CS 681:
Simulation Assignment

Anuj Mittal • 140050024
Sumith Kulal • 140050081

Goal of Assignment

- Study the performance of a server through Discrete Event Simulation
- Model servers which are difficult to study through theoretical queuing systems
- Analyse the results to check if they are consistent with the theoretical laws and results

Model Description

- Server
  - Multi-core system
  - Thread for each Request
  - Thread to core affinity
  - Round Robin Scheduling
  - Limited Number of Threads
- Users
  - In loop : think, issue request, wait for response

Model Description

- Queue
  - Finite buffer : Requests gets dropped if buffer full
    - User will issue new request only after time out
- Requests
  - Timeouts possible
    - Server will keep processing the request
    - User issue another request after think time

Results

Fixed Parameters for Experiments

- Num of CPU Cores = 8
- Thread Limit = 100
- Buffer Size = 200
- Time Quantum for each Core = 100 ms
- Context Switch Overhead = 1 ms
- Simulation Run Time = 3000 s
- Transient Time (Not used for metric calculation) = 200 s

Think Time Distribution

- Normal Dstribution
- Mean = 10
- Variance = 4
- Negative values sampled are ignored

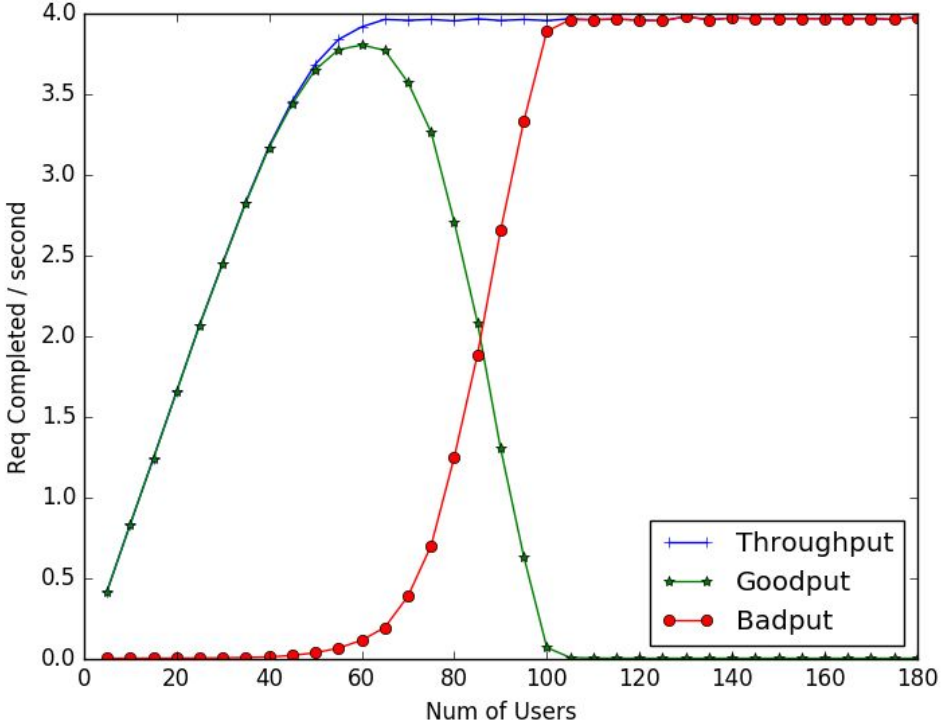
TimeOut Distribution

- Minm Threshold + Variable Time
- Constant Threshold = 10
- Variable Time has Exponential Distribution
- Mean of Variable Time = 5
- Timeout = 10 + EXP(⅕)

Service Time Distribution

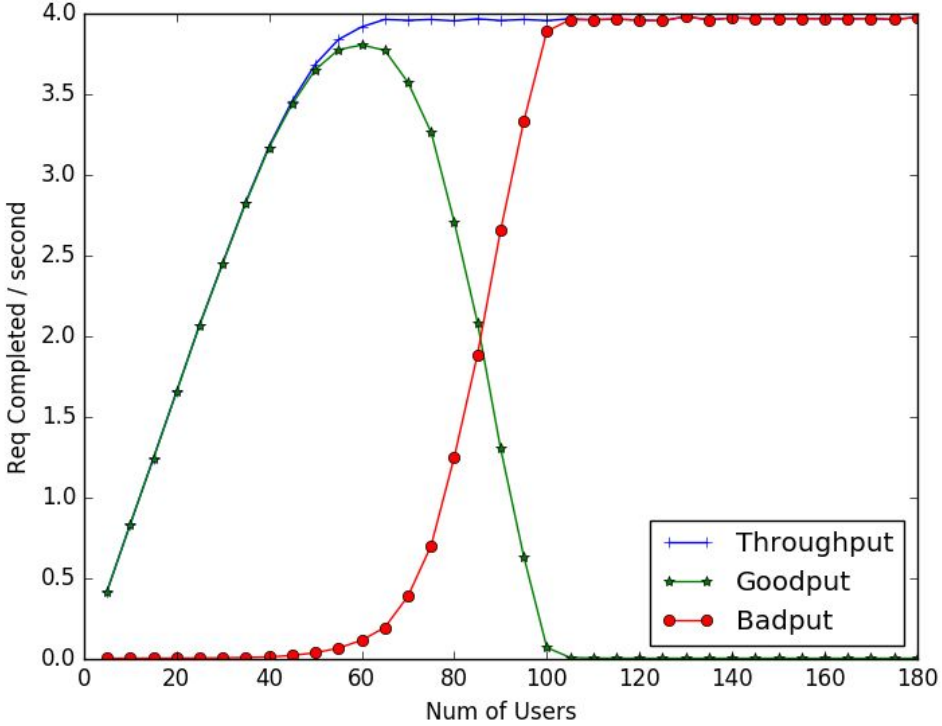
- Service Time can be Constant, Uniform or Exponential
- Probability of choosing Constant Time = 0.2
- Probability of choosing Uniform Distibution = 0.5
- Probability of choosing Exponential Distibution = 0.3
- Value of Constant Service Time = 2 s
- Uniform Distribution ~ U (1, 3)
- Exponential Distribution ~ EXP (½)
- Expected Value of Service Time = 2 s

Throughput vs Number of Users



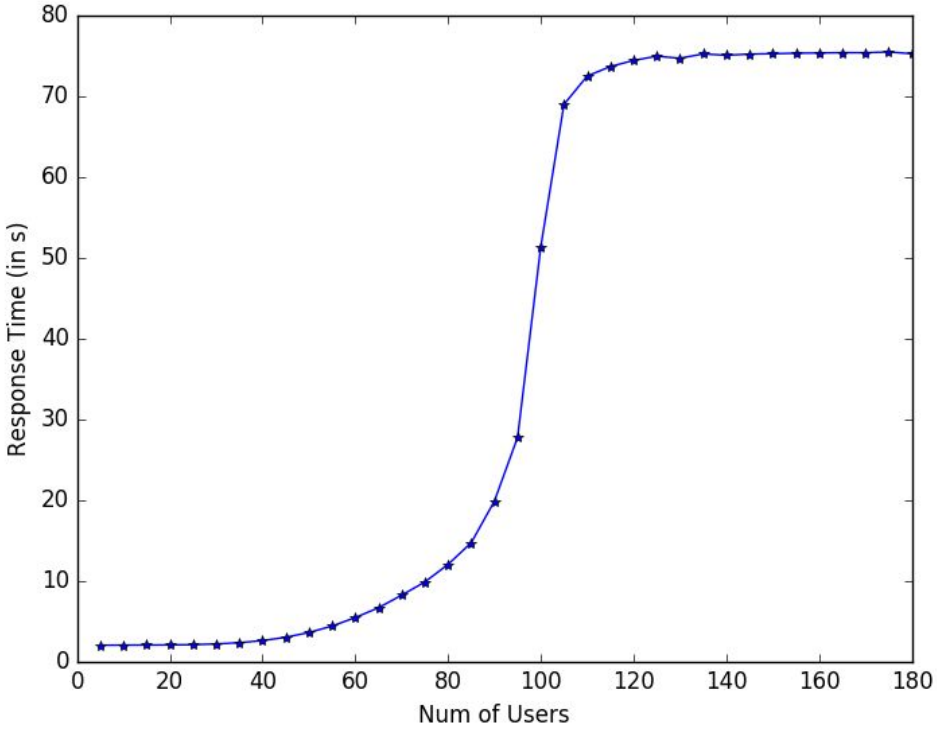
- Throughput increases linearly initially as all the requests can be processed by the server
- At 65 users, throughput saturates at a value very close to 4
- Remains at around the same level for subsequent increase in number of users

Throughput vs Number of Users



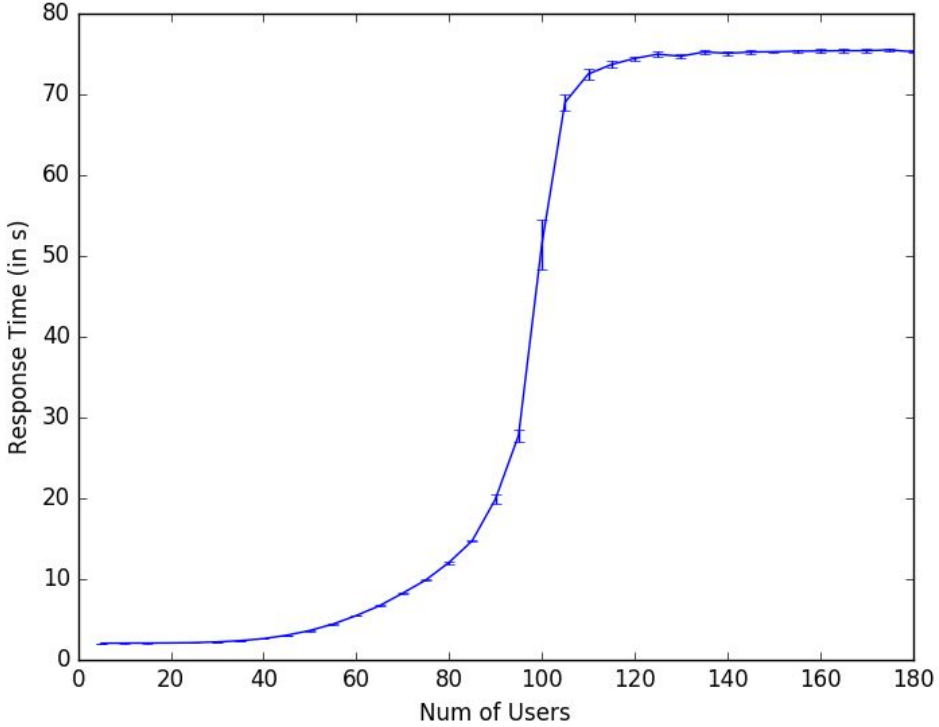
- As number of user increases, goodput increases upto 60 users, and starts decreasing after that as request starts getting time out
- Badput is almost 0 until 50 users and has sharp increase after 65 users.
- At 100 users, goodput becomes almost 0 and badput saturates

Response Time vs Number of Users

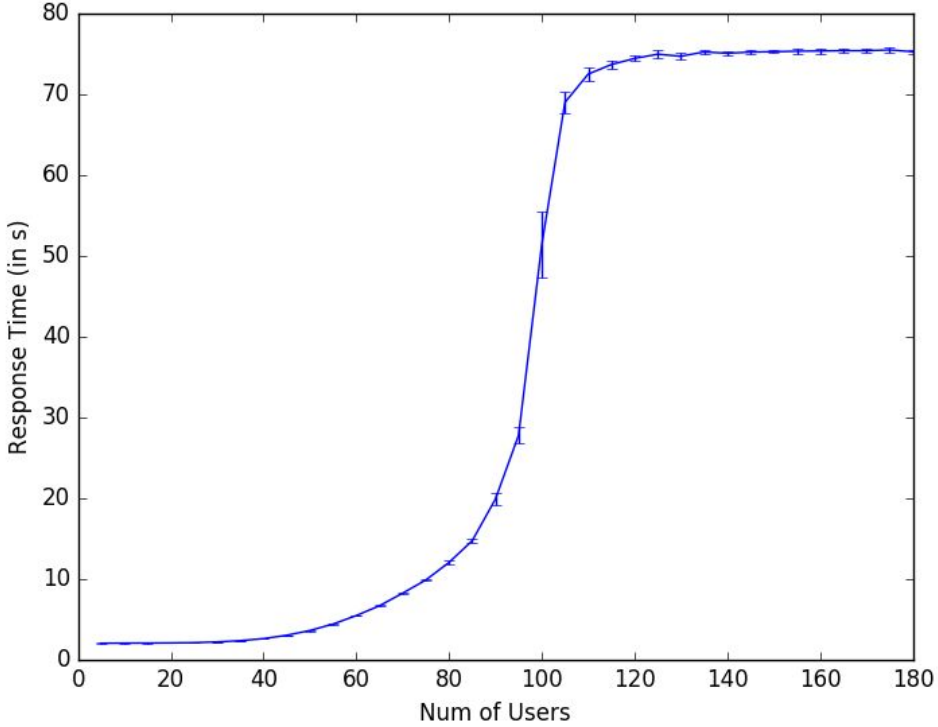


- Response time increases gradually at the start
- For 5 users, response time is 1.997 s
- Response first gradually increases till 90 users and then have sharp increase
- At 120 users, the response time saturates because the buffer is finite

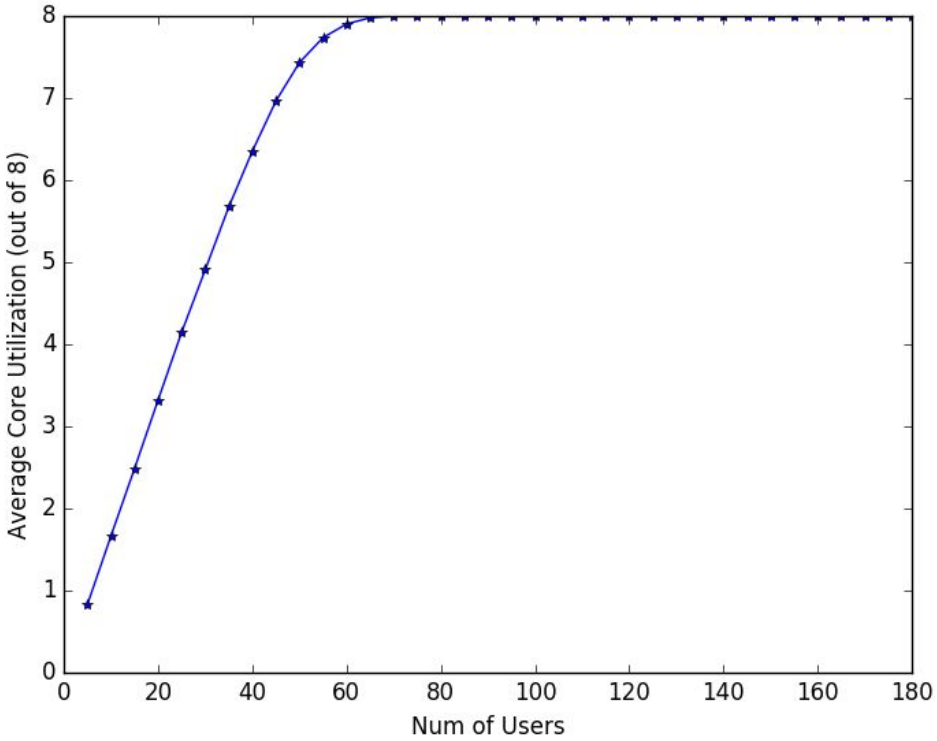
Response Time (95 % Confidence Interval) vs Number of Users



Response Time (99 % Confidence Interval) vs Number of Users

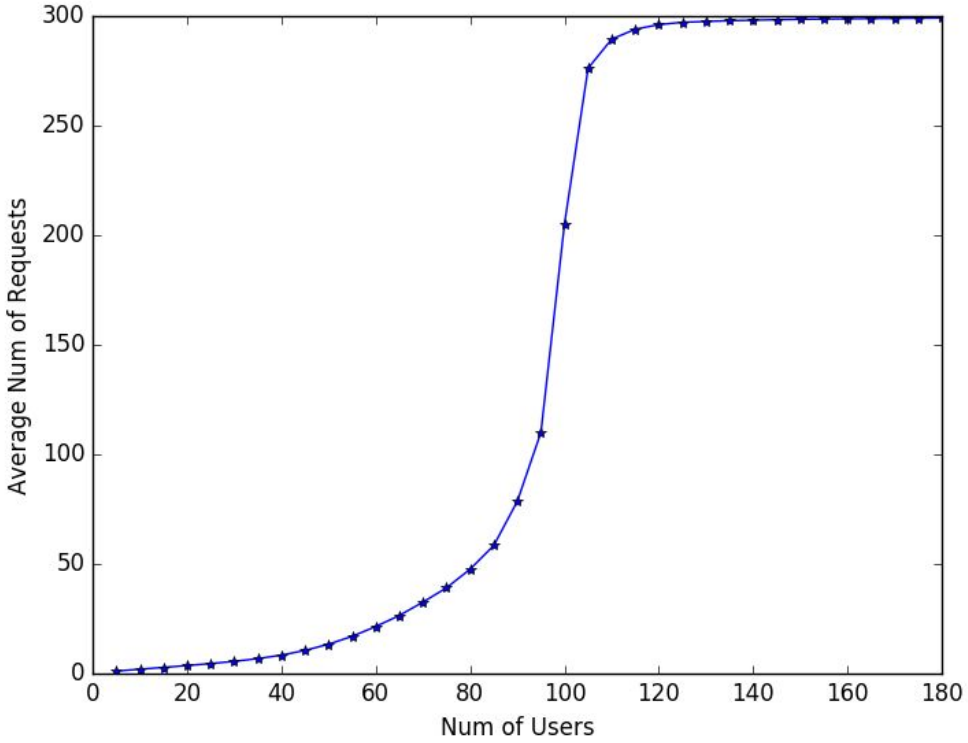


Core Utilization vs Number of Users



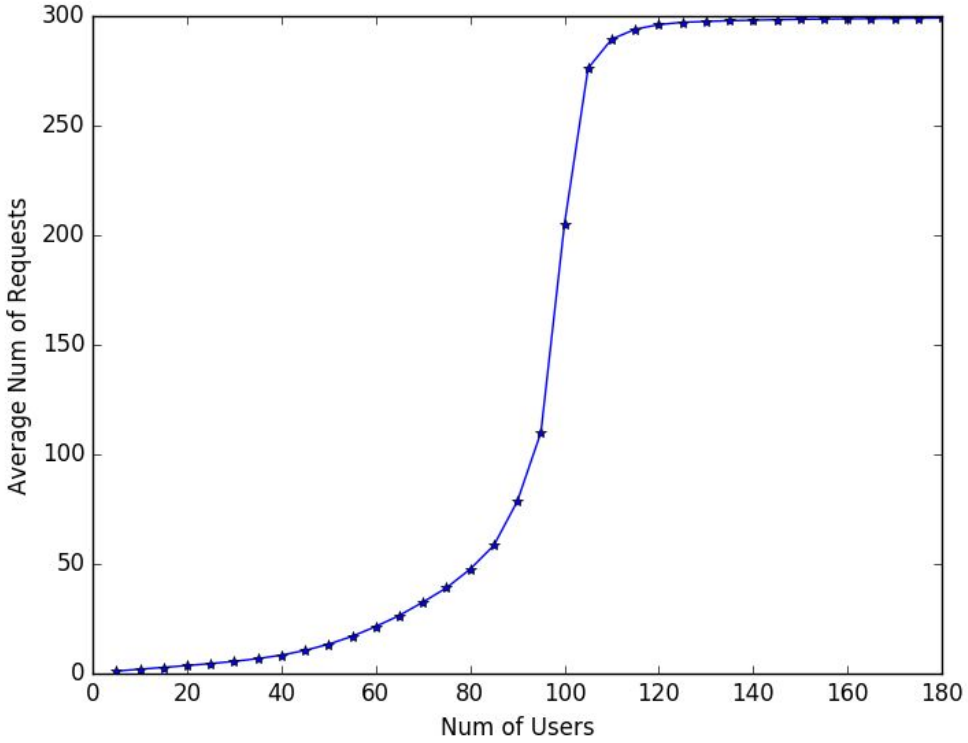
- Initially increases as the CPU is underutilized and with more number of users, there is more utilization
- At 65 users, Utilization reaches a horizontal line at saturation when utilization hits close to 100% and all 8 cores are utilized

Number of Requests in System vs Number of Users



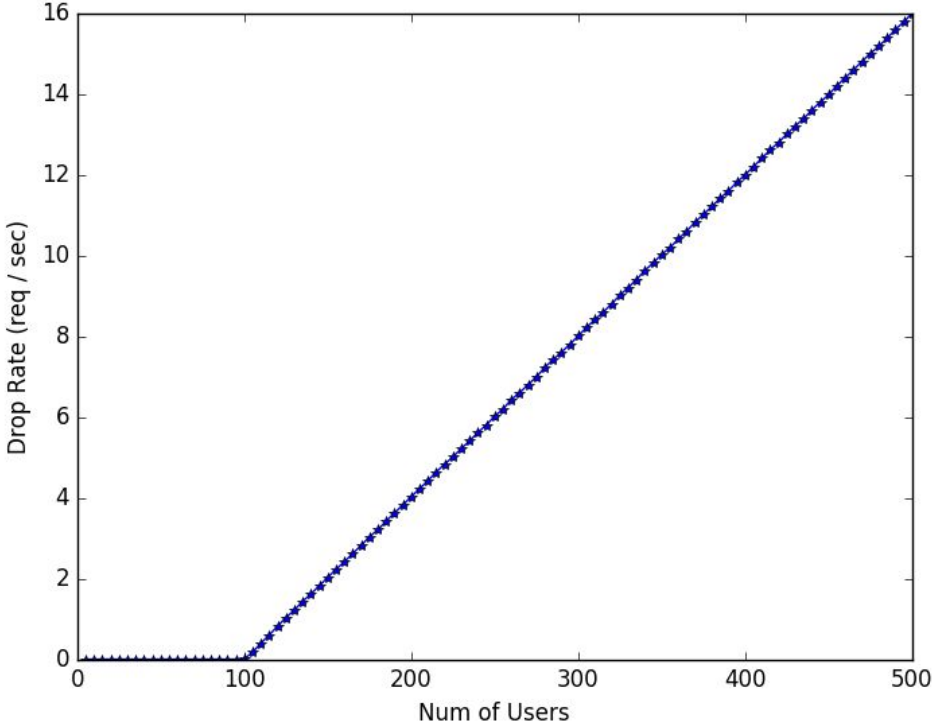
- Initially number of requests is less than the number of users and increases slowly
- At 80 users, the number of requests in the system starts increasing at a steep rate due to server getting overburdened

Number of Requests in System vs Number of Users



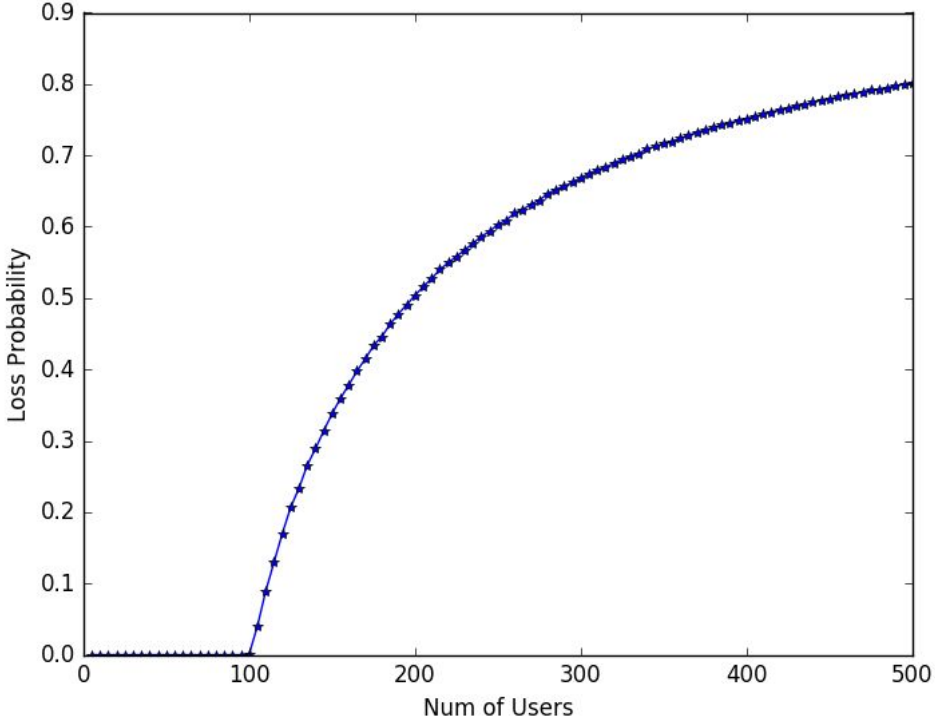
- At 95 users, the number of requests bypasses number of users. This is because request starts getting timed out and user issues new request
- At 120 users, it saturates to the value of 300 (buffer size + thread limit)

Drop Rate vs Number of Users



- Till 95 users, no request is dropped
- After 100 users, the drop rate increases at linear rate
- This is because buffer can keep a fixed number of requests. All the requests added due to adding a new user will get dropped
- slope of graph = 1 / 25

Loss Probability vs Number of Users



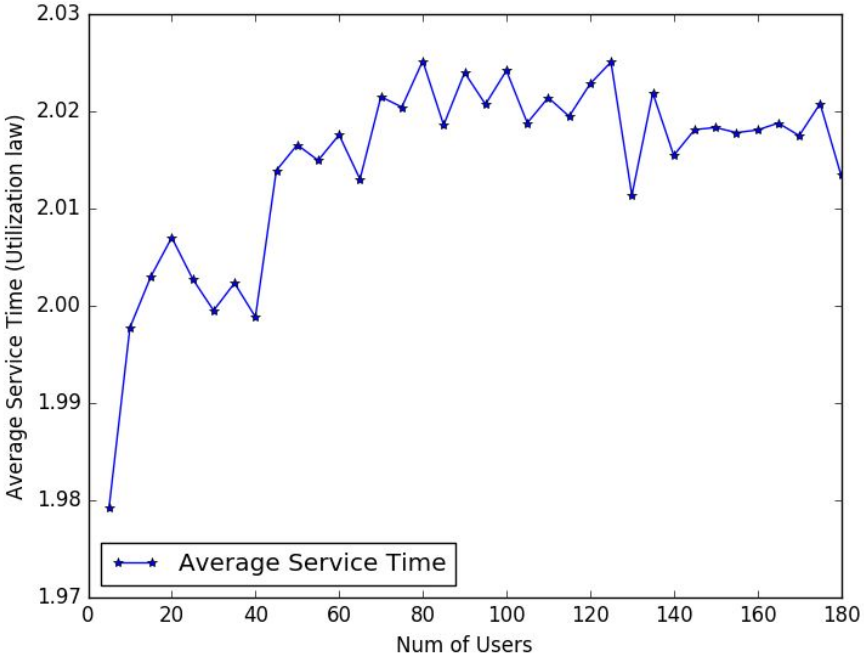
- Till 95 users, loss probability is 0
- The probability starts increasing after 100 users at steep rate
- The slope of graph decreases with increase in number of users
- On further increasing the number of users, losss probabilty will saturate at a value close to 0 when most of the requests are getting dropped

Utilization Law - Service Time

- Using Utilization Law

  `(Mean Service time) = (Core Utilization) / (Throughput)`
- Utilization and throughput graph are linear till saturation so we use utilization law to estimate service time
- Calculations for different data points give us a value close to 2 s
- This value is very close to the response time observed when the server is under-utilized when waiting time is almost zero

Utilization Law - Service Time



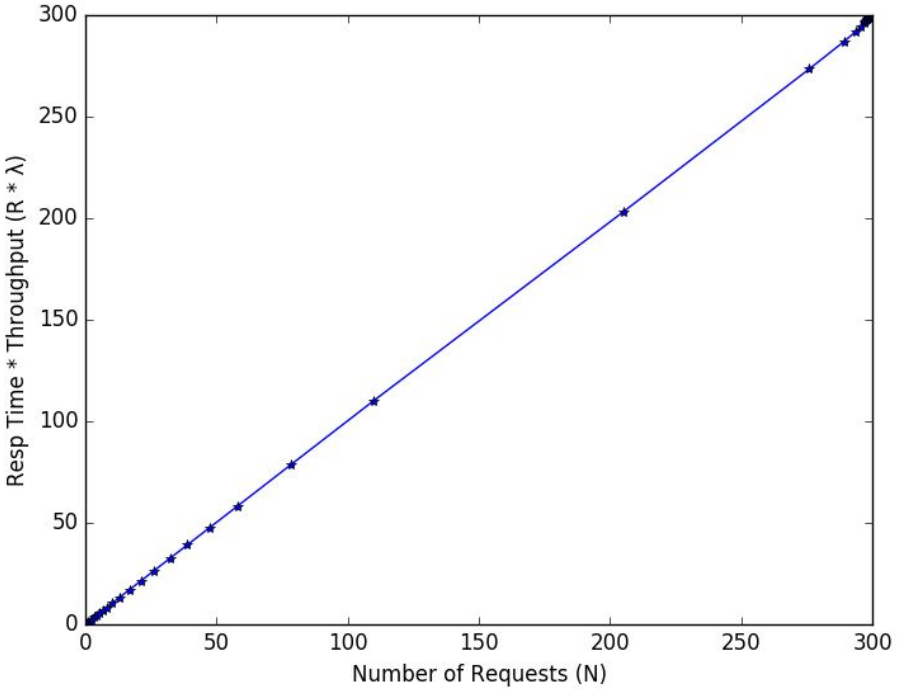
- The graph shows service time varies from 1.97 to 2.03 s
- As per the values set by us for service time distribution, the expected value of service time is 2 s which is close to the value we get from utilization law in all cases

Little's Law - Number of Requests

- Using Little's Law

  `(No. of requests in the system) = (Response Time) * (Throughput)`
- We plotted RHS vs LHS on a plot to verify Little's law

Little's Law - Number of Requests



- As expected, we got a linear line with a slope of 1

Conclusion

We simulated a web server sing discrete event simulation

The plots obtained from simulation results are consistent with the theoretically expected behaviour

We also verified that our experimental results satisfies the utilization and little’s law