VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING

SYSTEM PERFORMANCE EVALUATION

Topic

EVALUATE M/M/1/B MODEL

Lecturer:	PhD.Le Hong Trang
	PhD.Tran Van Hoai
Student:	Hoang Tran Viet Long
	1652350
	Ngo Tu Duy
	1752133

HO CHI MINH, 11/2019

Contents

In this semester, we are required to work on a Simulation of single queue by **Simpy** based on **Python** language. We are also required to do the assignment with the mix of as many as possible techniques that we already studied on the lecture time to analyze the model to get further meaning of some real problems.

1 Introduction

1.1 Topic

The assignment required us to build a simple simulation system to measure the performance of M/M/1/B=n queue.

The simplest queue model consists of a single server in which the service times are i.i.d. Exponential random variables with mean $1/\mu$, and the customers(processes, jobs) arrive into the system according to a Poisson process with rate λ. Such a system is referred to as an M/M/1/B=n (limit the number of buffers in the ready queue) queue system, the figure below shows the main ideal:

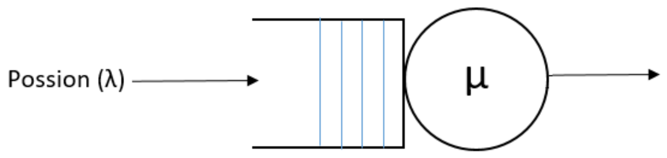


Figure 1: M/M/1 Queue model

M/M/1 is Kendall notation and describes the queueing system architecture. "M" stands for "memoryless" and it mean the arrival times are Exponentially distributed, the second slot of "M" stands for Exponential distribution of service times. "1" is the number of server and "B = n" is the number of slots in buffer.

1.2 Simpy introduction

Simpy is a process-based discrete-event simulation framework based on standard Python.

Processes in SimPy are defined by Python generator functions and may, for example, be used to model active components like customers, vehicles or agents. SimPy also provides various types of shared resources to model limited capacity congestion points (like servers, checkout counters and tunnels).

2 Methodology

2.1 Overview

In general, we are using Python 3.2.3 version on Ubuntu 12.04 to implement the project.

2.1.1 Goals

• Goal 1: Simulate a simple queue that can be using in real life base on M/M/1/B = n queueing technique.
• Goal 2: An overview of a system with M/M/1/B = 1 which affects by service rate and arrival rate.

2.1.2 Service and Outcomes

The system that we will implement handled that jobs generators and control the served of the system. The outcome that a job (customer) is served or rejected.

2.1.3 Metric selection

In this assignment, we are focus on cases that the system working correctly, but we are also listing all metric that we assumed possible to implement in the future:

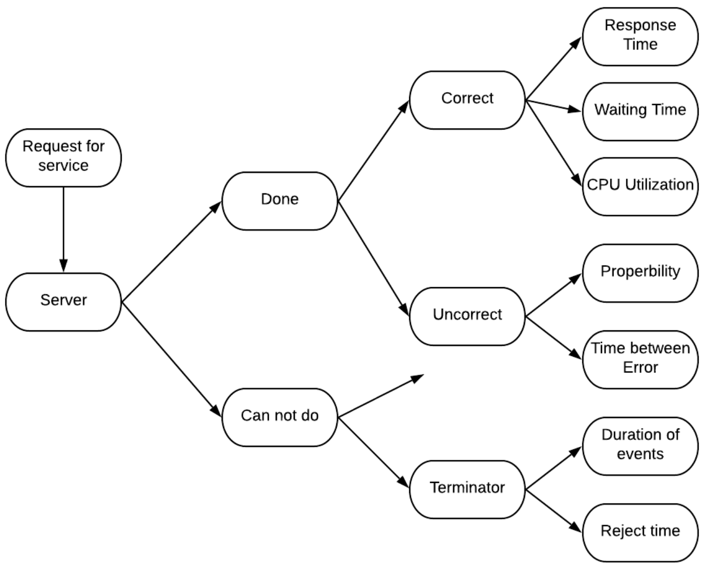


Figure 2: Metric selecting chart

2.1.4 Parameters

We choose some crucial parameter to test such as Mean Service Time, Mean Waiting Time and Mean Queue Length (**Terminating Simulation**). The factors that system focus on are service rate, arrival rate, arrival time, burst time, number of customer, number of buffer,...

2.1.5 Factors

A process is considered as a customer and a time that customer arrives and the time that the system need to serve them is randomly choose, but it is fitted with the service rate and arrival rate that given at the begin of system.

2.1.6 Evaluation technique

A simulation program using simpy base Python 3 will be applied for the system.

2.1.7 Workload

We are mainly generate the workload of the system is *addition instruction*. In particular, workload selection is included Services Exercises (for system instruction) with SUT is the queue system and CUT is Average waiting time, average response time, number of jobs in queue with real time and CPU Utilization.

2.1.8 Analyze and Interpret data

We applied the **Model verification techniques** in the Analysis of Simulation results. We focus on *Top-Down Modular Design, Structured Walk-Through* and *Run Simplified cases*

2.1.9 Present result

The model is using One-Line graphics displays, Continuity test, Degeneracy Tests and Seed Independence techniques to test the system.

We also count the time that system working, IDLE then calculate the CPU Utilization then display as a pie chart for consume some comparition.

2.2 First-Come First-Served (FCFS) algorithm

First in, first out (FIFO) is also know as First come, First served (FCFS) is the simplest algorithm. FCFS is simple queue processes. In this, the process comes first will be executed first, and next process start only when the previous one finishes the execution.

The advantages of this algorithm is very simple and easy to implement.

Meanwhile, there are some disadvantages of this algorithm, like:

- The scheduling method is non preemptive, the process will run to the completion.
- Problem of starvation may occur.
- Low performance because the average waiting time may be higher than other algorithm.

2.3 Average response time calculation

Response time is defined as time from the first time that a process get a request for CPU then entering the ready queue.

Average response time is the average response time of all process that have been executed.

Max response time is the highest value of processes's response time after filter.

$$\boxed{\text{Response Time} = \text{Time process first get CPU - arrival time}}$$

2.4 Average waiting time calculation

Waiting time is stood for the time that a process stay at a ready queue before moing into run queue.

Average waiting time is considered as average time calculation of multiple processes's waiting time of a CPU.

$$\boxed{\text{Waiting time} = \text{Turn around time - Burst time}}$$

2.5 CPU Utilization

CPU Utilization refers to a computer's usage of processing resources, or the amount of work handled by a CPU. It is also used to estimate system performance. CPU utilization can vary according to the type and amount of computing task because some tasks require heavy CPU time while others require less CPU time. Process time is another name for CPU time and is the amount of time used by a CPU for processing instruction of an operating system or a computer program. CPU time is quantified in clock ticks or seconds. CPU utilization shows the burden on a processor in terms of percentage that indicates if any changes are to be made in the system otherwise it may get exhausted of capacity.

$$\boxed{\text{U} = \frac{busy\_time}{busy\_time + Idle\_time}}$$

2.6 Examples

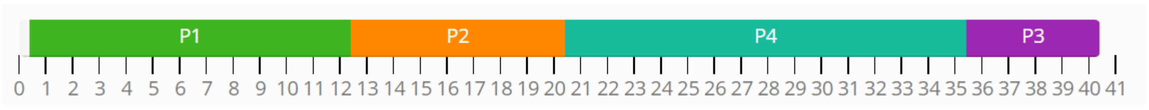


Figure 3: Gantt chart

Then we have:

Process	Arrival time	Burst time
P1	0.4	12
P2	1	8
P3	5	5
P4	2	15

Table 1: Example.

Process	Arrival time	Burst time	Turn around time	Response time	Waiting time
P1	0.4	12	12	0	0
P2	1	8	19.4	11.4	11.4
P3	5	5	35.4	18.4	18.4
P4	2	15	33.4	30.4	30.4
Average				19.8	19.8

3 Implementation and Result

3.1 Flow chart

The system working will base on the following flow chart:

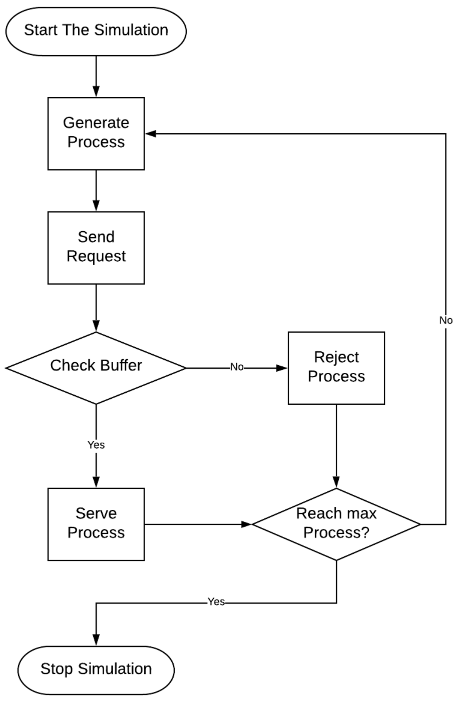


Figure 4: System working flow chart

3.2 Generator processing

Basing on the factors that the simulation is given, a process (job) contains of several of properties, in detail, a job have been generate randomly with specific arrival time, service time and there ID number (come fist then higher ID number).

Service rate and Arrival rate are mainly factors affect on generator process.

3.3 Server processing

3.3.1 Ready queue function

A single queue and single server simulation are created, Generator creates multiple job coming with different values of properties, the function we implement can be describes as 2 main purpose:

• Push: When a new job occur, that one will be added into a ready queue if and only if the number of free buffer in the queue if it is satisfied: number of buffer different than 0 and free buffer greater than 0. The higher ID job, the lower priority (because FCFS algorithm), if not the system rejects the job. For example: if the ready queue contains job: 1, 2, 3, 4, 5 and number of buffer equal 5, job 6 coming but the service of job 1 does not finish yet, job 6 will be rejected.

• Pop: When a job is completely served by the system, it will be eliminated from the ready queue and free a slot for a new one.

The number of rejected job and completed job will be counted at the end of the simulation.

3.3.2 Waiting

There is a flag to determine that the system is busy or not. When a new job served, system will turn on the flag to defend other job in the ready queue run. Otherwise, if a job occur while the flag turn off, other job may attach to the system to require that it is going to run, race condition occur,

System can waiting the occurrence of job if the queue is empty, the pseudo code below demonstrate the class Server:

```
class Server:
     def serve():
          loop forever:
                    if (ready queue full)
                    reject job
               else
                           push job in ready queue
               if (ready queue empty)
                           serve_wait()
               else
                    pop a job and serve it
                           job_wait()

                if (reach max job && complete final job)
                    break
```

Figure 5: Pseudo code for Server

3.4 Analysis function

And the end of file *Assignment.py* we are built a class to analysis the result that the system have been run before. In more detail, the class define some function to calculate: **completed jobs/total jobs, response time, waiting time, IDLE time, CPU time CPU Utilization**.

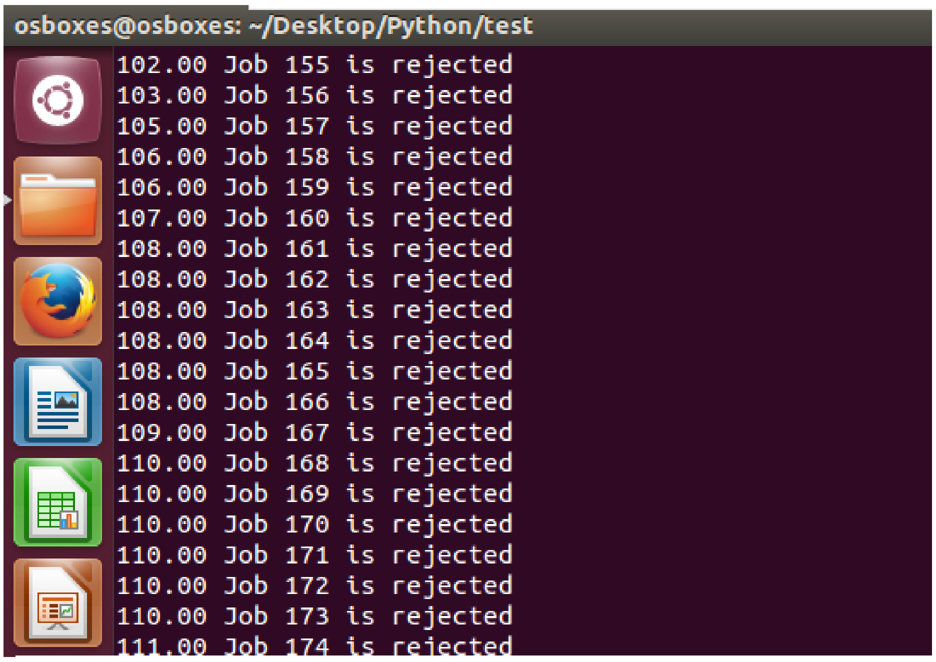
Otherwise, this class also provides method to draw charts, including: **Waiting time, response time** and **Number of jobs in ready queue at a time.**

3.5 Testing

3.5.1 Degeneracy Test

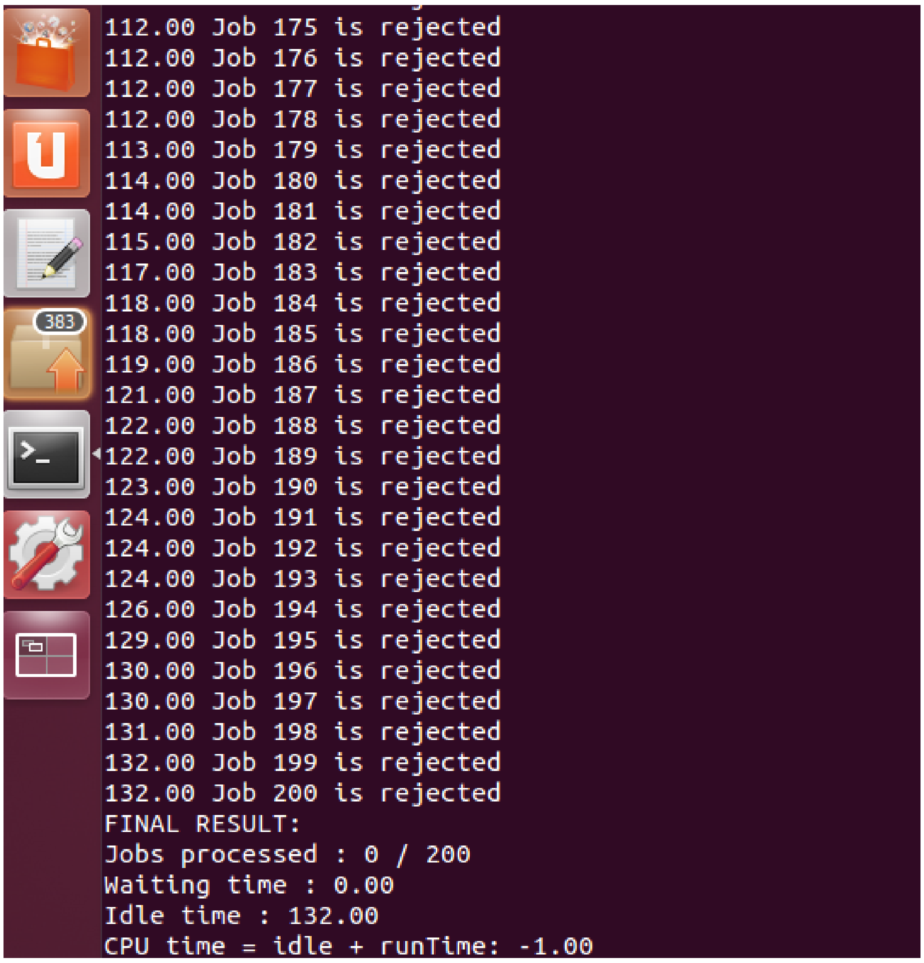
Degeneracy tests consist of checking that the model works for extreme (lowest or highest allowed)values of system, configuration, or workload parameters. Although these extreme cases may not represent a typical case, they help discover bugs that the analyst would not have thought of otherwise. It is useful to incorporate checks for input parameter values and to verify that they are within the allowed bounds. The analyst should have verified that the model works for any combination of these allowed bounds.

We are testing in the lowest case, buffer = 0, number of job = 200:



```
osboxes@osboxes: ~/Desktop/Python/test
102.00 Job 155 is rejected
103.00 Job 156 is rejected
105.00 Job 157 is rejected
106.00 Job 158 is rejected
106.00 Job 159 is rejected
107.00 Job 160 is rejected
108.00 Job 161 is rejected
108.00 Job 162 is rejected
108.00 Job 163 is rejected
108.00 Job 164 is rejected
108.00 Job 165 is rejected
108.00 Job 166 is rejected
109.00 Job 167 is rejected
110.00 Job 168 is rejected
110.00 Job 169 is rejected
110.00 Job 170 is rejected
110.00 Job 171 is rejected
110.00 Job 172 is rejected
110.00 Job 173 is rejected
111.00 Job 174 is rejected
```

Figure 6: Degeneracy Test



```
112.00 Job 175 is rejected
112.00 Job 176 is rejected
112.00 Job 177 is rejected
112.00 Job 178 is rejected
113.00 Job 179 is rejected
114.00 Job 180 is rejected
114.00 Job 181 is rejected
115.00 Job 182 is rejected
117.00 Job 183 is rejected
118.00 Job 184 is rejected
118.00 Job 185 is rejected
119.00 Job 186 is rejected
121.00 Job 187 is rejected
122.00 Job 188 is rejected
122.00 Job 189 is rejected
123.00 Job 190 is rejected
124.00 Job 191 is rejected
124.00 Job 192 is rejected
124.00 Job 193 is rejected
126.00 Job 194 is rejected
129.00 Job 195 is rejected
130.00 Job 196 is rejected
130.00 Job 197 is rejected
131.00 Job 198 is rejected
132.00 Job 199 is rejected
132.00 Job 200 is rejected
FINAL RESULT:
Jobs processed : 0 / 200
Waiting time : 0.00
Idle time : 132.00
CPU time = idle + runTime: -1.00
```

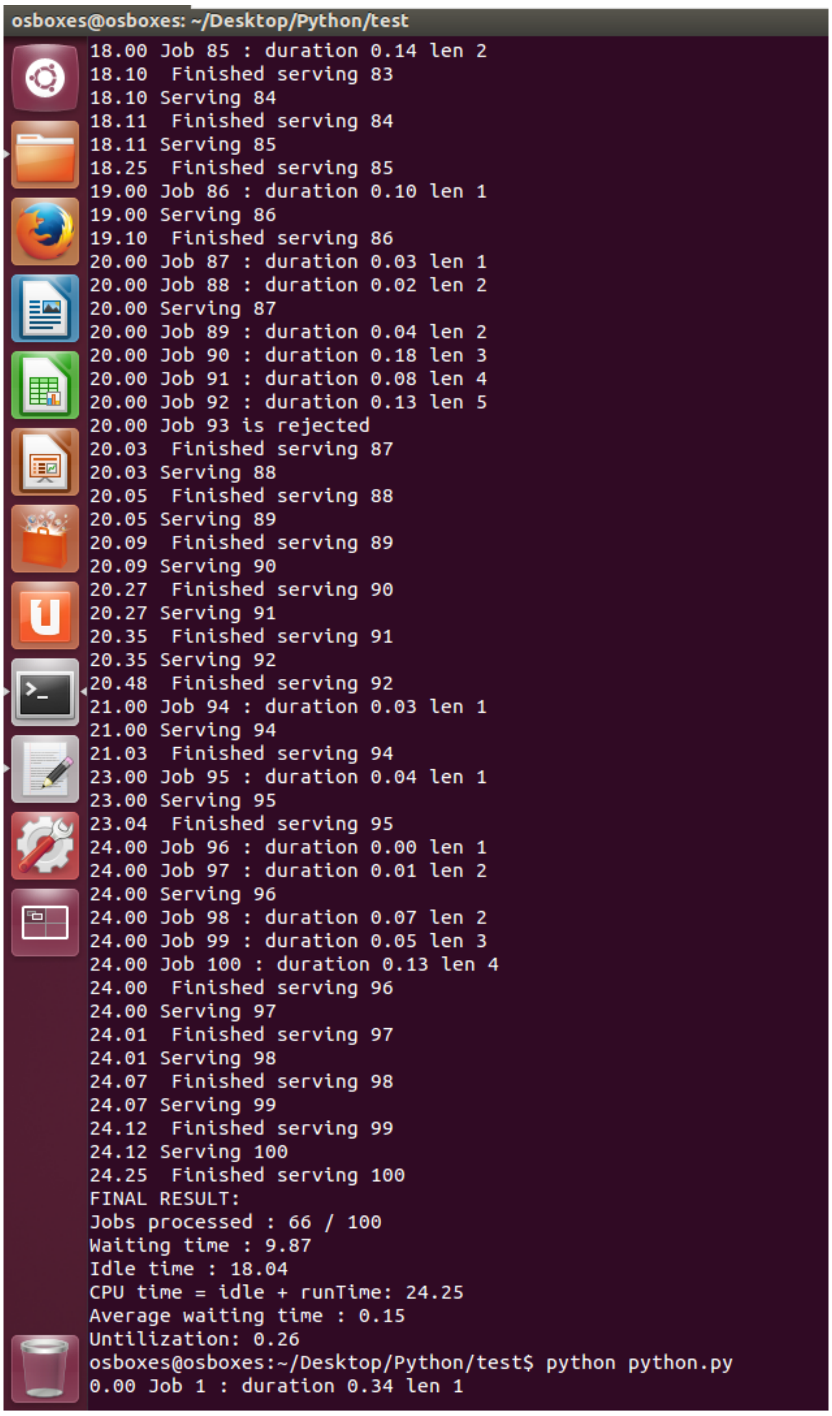
Figure 7: Degeneracy Test

3.5.2 Continuity Test

Continuity tests consist of running the simulation several times for slightly different values of input parameters. For any one parameter, a slight change in input should generally produce only a slight change in the output. Any sudden changes in the output should be investigated. Often, they are due to modeling error. In this kind of test, We run with different values of service rate, arrival rate, number of buffers and number of jobs.

- Case:

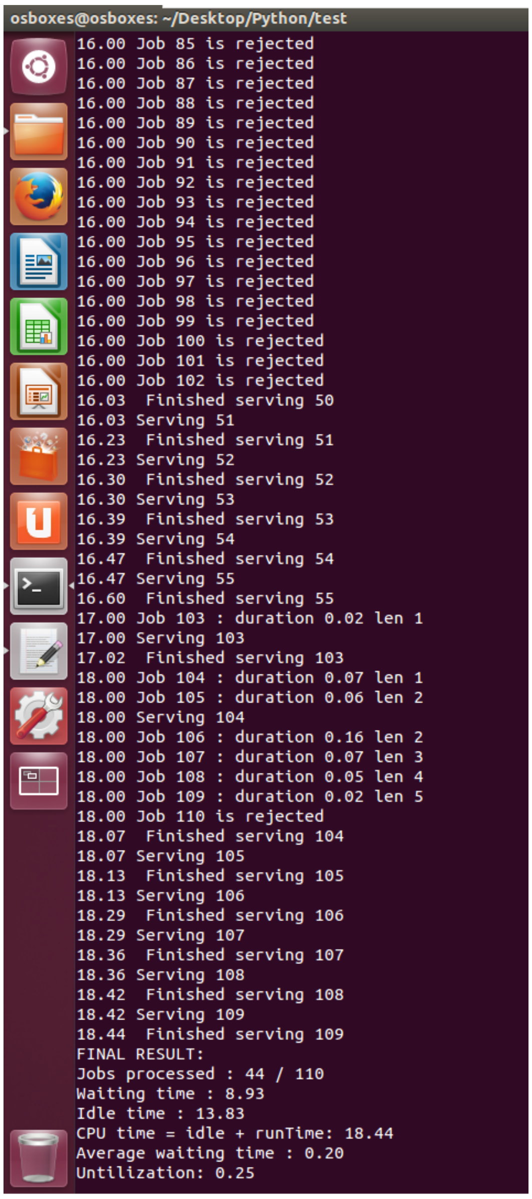
  Service rate = arrival rate = 0.1, number of buffers = 5, number of jobs = 100:


osboxes@osboxes: ~/Desktop/Python/test

```
18.00 Job 85 : duration 0.14 len 2
18.10  Finished serving 83
18.10 Serving 84
18.11  Finished serving 84
18.11 Serving 85
18.25  Finished serving 85
19.00 Job 86 : duration 0.10 len 1
19.00 Serving 86
19.10  Finished serving 86
20.00 Job 87 : duration 0.03 len 1
20.00 Job 88 : duration 0.02 len 2
20.00 Serving 87
20.00 Job 89 : duration 0.04 len 2
20.00 Job 90 : duration 0.18 len 3
20.00 Job 91 : duration 0.08 len 4
20.00 Job 92 : duration 0.13 len 5
20.00 Job 93 is rejected
20.03  Finished serving 87
20.03 Serving 88
20.05  Finished serving 88
20.05 Serving 89
20.09  Finished serving 89
20.09 Serving 90
20.27  Finished serving 90
20.27 Serving 91
20.35  Finished serving 91
20.35 Serving 92
20.48  Finished serving 92
21.00 Job 94 : duration 0.03 len 1
21.00 Serving 94
21.03  Finished serving 94
23.00 Job 95 : duration 0.04 len 1
23.00 Serving 95
23.04  Finished serving 95
24.00 Job 96 : duration 0.00 len 1
24.00 Job 97 : duration 0.01 len 2
24.00 Serving 96
24.00 Job 98 : duration 0.07 len 2
24.00 Job 99 : duration 0.05 len 3
24.00 Job 100 : duration 0.13 len 4
24.00  Finished serving 96
24.00 Serving 97
24.01  Finished serving 97
24.01 Serving 98
24.07  Finished serving 98
24.07 Serving 99
24.12  Finished serving 99
24.12 Serving 100
24.25  Finished serving 100
FINAL RESULT:
Jobs processed : 66 / 100
Waiting time : 9.87
Idle time : 18.04
CPU time = idle + runTime: 24.25
Average waiting time : 0.15
Untilization: 0.26
osboxes@osboxes:~/Desktop/Python/test$ python python.py
0.00 Job 1 : duration 0.34 len 1
```

Figure 8: 100 jobs, 5 buffers, $\mu = \lambda = 0.1$

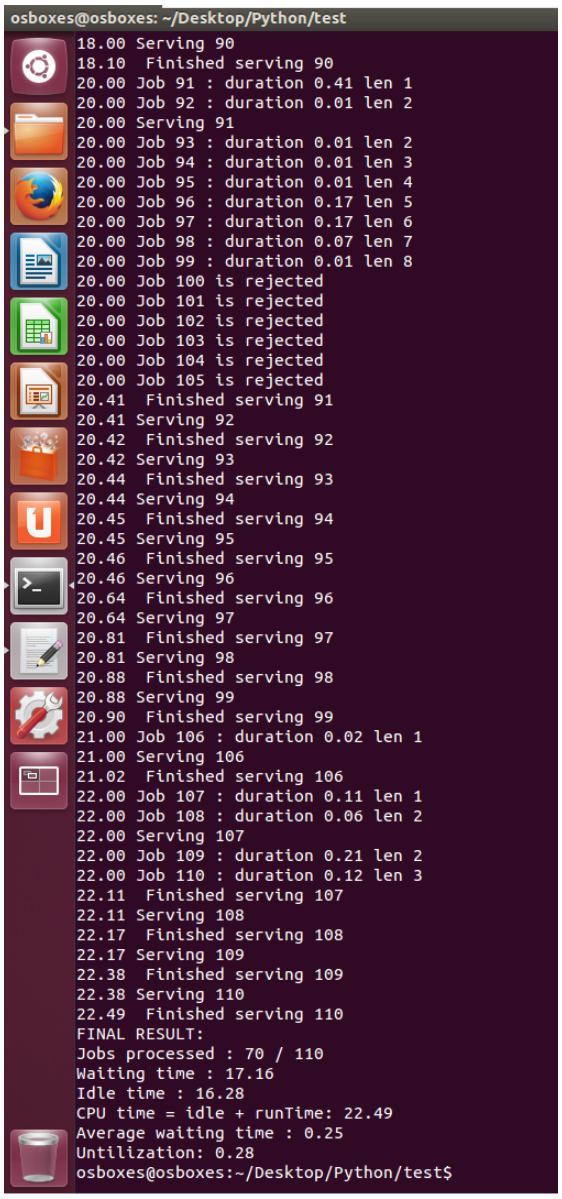
- Case:
  Service rate = 0.5, arrival rate = 0.1, number of buffers = 10, number of jobs = 200:



```
osboxes@osboxes: ~/Desktop/Python/test
16.00 Job 85 is rejected
16.00 Job 86 is rejected
16.00 Job 87 is rejected
16.00 Job 88 is rejected
16.00 Job 89 is rejected
16.00 Job 90 is rejected
16.00 Job 91 is rejected
16.00 Job 92 is rejected
16.00 Job 93 is rejected
16.00 Job 94 is rejected
16.00 Job 95 is rejected
16.00 Job 96 is rejected
16.00 Job 97 is rejected
16.00 Job 98 is rejected
16.00 Job 99 is rejected
16.00 Job 100 is rejected
16.00 Job 101 is rejected
16.00 Job 102 is rejected
16.03  Finished serving 50
16.03 Serving 51
16.23  Finished serving 51
16.23 Serving 52
16.30  Finished serving 52
16.30 Serving 53
16.39  Finished serving 53
16.39 Serving 54
16.47  Finished serving 54
16.47 Serving 55
16.60  Finished serving 55
17.00 Job 103 : duration 0.02 len 1
17.00 Serving 103
17.02  Finished serving 103
18.00 Job 104 : duration 0.07 len 1
18.00 Job 105 : duration 0.06 len 2
18.00 Serving 104
18.00 Job 106 : duration 0.16 len 2
18.00 Job 107 : duration 0.07 len 3
18.00 Job 108 : duration 0.05 len 4
18.00 Job 109 : duration 0.02 len 5
18.00 Job 110 is rejected
18.07  Finished serving 104
18.07 Serving 105
18.13  Finished serving 105
18.13 Serving 106
18.29  Finished serving 106
18.29 Serving 107
18.36  Finished serving 107
18.36 Serving 108
18.42  Finished serving 108
18.42 Serving 109
18.44  Finished serving 109
FINAL RESULT:
Jobs processed : 44 / 110
Waiting time : 8.93
Idle time : 13.83
CPU time = idle + runTime: 18.44
Average waiting time : 0.20
Untilization: 0.25
```

Figure 9: 110 jobs, 5 buffers, $\mu = \lambda = 0.1$

- Case:
  Service rate = 0.1, arrival rate = 0.5, number of buffers = 10, number of jobs = 100:



```
osboxes@osboxes: ~/Desktop/Python/test
18.00 Serving 90
18.10  Finished serving 90
20.00 Job 91 : duration 0.41 len 1
20.00 Job 92 : duration 0.01 len 2
20.00 Serving 91
20.00 Job 93 : duration 0.01 len 2
20.00 Job 94 : duration 0.01 len 3
20.00 Job 95 : duration 0.01 len 4
20.00 Job 96 : duration 0.17 len 5
20.00 Job 97 : duration 0.17 len 6
20.00 Job 98 : duration 0.07 len 7
20.00 Job 99 : duration 0.01 len 8
20.00 Job 100 is rejected
20.00 Job 101 is rejected
20.00 Job 102 is rejected
20.00 Job 103 is rejected
20.00 Job 104 is rejected
20.00 Job 105 is rejected
20.41  Finished serving 91
20.41 Serving 92
20.42  Finished serving 92
20.42 Serving 93
20.44  Finished serving 93
20.44 Serving 94
20.45  Finished serving 94
20.45 Serving 95
20.46  Finished serving 95
20.46 Serving 96
20.64  Finished serving 96
20.64 Serving 97
20.81  Finished serving 97
20.81 Serving 98
20.88  Finished serving 98
20.88 Serving 99
20.90  Finished serving 99
21.00 Job 106 : duration 0.02 len 1
21.00 Serving 106
21.02  Finished serving 106
22.00 Job 107 : duration 0.11 len 1
22.00 Job 108 : duration 0.06 len 2
22.00 Serving 107
22.00 Job 109 : duration 0.21 len 2
22.00 Job 110 : duration 0.12 len 3
22.11  Finished serving 107
22.11 Serving 108
22.17  Finished serving 108
22.17 Serving 109
22.38  Finished serving 109
22.38 Serving 110
22.49  Finished serving 110
FINAL RESULT:
Jobs processed : 70 / 110
Waiting time : 17.16
Idle time : 16.28
CPU time = idle + runTime: 22.49
Average waiting time : 0.25
Untilization: 0.28
osboxes@osboxes:~/Desktop/Python/test$
```

Figure 10: 110 jobs, 8 buffers, $\mu = \lambda = 0.1$

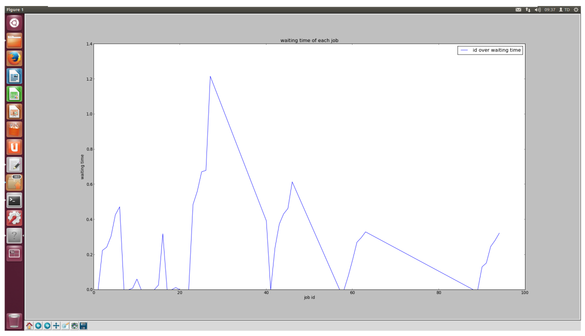
4 Conclusion

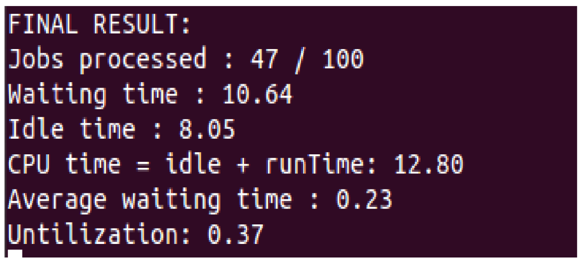
In conclusion, our group decided to analysis the result by One-Line graphic display to display the waiting time, response time and number of jobs in queue for clearly interaction with M/M/1/B = n queue. We applied **Terminating Simulation** to conduct the result.

After simulate the M/M/1/B=n queue, we have some consideration:

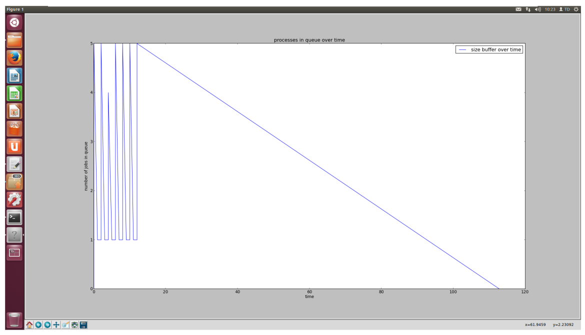
- Advantage: simple, easy to implement and easy to control, use less resource, high CPU Utilization because it's limit the number of buffers.
- Disadvantage:
  - High rate of rejected jobs.
  - Because using FCFS, high average waiting time than other algorithm.

The result considering to each finished jobs requirements with 4 graphs:





```
FINAL RESULT:
Jobs processed : 47 / 100
Waiting time : 10.64
Idle time : 8.05
CPU time = idle + runTime: 12.80
Average waiting time : 0.23
Untilization: 0.37
```



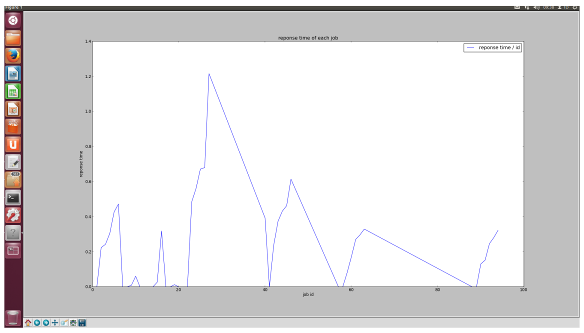
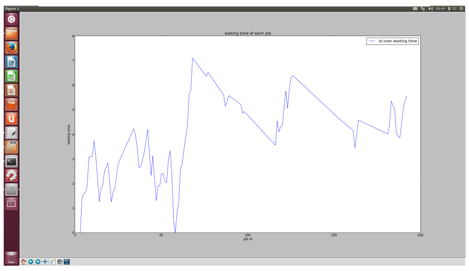
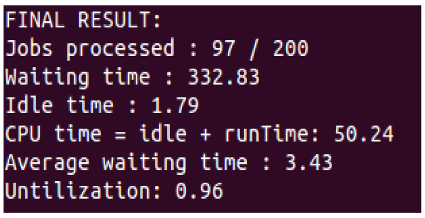
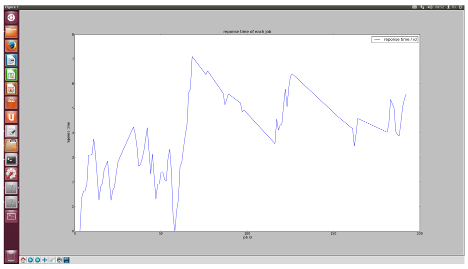


Figure 11: 100 jobs, 5 buffers, $\mu = \lambda = 0.1$





```
FINAL RESULT:
Jobs processed : 97 / 200
Waiting time : 332.83
Idle time : 1.79
CPU time = idle + runTime: 50.24
Average waiting time : 3.43
Untilization: 0.96
```



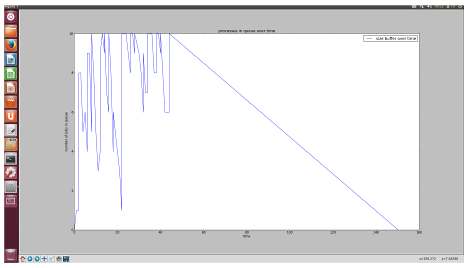
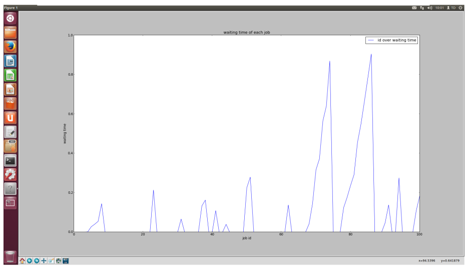
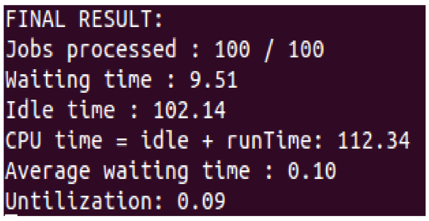
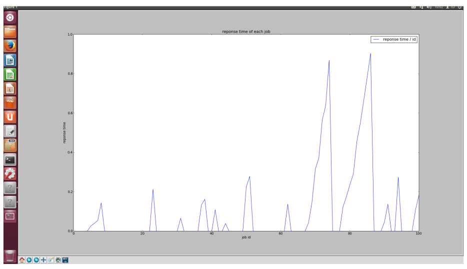


Figure 12: 200 jobs, 10 buffers, $\mu = 0.5$, $\lambda = 0.1$





```
FINAL RESULT:
Jobs processed : 100 / 100
Waiting time : 9.51
Idle time : 102.14
CPU time = idle + runTime: 112.34
Average waiting time : 0.10
Untilization: 0.09
```



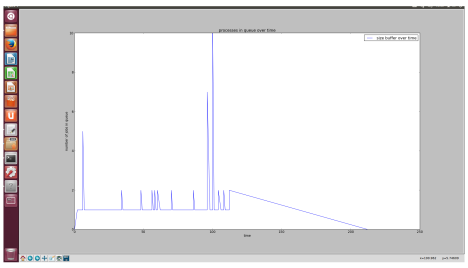


Figure 13: 100 jobs, 10 buffers, $\mu = 0.1$, $\lambda = 0.5$

Discussion we are measured the change of the system with different value of service rate and arrival rate:

- Lower arrival rate, higher service rate:

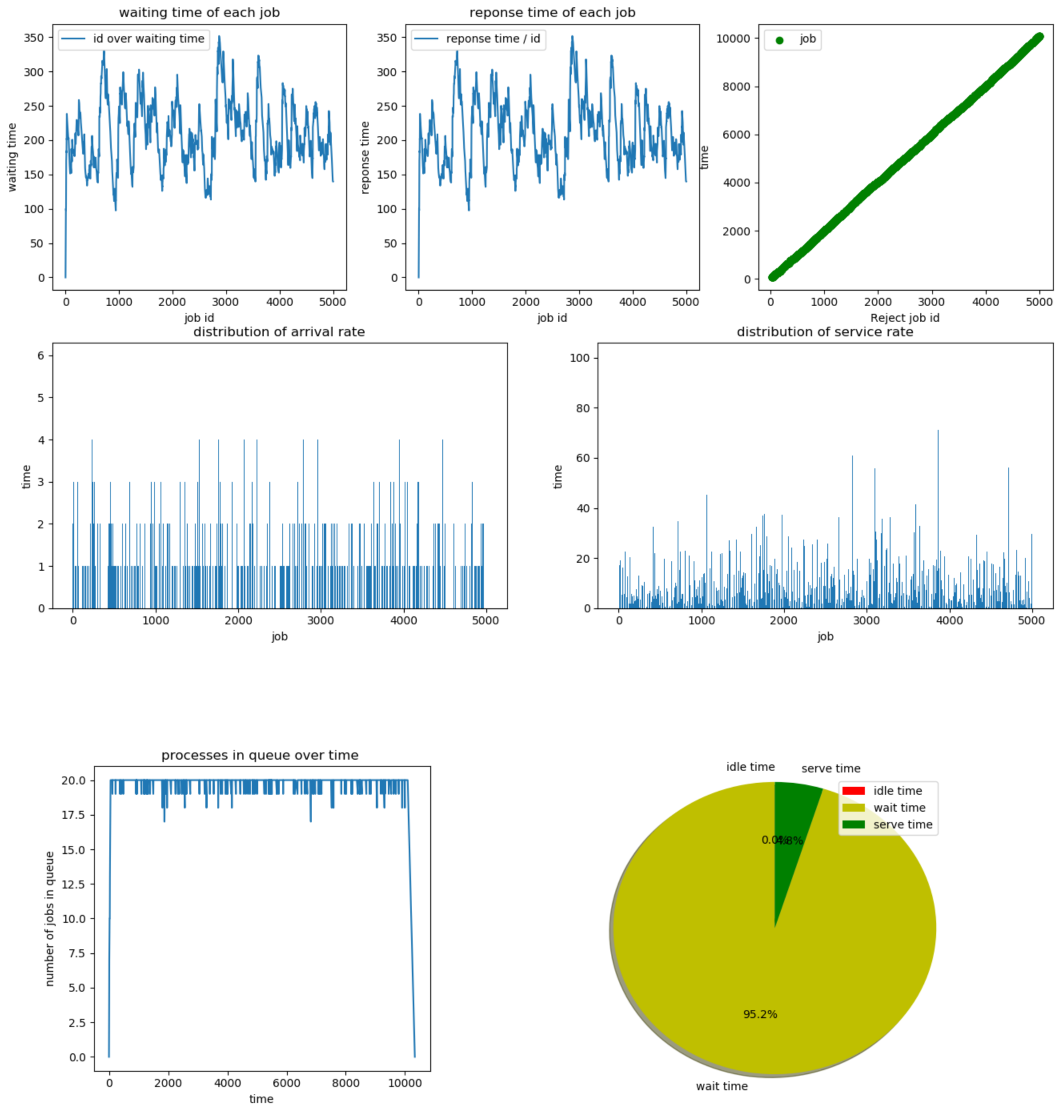


Figure 14: 5000 jobs, 20 buffers, arrival time = 1, service time = 10

- Higher arrival rate, lower service rate:

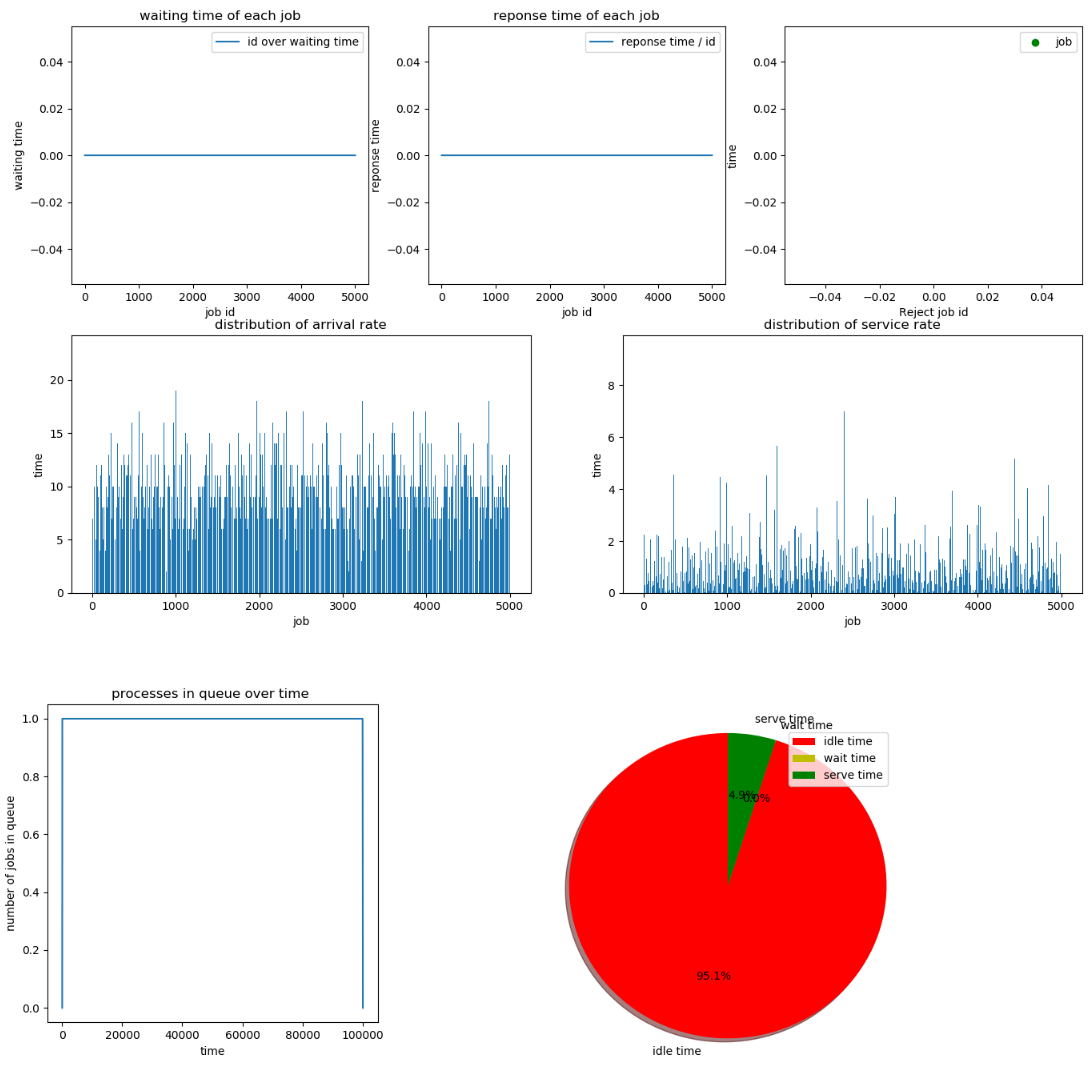


Figure 15: 5000 jobs, 20 buffers, arrival time = 10, service time = 1

- Higher arrival rate, lower service rate:

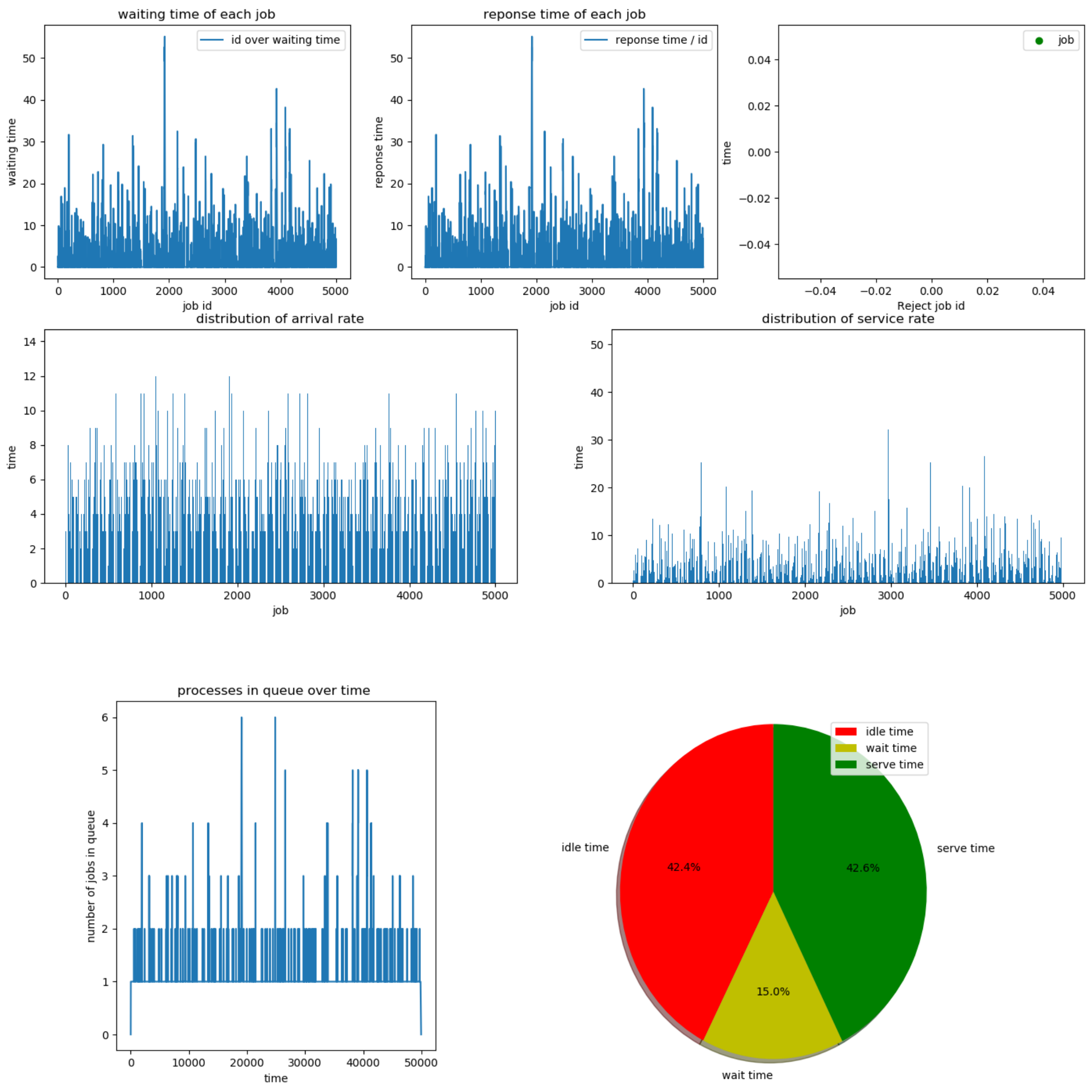


Figure 16: 5000 jobs, 20 buffers, arrival time = 5, service time = 5

5 Log File

Day	Work Content	Result
6/11/2019	- Have a conversation to choose topic of the assignment - Review some document on the internet to define the problems	- Initially, choose topic 2 to do - Find some document at Wiki and other pages as well as the lecture slide to understand what is M/M/n
9/11/2019	- Try to install Pycharm on Window OS and Python compiler - Works to find what is Simpy and how to install it	- Success to install Pycharm and compiler on Window OS - Get a overview of Simpy on Simpy's main page
12/11/2019	- Change OS: Using Ubutu instead of Window - Install Simpy and Python3 in Ubuntu 12.04 - Learning basic concept of Simpy	- Installed package simpy, numpy, python3,.. in Ubuntu OS - Understand about generator(), random(), interrupt() waiting(),... of simpy
13/11/2019	- Learning python using simpy - Read and try to understand examples that teacher given	- We got some understand about python coding structure but still be hard because we don't have any experience on simpy
15/11/2019	- Still working on learning Python with simpy - Ask teacher about the main purpose of the project	- It's better because we are in flow - Understand about class and other basic concept of Python - Know exactly what is the requirement of the topic
17/11/2019	- First time try to write a simulation on simpy	- Unfortunately, the project wasn't working well as we expect
20/11/2019	- We decided to change the project topic again.	- Finally, we choose topic 1 with M/M/1/B = n queue
22/11/2019	- Ask teacher about Validation - Discuss about how to do the project as well as complete some of the plus point	- Complete the format of the assignment Python project
24/11/2019	- Write Generation of events in simpy - Trai try to write M/M/1 queue first, write server and generator class - Calculate response time and waiting time of the system	- Complete M/M/1 with no limitation of number of buffers - We try about 20 times with different number of customers and arrival of each one. - Make sure that the calculation is right - Calculated response time and waiting time
25/11/2019	- Trai and Thu discuss and write M/M/1/B = n queue - We begin to write report	- Completely the simulation implementation on simpy base Python3 - Complete introduction and contents of the report

26/11/2019	- Thu collect data and writing theorem then sum up in the report - Trai help to write implementation part of the report with Thu	- Almost 60% of the report completed
27/11/2019	- Thu prepare the slide for the presentation - Continue to write the remain part of the report	- Nearly 90% of the report is done
28/11/2018	- Discuss together to get the final slides - Trai write a conclusion and validation of the report	- Complete report - Complete slides

References

[1] PhD Le Hong Trang, PhD Tran Van Hoai. Lecture slides. HCMUT, 11/2019.

[2] Simpy Overview. Day accessed: 24/11/2049. Received from: https://simpy.readthedocs.io/en/latest/index.html

[3] Program for FCFS CPU Scheduling, *GeeksforGeeks*, Set 1, N/A. Received from: https://www.geeksforgeeks.org/program-for-fcfs-cpu-scheduling-set-1/

[4] FCFS Scheduling, *javaTpoints*. Received from: https://www.javatpoint.com/os-fcfs-scheduling

[5] CPU Ultilization, *Techopedia*. Received from: https://www.techopedia.com/definition/28291/cpu-utilization

[6] Raj Jain, The Art of Computer Systems Performance Analysis, Wiley-Interscience, 1991. (www.cse.wustl.edu/ jain/books/perfbook.htm)