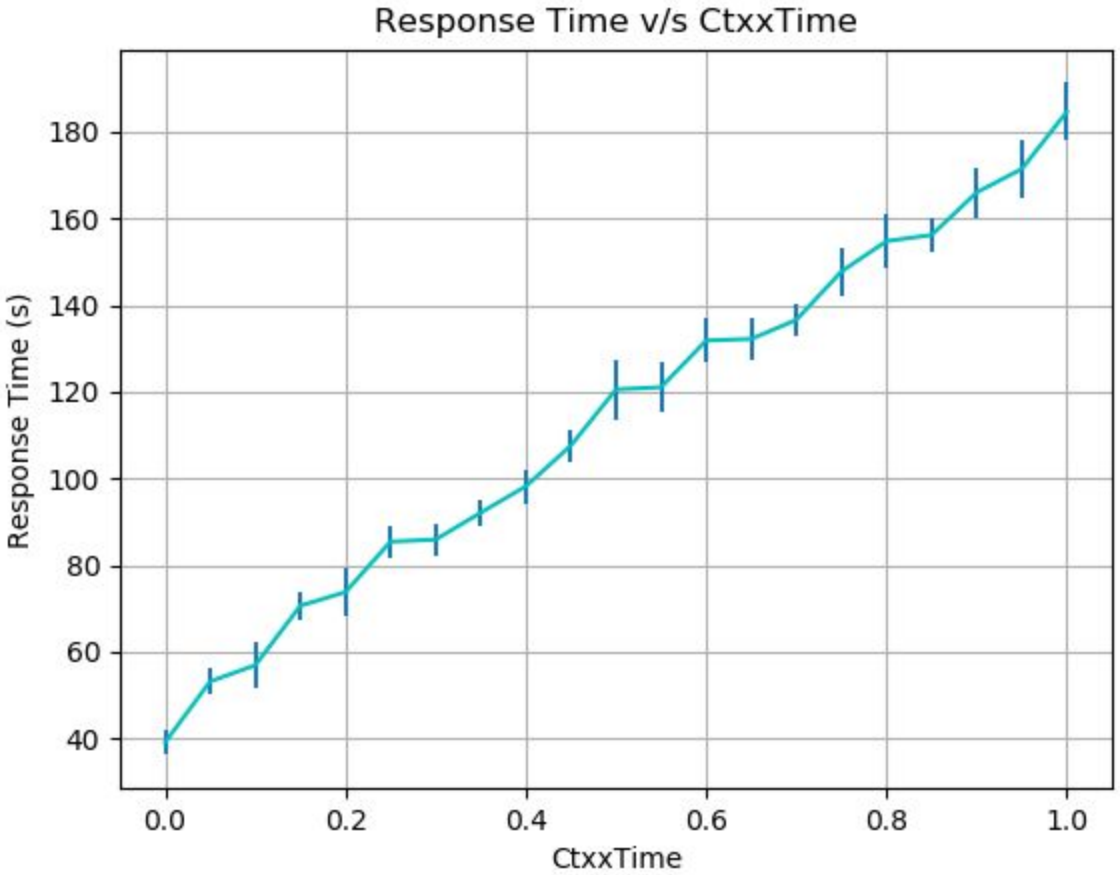
Simulation Parameters - 2 (Context Switching Time)

- n_users = 25
- n_CPUs = 4
- easeInTime = 20.0
- maxIters = 10000
- bufferCapacity = 800
- threadpoolSize = 200
- quantum = 0.5
- ctxSwitchTime = [0.0, 1.0, 0.05]
- serviceTimeMean = 2
- requestTimeoutMin = 5.0
- requestTimeoutMax = 15.0
- thinkTimeMean = 12.0
- thinkTimeStdv = 4.0
- retryThinkTimeMean = 12.0
- retryThinkTimeStdv = 4.0


Throughput v/s CtxxTime
Throughput (req / s)
1.8
1.6
1.4
1.2
1.0
0.8
0.0
0.2
0.4
0.6
0.8
1.0
CtxxTime

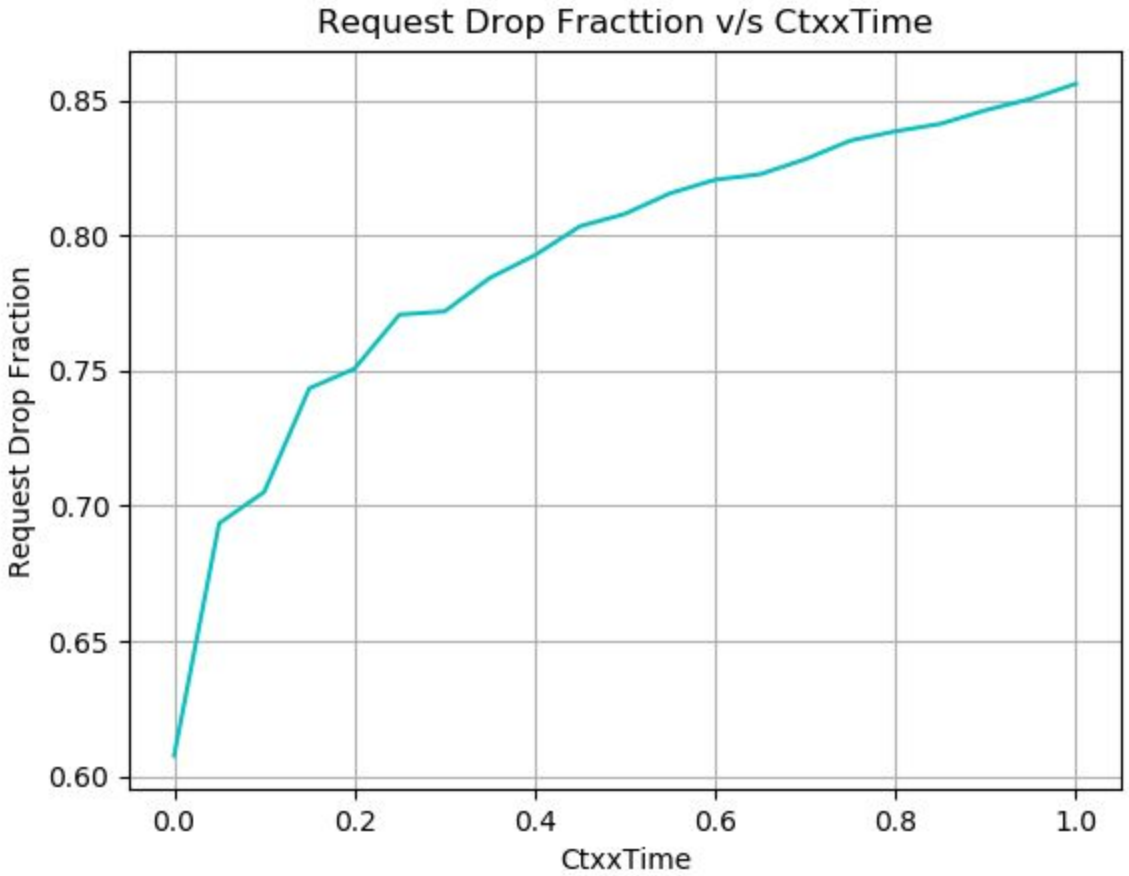
Observations

- As the context switching time increases, it is obvious that the throughput will decrease
- Higher context switching time will cause system to reach saturation faster


Response Time v/s CtxxTime
Response Time (s)
180
160
140
120
100
80
60
40
0.0
0.2
0.4
0.6
0.8
1.0
CtxxTime

Observations

- Similar reasoning as that of throughput
- Higher context switching time means more delays between processing the requests

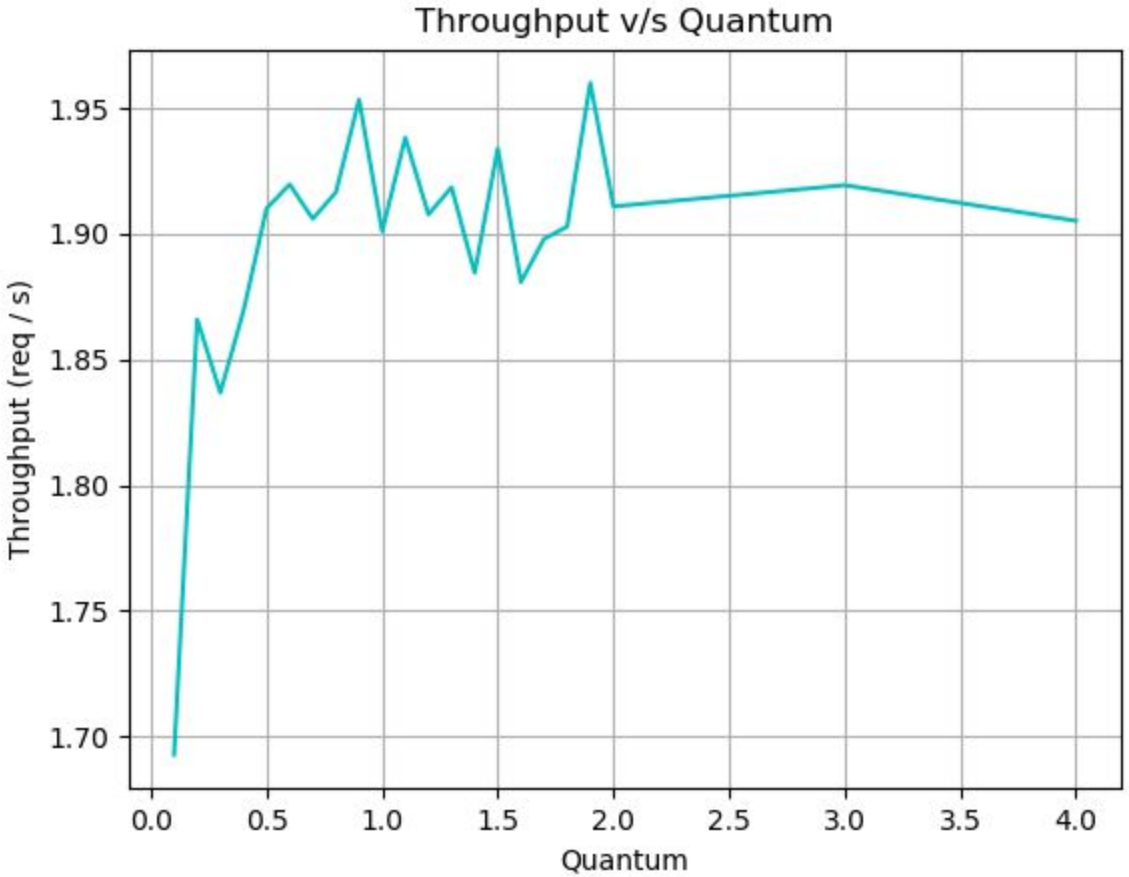

Request Drop Fracttion v/s CtxxTime
Request Drop Fraction
0.85
0.80
0.75
0.70
0.65
0.60
0.0
0.2
0.4
0.6
0.8
1.0
CtxxTime

Observations

- Since the requests take more time to process, the requests in buffer has to stay longer
- The buffer empties at a lower rate so the requests drop rate increases

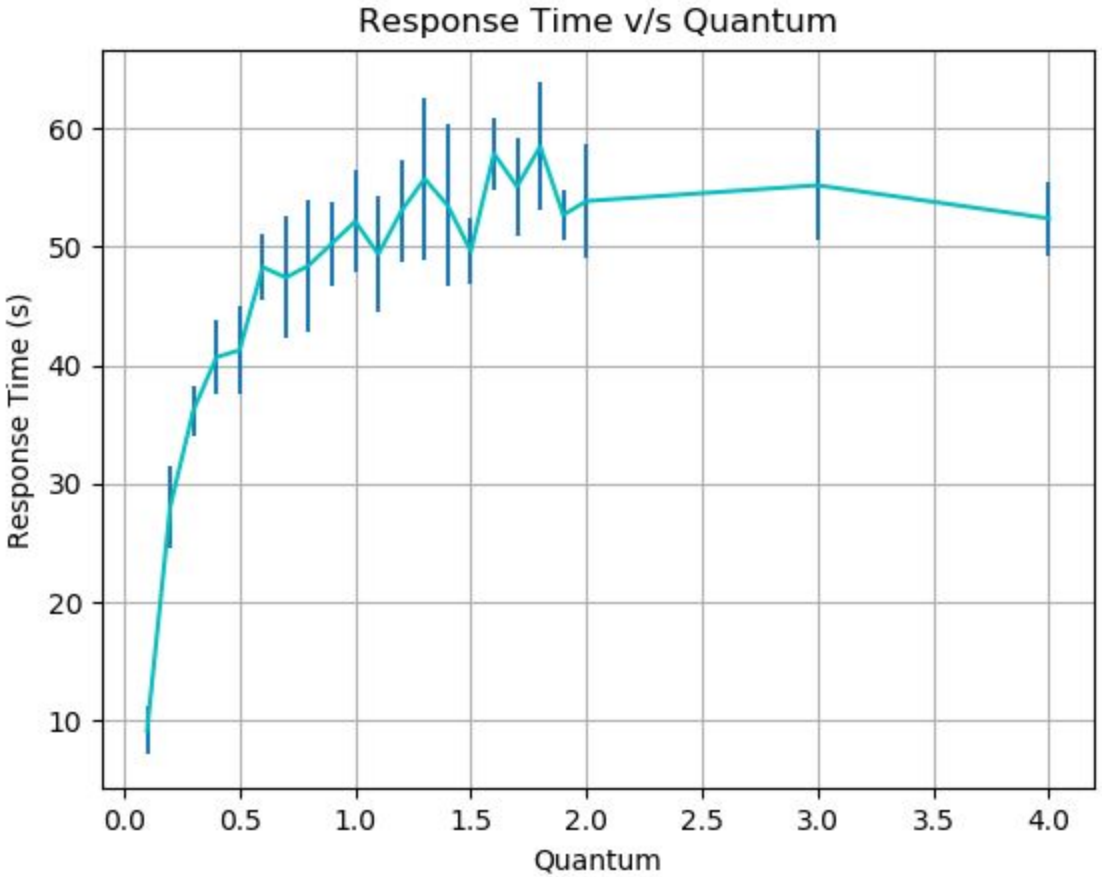
Simulation Parameters - 3 (Quantum)

- n_users = 25
- n_CPUs = 4
- easeInTime = 20.0
- maxIters = 10000
- bufferCapacity = 800
- threadpoolSize = 200
- quantum = [0.1, 2.0, 0.1] + (3.0, 4.0)
- ctxSwitchTime = 0.01
- serviceTimeMean = 2
- requestTimeoutMin = 5.0
- requestTimeoutMax = 15.0
- thinkTimeMean = 12.0
- thinkTimeStdv = 4.0
- retryThinkTimeMean = 12.0
- retryThinkTimeStdv = 4.0


Throughput v/s Quantum
Throughput (req / s)
1.95
1.90
1.85
1.80
1.75
1.70
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quantum

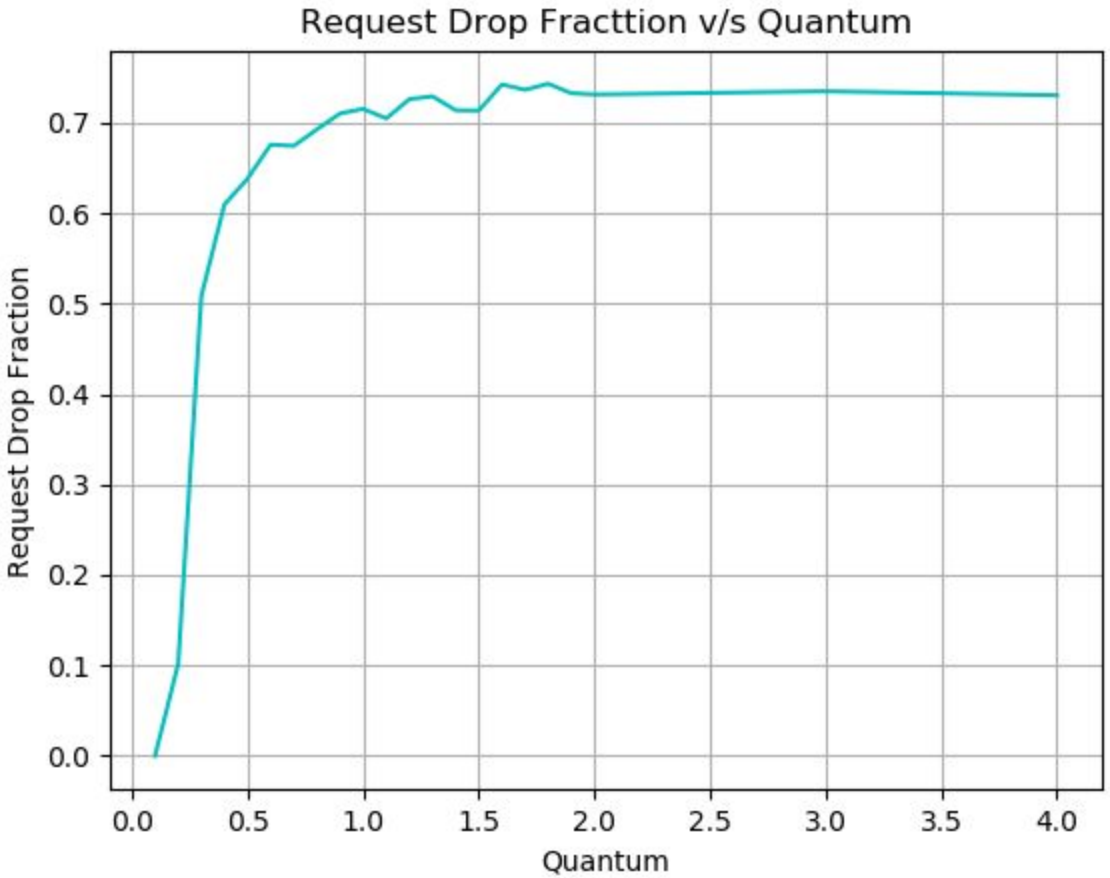
Observations

- When quantum size is very low, there are more context switches therefore less throughput
- When quantum size is very high, there are very less context switches. So the throughput saturates as quantum is much higher than the service time of the request


Response Time v/s Quantum
Response Time (s)
10
20
30
40
50
60
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quantum

Observations (Bug)

- Unable to explain the graph
- As the context switches are very high (low quantum), response time should be high and it should decrease as context switches decrease
- The graph is contradicting this


Request Drop Fracttion v/s Quantum
Request Drop Fraction
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quantum

Observations (Bug)

- Since the throughput is low at low quantum, the requests stay in buffer for a longer time so the new requests should be dropped as the buffer is full
- Similarly at high quantum, the requests get processed very fast and the buffer clears at a higher rate. So requests drop rate should be low
- The graph is contradicting this

End