# Assignment 2

Team Members
-Sridevi
-Helen Sathvika

# Closed System Configuration

ThinkTime = 2 secs

Service Time = 0.027 secs (exponential distribution)

Number of threads = 10

Number of cores = 1

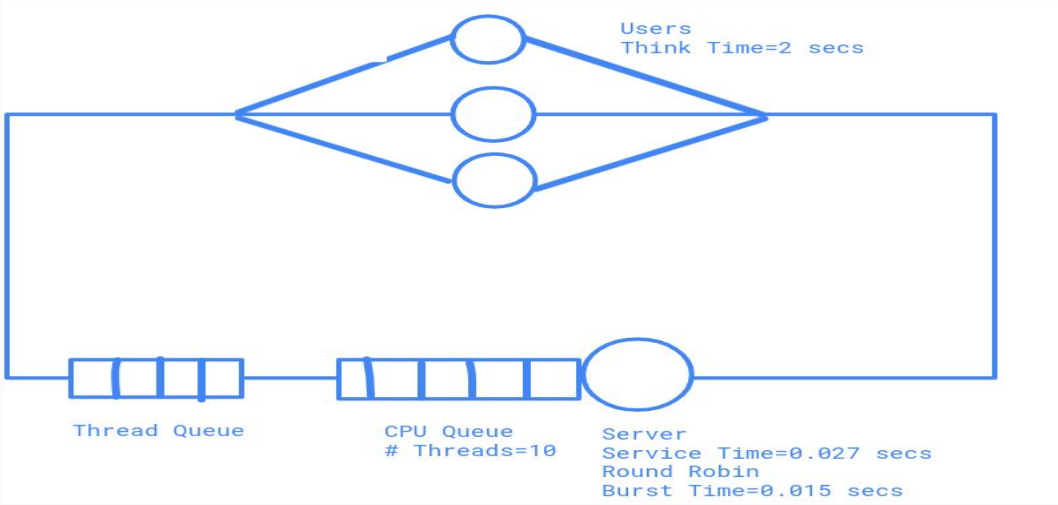
Buffer size=Infinity

Scheduling algorithm = Round Robin

Burst Time= 0.015 secs

Timeout = 0.27 secs  (exponential distribution)

Context switch overhead=1e-30



# Type of Events

Arrival to thread queue

Arrival to CPU

Schedule

Context Switch

Departure

# Classes

Simulation Class - Start simulation, pick the next event from event list and perform the event and output the performance metrics.

Users Class - Maintains thinktime for users.addUser method adds request to the event list after think time.

Server Class - All server events are handled.

Event Class - Maintains event details like event type, event time, core number .

Request Class - Maintains each request details like request ID, arrival time, timeout, remaining time, service time retry.

DS class - Maintains global data structures such as eventlist and simulation time.

# Simulation Class

Inputs - Number of cores, Number of threads, Burst time, Number of users, thinktime, service time, number of departures,seed.

Run method reads the input and starts the simulation.

ExecuteNextEvent method picks the peek element from the event list and based upon type of event , event handler is called.

Simulation time= Event time


After performing the event,the event object is removed from event list.

# Server Class

Arrival to the thread queue event handler:

If threads are available, arrival to cpu queue event is added to the event list else request is added to the arrival thread queue.

Arrival to the CPU event handler:

Request is removed from the arrival thread queue. If assigned core is free, schedule event is added to the event list else request is added to arrival CPU queue.

Schedule event handler:

Makes assigned core status as busy. If remaining time of the request is less than burst time then departure event is added to event list else context switch event is added to event list.

# Cont..

Context Switch event handler:

Context switch will happen and the schedule to assigned core is added to event list for next event in queue.

Departure event handler:

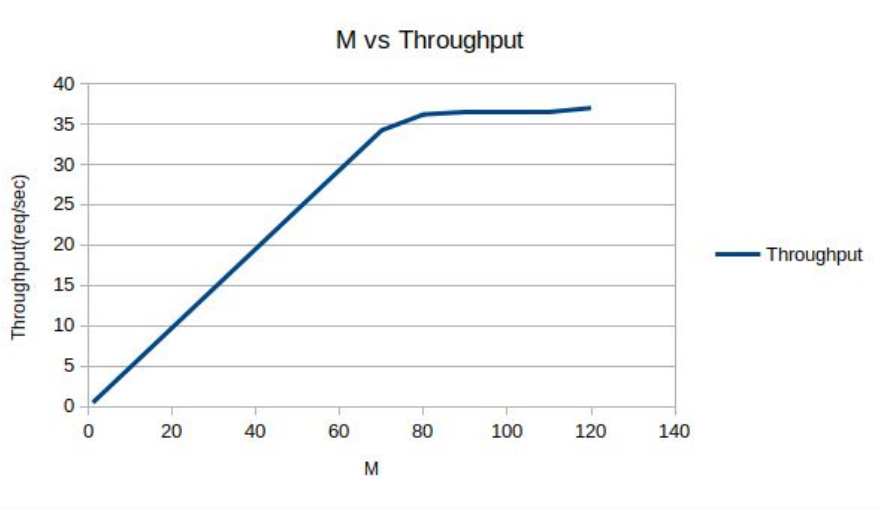
Response time,goodput and badput is calculated. Arrival to thread queue event is added to event list where event time is simulation time+thinktime and thread is released.

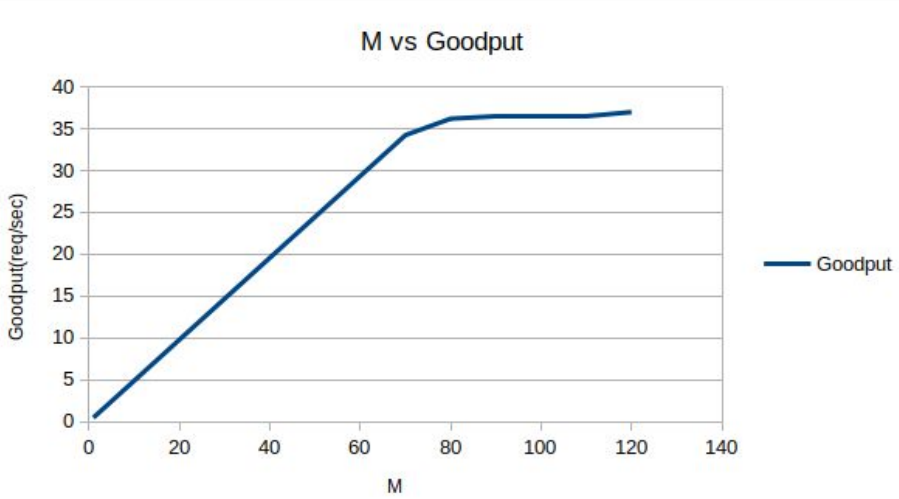
Timeout:

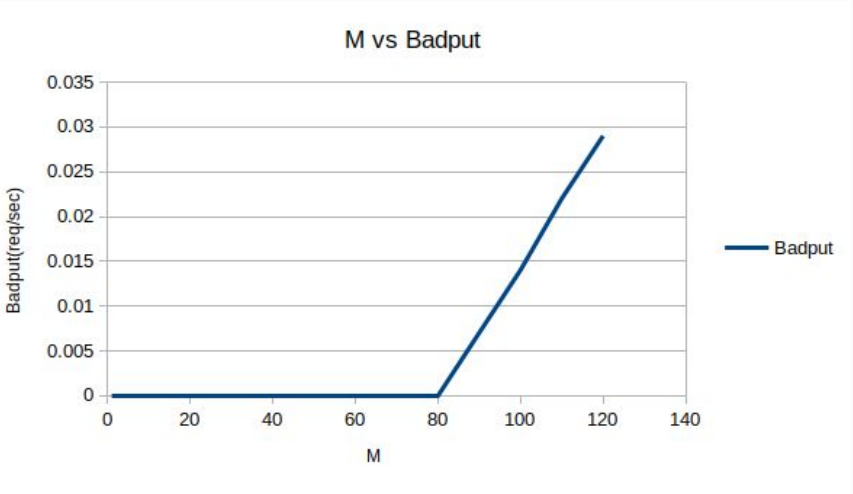
If timeout of a request is less than or equals to next event time, timeout happens and the users retry again until it reaches maximum number of retries.

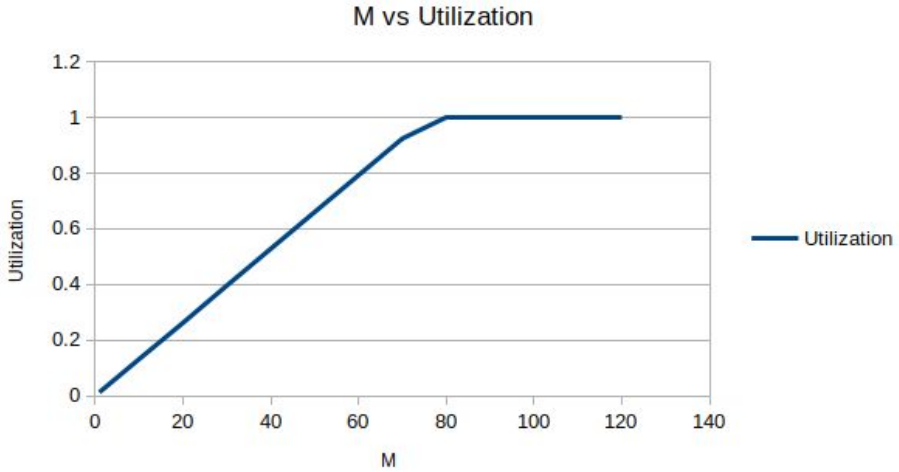
# Observed values:

| M | N | Goodput | Badput | Throughput | Utilization | Respnse Time | Drop rate |
|---|---|---|---|---|---|---|---|
| 1 | 0.0133 | 0.493 | 0 | 0.493 | 0.0133 | 0.0269 | 0 |
| 10 | 0.2 | 4.896 | 0 | 4.896 | 0.132 | 0.0352 | 0 |
| 20 | 0.407 | 9.787 | 0 | 9.787 | 0.262 | 0.0489 | 0 |
| 30 | 0.617 | 14.685 | 0 | 14.685 | 0.396 | 0.0735 | 0 |
| 40 | 0.824 | 19.579 | 0 | 19.579 | 0.528 | 0.0981 | 0 |
| 50 | 1.033 | 24.467 | 0 | 24.467 | 0.66 | 0.1265 | 0 |
| 60 | 1.242 | 29.347 | 0 | 29.347 | 0.792 | 0.161 | 0 |
| 70 | 1.453 | 34.229 | 0 | 34.229 | 0.924 | 0.2091 | 0 |
| 80 | 5.921 | 36.19 | 0 | 36.19 | 0.998 | 0.3243 | 0 |
| 90 | 9.985 | 36.44 | 0 | 36.44 | 0.998 | 0.5051 | 0.007 |
| 100 | 9.985 | 36.526 | 0 | 36.526 | 0.998 | 0.7401 | 0.014 |
| 110 | 9.985 | 36.536 | 0 | 36.536 | 0.998 | 0.9823 | 0.022 |
| 120 | 9.985 | 36.982 | 0 | 36.982 | 0.998 | 1.2261 | 0.029 |

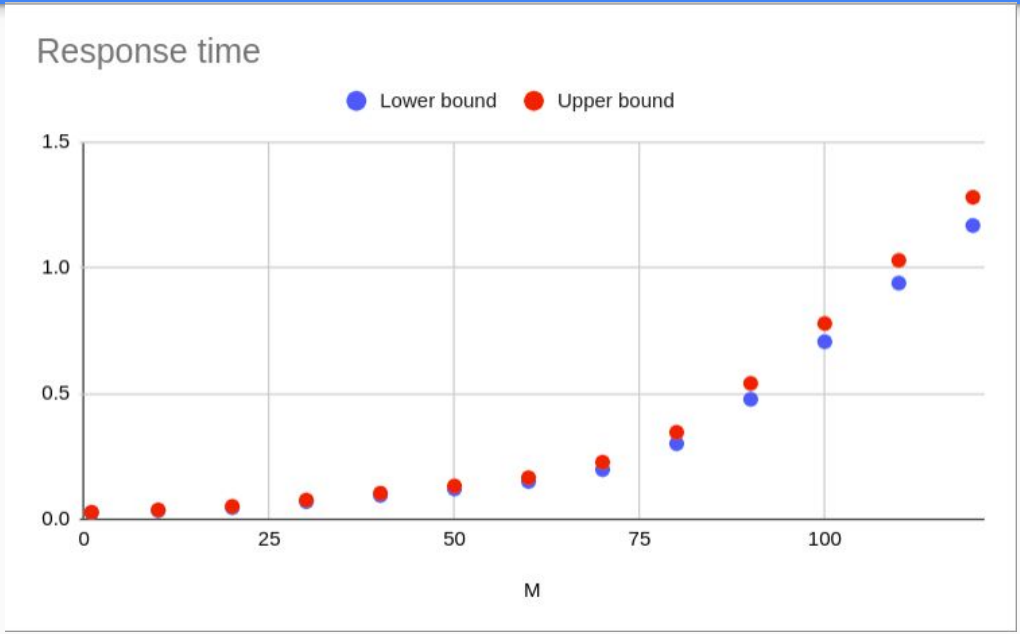

M vs Throughput
Throughput(req/sec)
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
120
140
M
Throughput


M vs Goodput
Goodput(req/sec)
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
120
140
M
Goodput

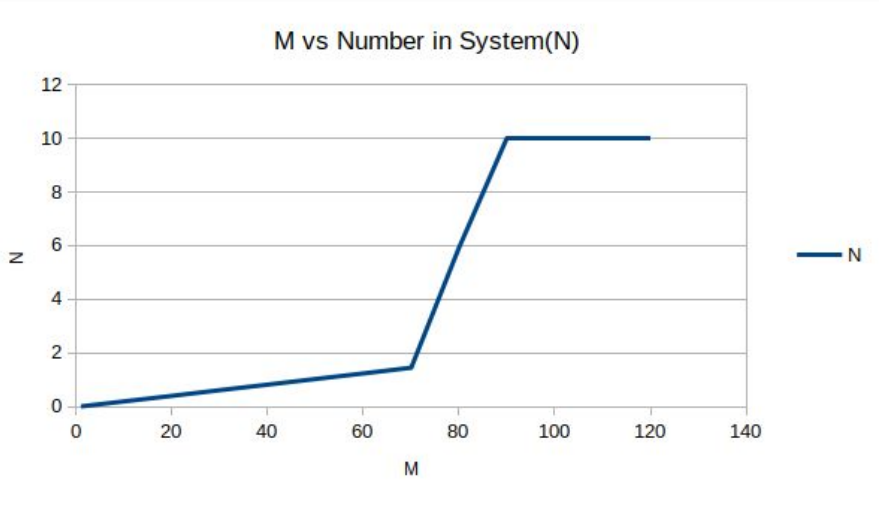

M vs Badput
Badput(req/sec)
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0
20
40
60
80
100
120
140
M
Badput

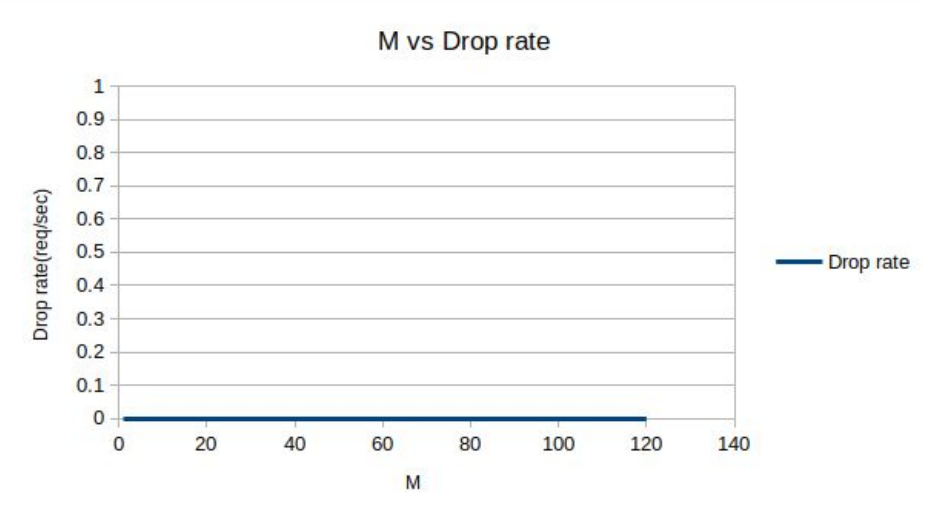

M vs Utilization
Utilization
0
0.2
0.4
0.6
0.8
1
1.2
0
20
40
60
80
100
120
140
M
Utilization

# Graph: Number of users(M) vs Response time




95% confidence interval for response time is plotted.


M vs Number in System(N)
12
10
8
6
4
2
0
N
0
20
40
60
80
100
120
140
M
N


M vs Drop rate
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Drop rate(req/sec)
0
20
40
60
80
100
120
140
M
Drop rate

# Results

The maximum number of users the system can support is approximately 80.

Throughput increases linearly upto 80 users and becomes constant after that.

The maximum throughput=  36.9

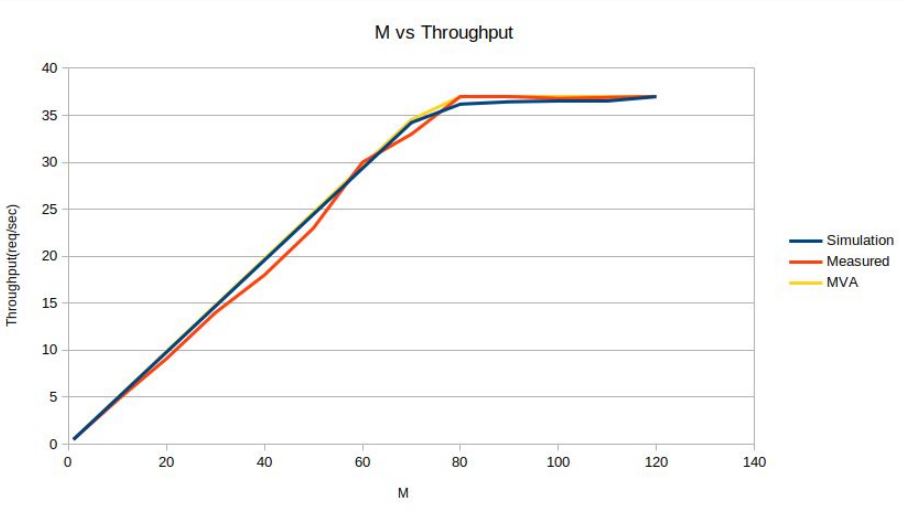
Utilization  increases linearly upto 80 users and becomes constant after that.

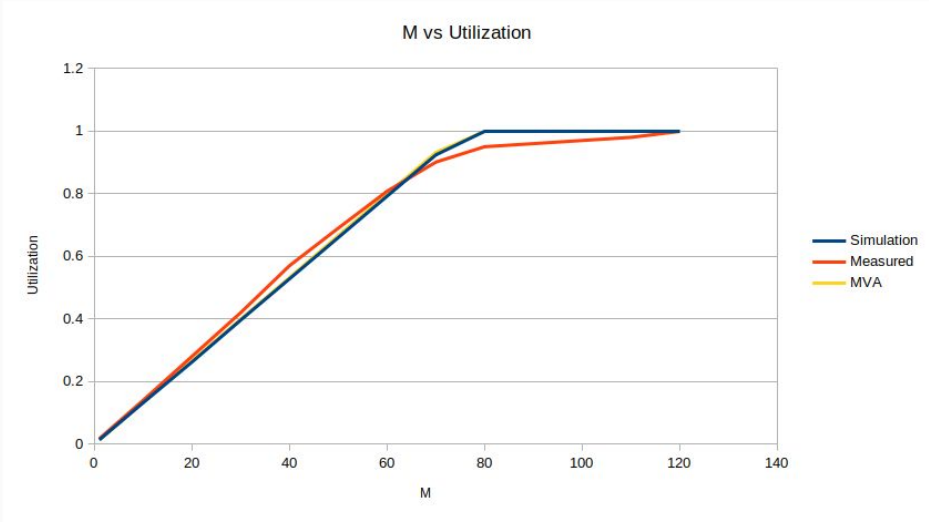
The maximum utilization=  99.8%

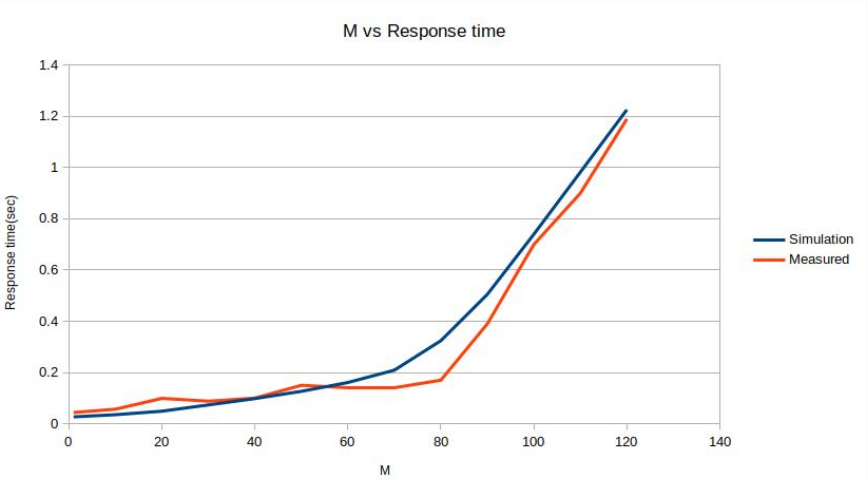
Response time increases with small intervals upto 80 users, then increases linearly

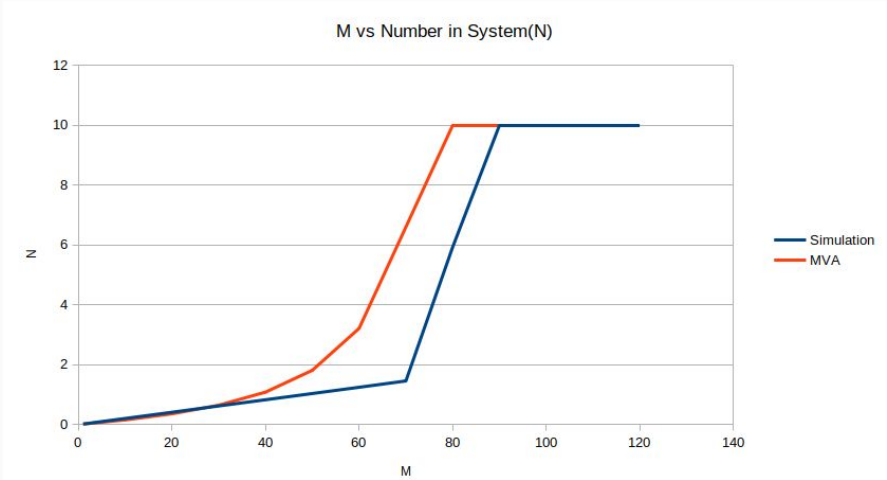
# Experiment-1

Does the simulation model generate results that imitates the results observed when measured?


M vs Throughput
Throughput(req/sec)
M
Simulation
Measured
MVA


M vs Utilization
Utilization
M
Simulation
Measured
MVA


M vs Response time
Response time(sec)
M
Simulation
Measured


M vs Number in System(N)
N
M
Simulation
MVA

# Observation

It is observed that the Simulation, MVA and Measured results for throughput, utilization, response time and number in system matches approximately.

# Link to MVA calculations

https://docs.google.com/spreadsheets/d/1rE4uYVLIyDGJ0rOsWIEKU4wFNlpBbjHsK4mIxhQJYGA/edit?usp=sharing

# Experiment 2

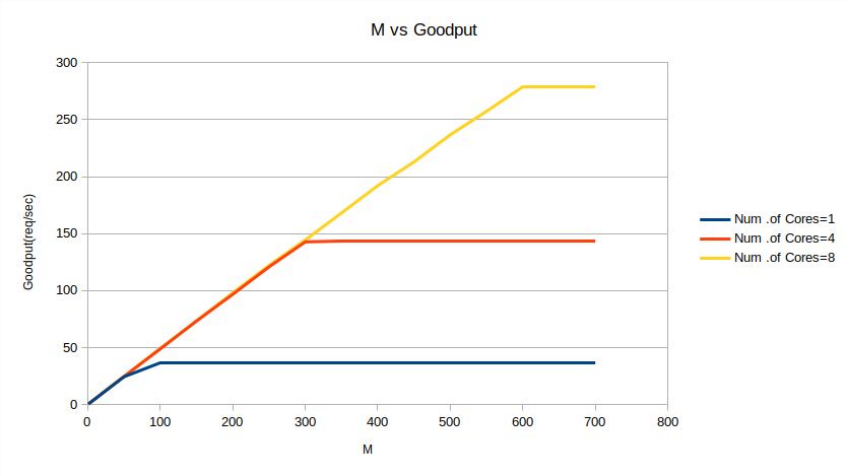
What if the number of cores increase?

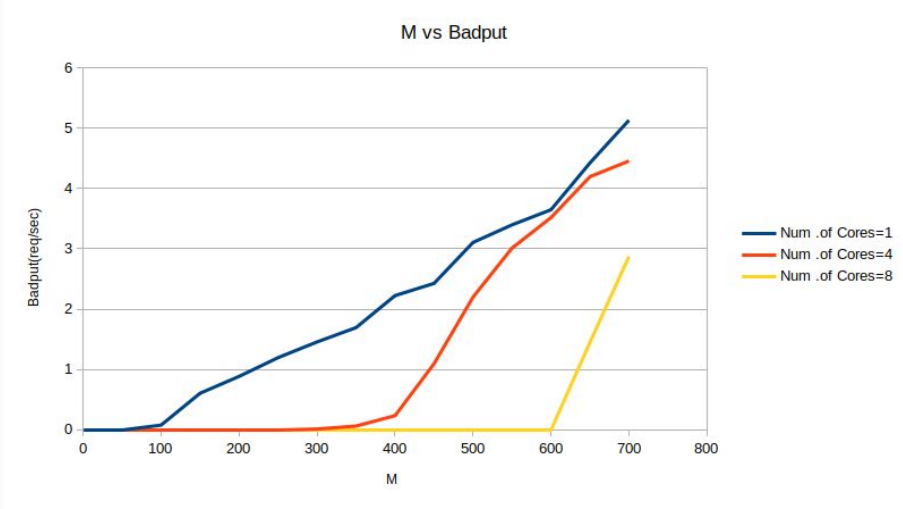
Expected behavior:

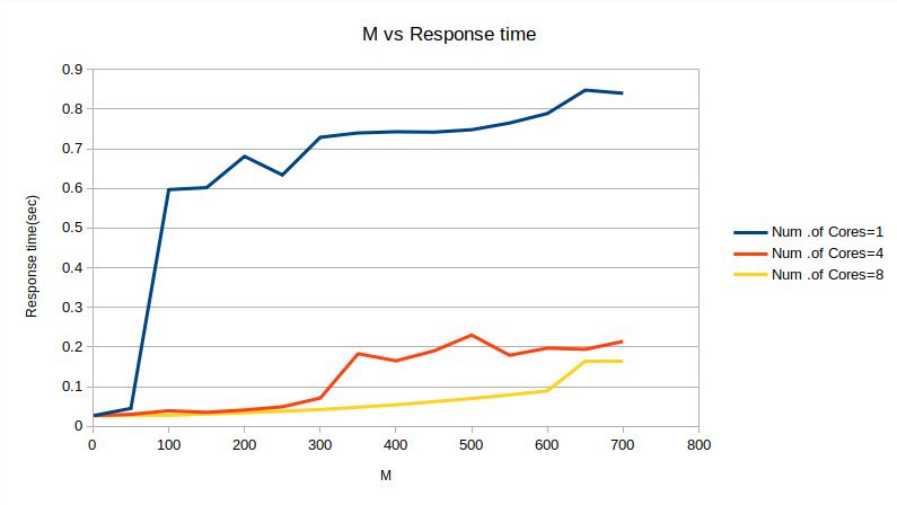
- Goodput increases
- Badput decreases
- CPU utilization decreases
- Response time decreases
- Number in system decreases

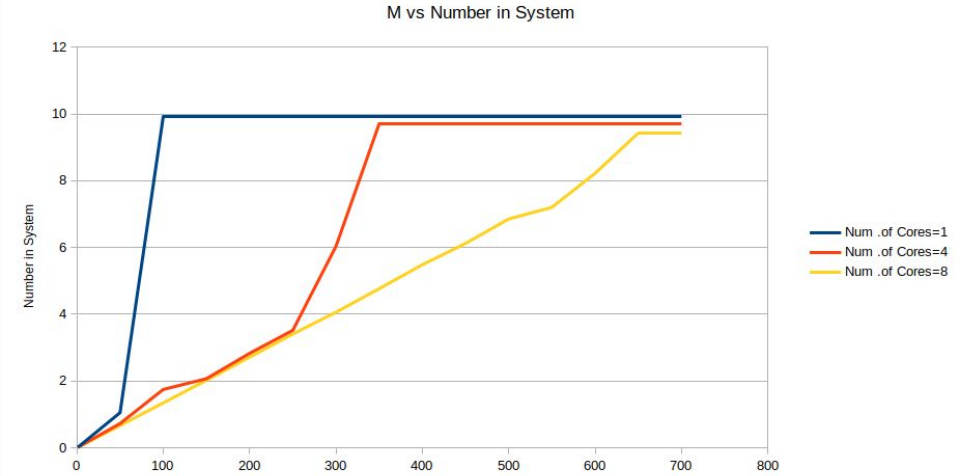
System Configurations:

- ThinkTime = 2 secs
- Service Time = 0.027 secs (exponential distribution)
- Number of threads = 10
- Number of cores = 1,4,8
- Scheduling algorithm = Round Robin
- Burst Time= 0.015 secs
- Timeout = 0.27 secs (exponential distribution)
- Context switch overhead=1e-30


M vs Goodput
Goodput(req/sec)
M
Num .of Cores=1
Num .of Cores=4
Num .of Cores=8


M vs Badput
Badput(req/sec)
M
Num .of Cores=1
Num .of Cores=4
Num .of Cores=8


M vs Response time
Response time(sec)
M
Num .of Cores=1
Num .of Cores=4
Num .of Cores=8


M vs Number in System
Number in System
Num .of Cores=1
Num .of Cores=4
Num .of Cores=8


## Observed:

When number of cores increase, the requests are distributed across all the cores. So

- Goodput increases
- Badput decreases
- CPU utilization decreases
- Response time decreases
- Number in system decreases

# Experiment 3

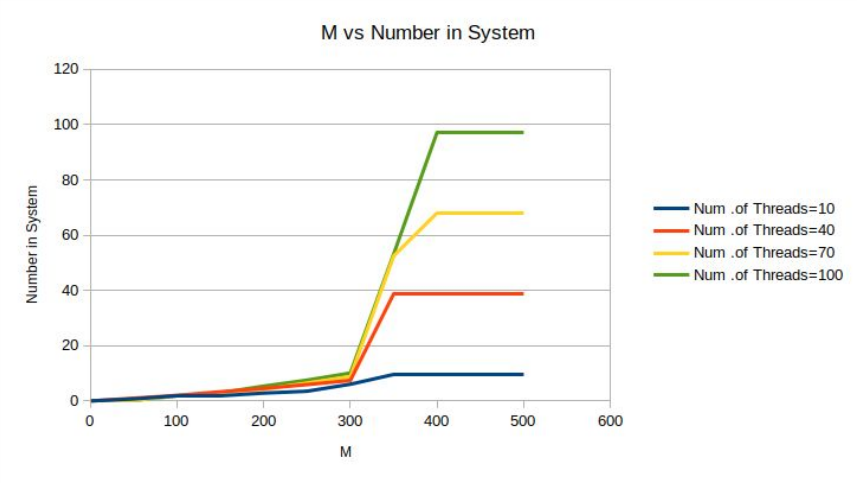
What if the number of threads varies?

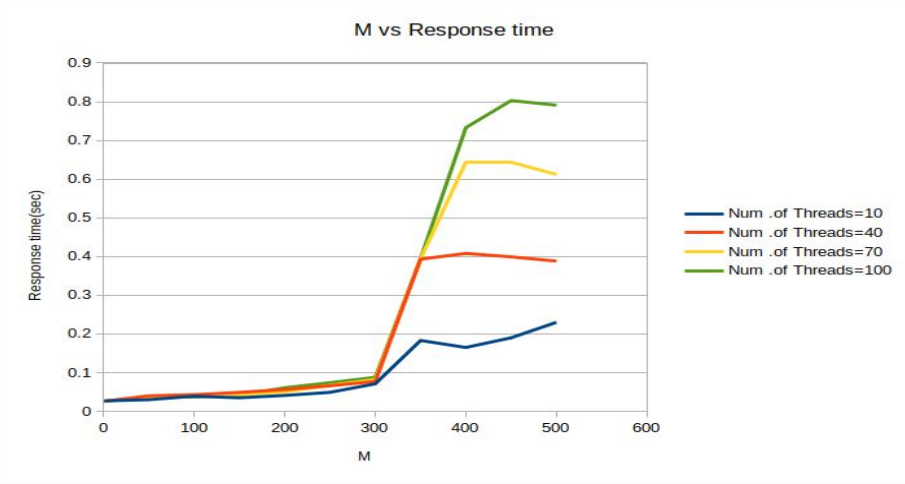
Expected behavior(with increase in threads):

- Response time increases
- Number in system increases
- Badput increases

System Configurations:

- ThinkTime = 2 secs
- Service Time = 0.027 secs (exponential distribution)
- Number of threads = 10,40,70,100
- Number of cores = 4
- Scheduling algorithm = Round Robin
- Burst Time= 0.015 secs
- Timeout = 0.27 secs (exponential distribution)
- Context switch overhead=1e-30

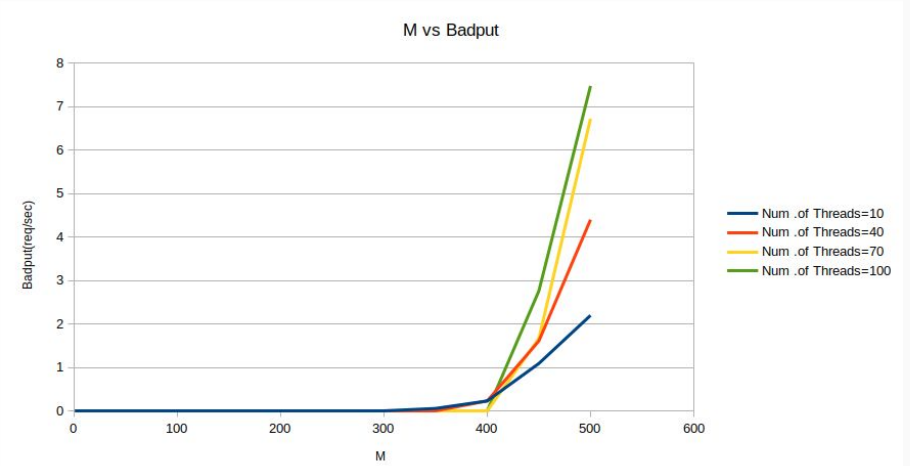



## Observed:

When number of number of threads increase, the number of requests in the system will increase. Number of context switches between 2 execution of a process will increase. So

- Response time increases
- Number in system increases
- Badput increases



# Experiment 4

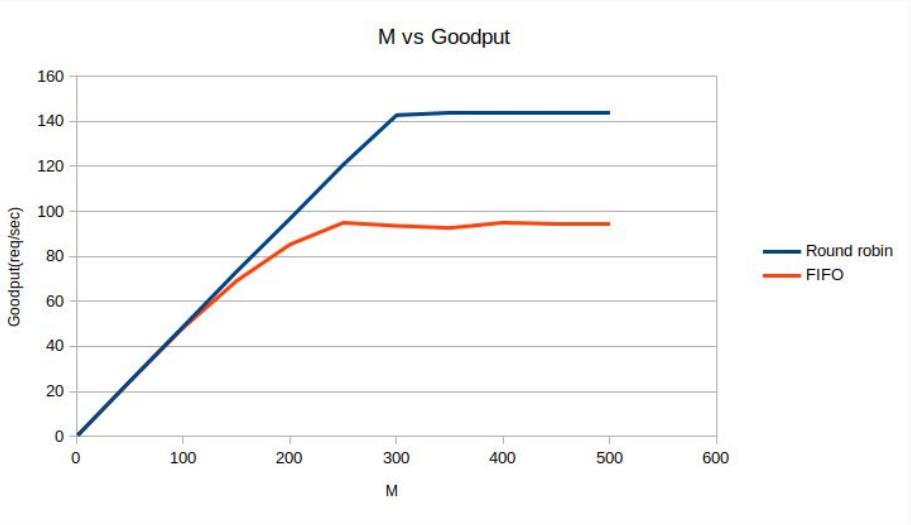
What if the scheduling policy is FIFO

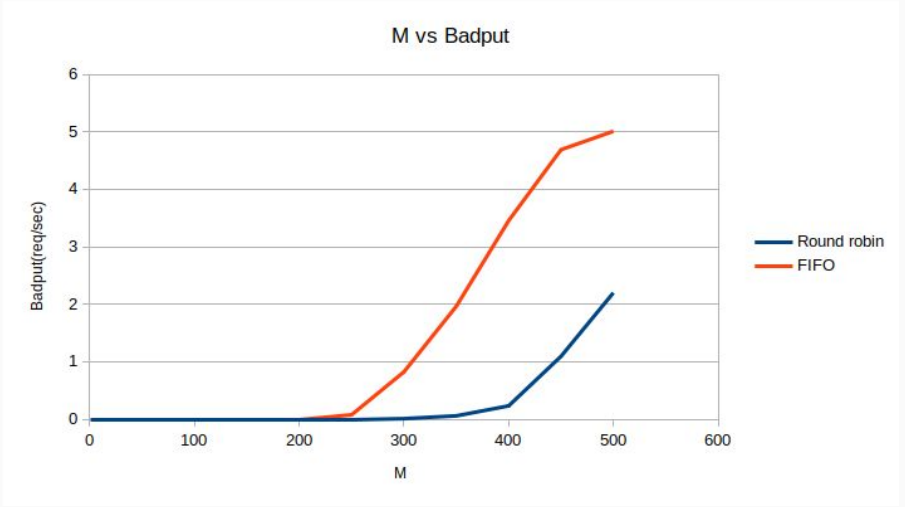
Expected behavior:

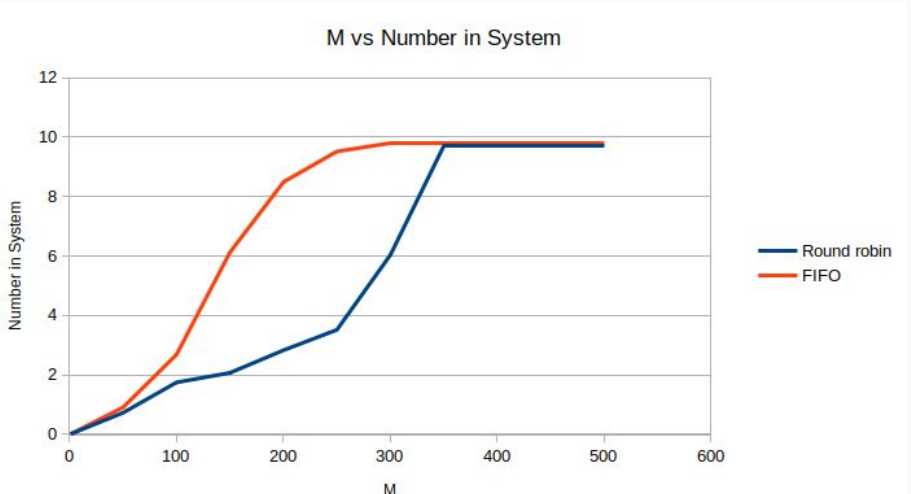
- Response time increases
- Number in system increases
- Badput increases
- Goodput may decrease

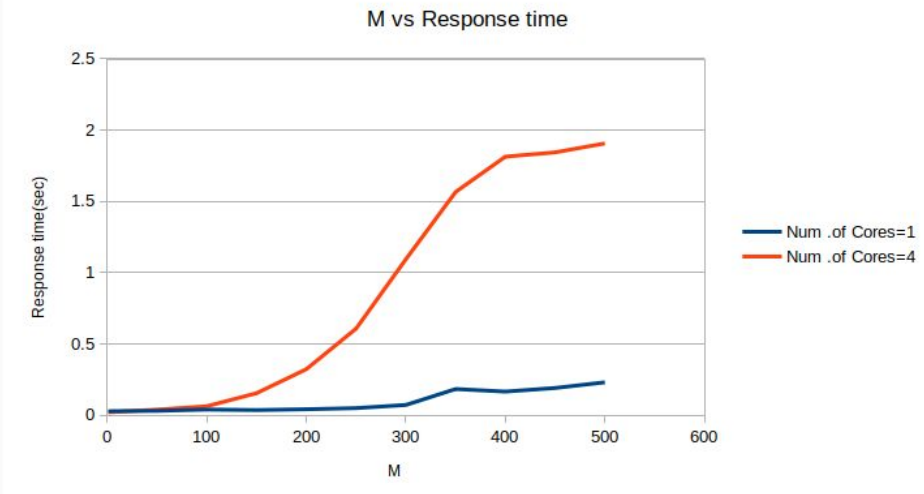
System Configurations:

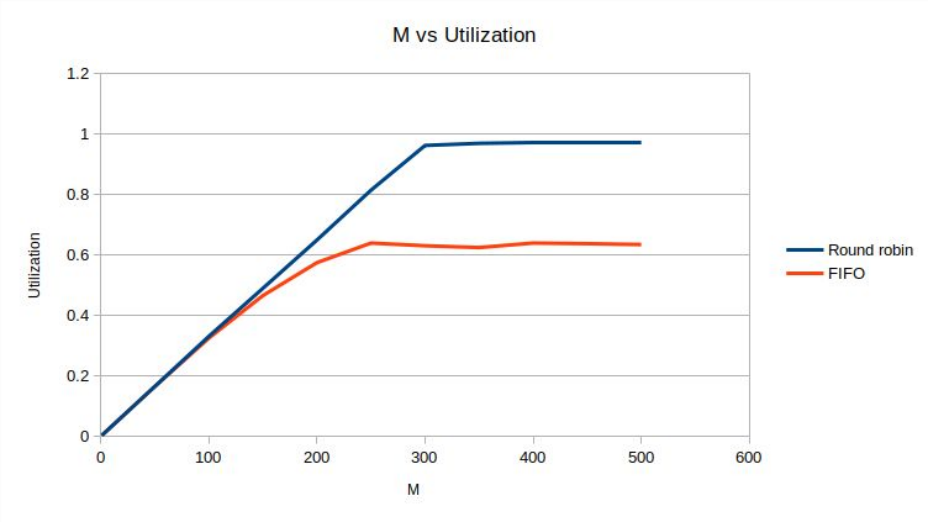
- ThinkTime = 2 secs
- Service Time = 0.027 secs (exponential distribution)(70% of times, mean service time is 0.001 and 30% of the times, mean service time is 0.087)
- Number of threads = 10
- Number of cores = 4
- Scheduling algorithm = Round Robin, FIFO
- Burst Time= 0.015 secs
- Timeout = 0.27 secs (exponential distribution)
- Context switch overhead=1e-30


M vs Goodput
Goodput(req/sec)
M
Round robin
FIFO


M vs Badput
Badput(req/sec)
M
Round robin
FIFO


M vs Number in System
Number in System
M
Round robin
FIFO


M vs Response time
Response time(sec)
M
Num .of Cores=1
Num .of Cores=4




70% of times, mean service time is 0.001 and 30% of the times, mean service time is 0.087)

## Observed:

When the request with high service service time is getting served, the requests in queue have to wait for longer duration. So

- Goodput decreases
- Badput increases
- Response time increases
- Number in system increases
- Utilization decreases

# THANK YOU