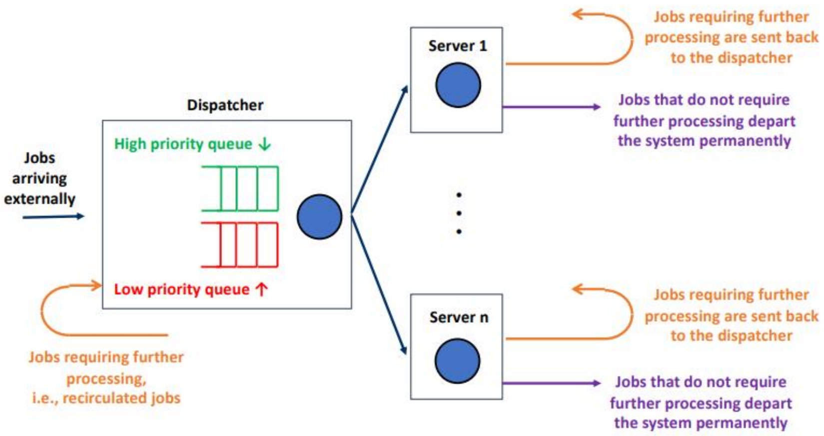
Z5211414
Jin-ao Olson Zhang
COMP9334 23T1 Project Report

Declaration for 6.6

All code is written by myself , except for the use of simulation, which refers to the logic of sim_mmm_lib.py in solution4b and also uses master_clock to monitor the actions of jobs.

There are two main file, main.py and jobclass.py, jobclass file contains all the classed that may use when doing the simulation,which according to the system for this project shows in Figure 1, they are
Jobs:
Dispatcher: hold servers and jobs
Server: hold jobs



1. Simulation program

1.1 test case correctness
For test the correctness of my code, I used given shell run_test.sh by
./runtest.sh $index_of_test_case
to get the simulation output,$index_of_test_case was a integer which helps us to find the corresponding reference that used to simulate such as paramater/simulation mode(in 6.1).

After that our code will generate two files dep_ $index.txt (which contains the completion times of of the server visits from the servers). and mrt_$index.txt (The mean response time)

to test the correctness we can run the project given correctness check file cf_output_with_ref.py to know our output was equal or not to the expected output,if so it will print

```
PS C:\Users\Olson\Desktop\comp9334\project_sample_files_25Mar> python3 cf_output_with_ref.py 3
Test 3: Mean response time matches the reference
Test 3: Completion times match the reference
```

1.2 Correctness of the inter-arrival probability distribution

inter-arrival probability distribution is exponentially distributed with parameter λ. This means the mean arrival rate of the jobs is λ ,in sim_mmm_lib.py it used random.expovariate to generate a series of pseudo-random numbers

```
# generate a new job and schedule its arrival
next_arrival_time = master_clock + random.expovariate(lamb)
service_time_next_arrival = random.expovariate(mu)
```

But our project recommend me to generate the inter-arrival times is to multiply an exponentially distributed random number with the given rate and a uniformly distributed random number in the given range,so my new arrival was generated by

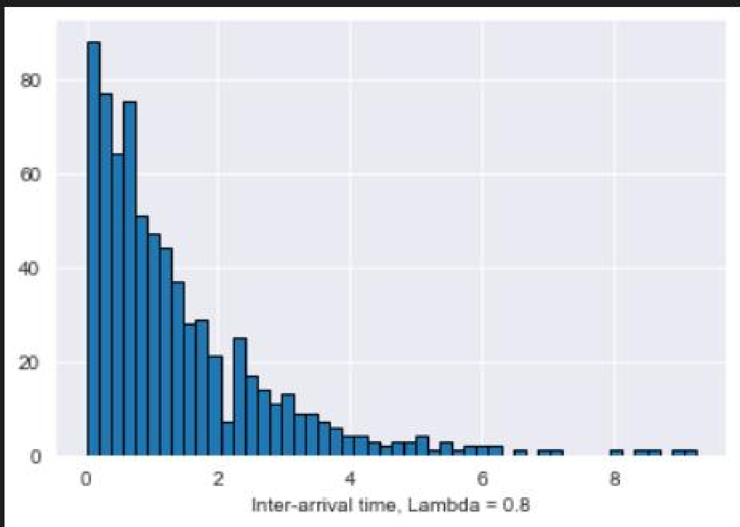
```
while cumulative_T < et:
    new_arrival = random.expovariate(lamda) * random.uniform(alpha2l,alpha2u)
```

lets using test_case 7 to show the result of inter-arrival probability distribution, we have

End Time	1000
Lamda	0.8
Mode	Random
Num of jobs	721

By using those argument as the input and plot the distribution of inter arrival time and number of jobs,we got

```
number of jobs is 721
```




```
mean arrival time is:1.3850733478900756
```

Where lamda here equals to 1.3850

	Expected value	Actual value
Inter arrival	1/lambd*(alpha2l+alpha2u/2)=1.25*1.09=1.3625	1.385

They are almost the same, thus, we can prove the inter-arrival probability distribution of my code are correct.

1.3 correctness of probability distribution of the number of server visits, and service time distribution

First know the server time in generate by g(t) followed this distribution

$$g(t) = \begin{cases} 0 & \text{for } 0 \le t \le \alpha \\ \frac{\gamma}{t^{\beta}} & \text{for } \alpha < t \end{cases} \qquad (1)$$

where

$$\gamma = \frac{\beta - 1}{\alpha^{1-\beta}}$$

So I printed g(t) as the Comparator of the correctness of my code.

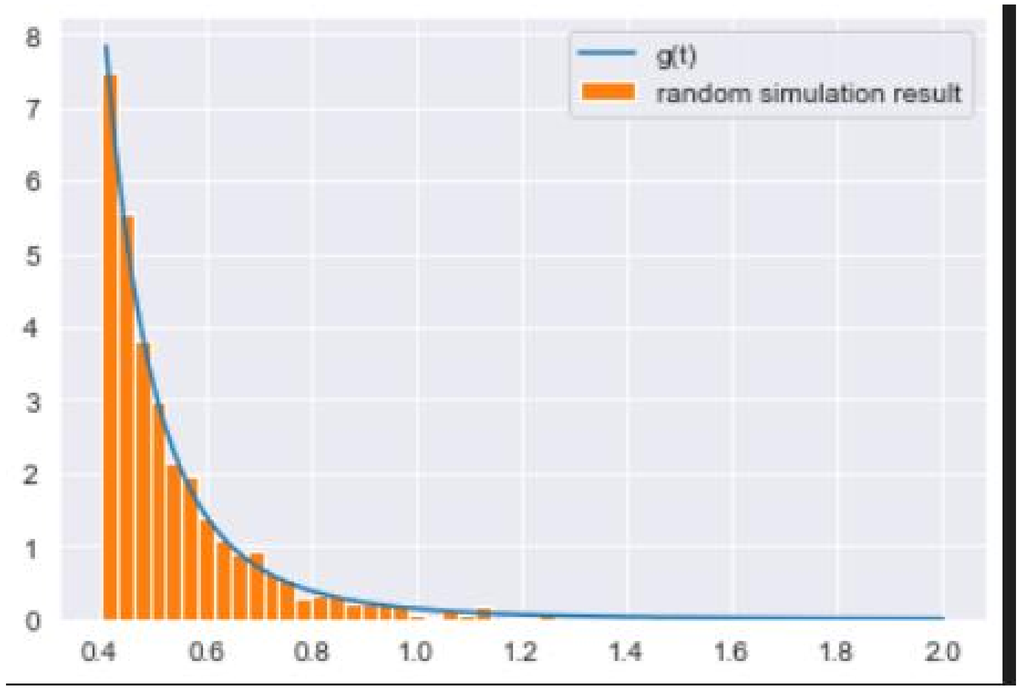
```
def g(t):
    gama = (beta - 1) / (alpha ** (1 - beta))
    return gama/(t**(beta))
x = np.linspace(alpha+0.01, 2, 100)
y = g(x)
plt.plot(x, y,label='g(t)')
```

Where x = np.linspace(alpha+0.01,2,100) generate a list of number from alpha+0.01 to 2 with same distance in 100steps.

In my code I got a get_server_time(alpha,beta) function in main.py to get random server time
Here I also used the reference of test_case_7
when alpha = 0.4,beta = 4.5,here I get chose to get 15000 values(No real requirement, but the more times the more accurate)

```
randomservisetime = []
for  i in range(1500):
    randomservisetime.append(get_server_time(alpha, beta))
```

By plotting both of g(t) and randomservisetime, I got



Its easy to find that they are almost the same, thus, we can prove the probability distribution of the number of server visits, and service time distribution of my code are correct.

1.3 simulation correctness

to check the correctness of my simulation,in "trace" mode I chose one of the examples in section 4 which is example 3

number of servers n = 3, threshold h = 1

Job index	Arrival time	Service times of the job's server visits
1	1.0	1.7, 2.8, 2.1
2	1.9	4.1, 4.9, 6.1
3	2.5	12.0
4	3.2	14.0

As I said befor,cf_output_with_ref.py can help me to check the ouput is or not equal to the expected output, for this example,the expected output was

Arrival time	Completion time	Number of completed server visits
1.0	2.7	1
1.0	5.5	2
1.9	6.0	1
1.0	8.1	3
1.9	13.0	2
2.5	14.5	1
1.9	19.1	3
3.2	19.5	1

Table 8 summarises the arrival and departure times of all the jobs. The mean response time of the 4 jobs in this example is $\frac{52.6}{4} = 13.15$.

And here is my code's output

dep_3.txt

文件 编辑 查看

```
1.0000 2.7000 1 3
1.0000 5.5000 2 3
1.9000 6.0000 1 3
1.0000 8.1000 3 3
1.9000 13.0000 2 3
2.5000 14.5000 1 1
1.9000 19.1000 3 3
3.2000 19.5000 1 1
```

mrt_3.txt

文件 编辑 查看

```
13.1500
```

Well for random mode ,I'm not sure how to verify the correctness of the code for now

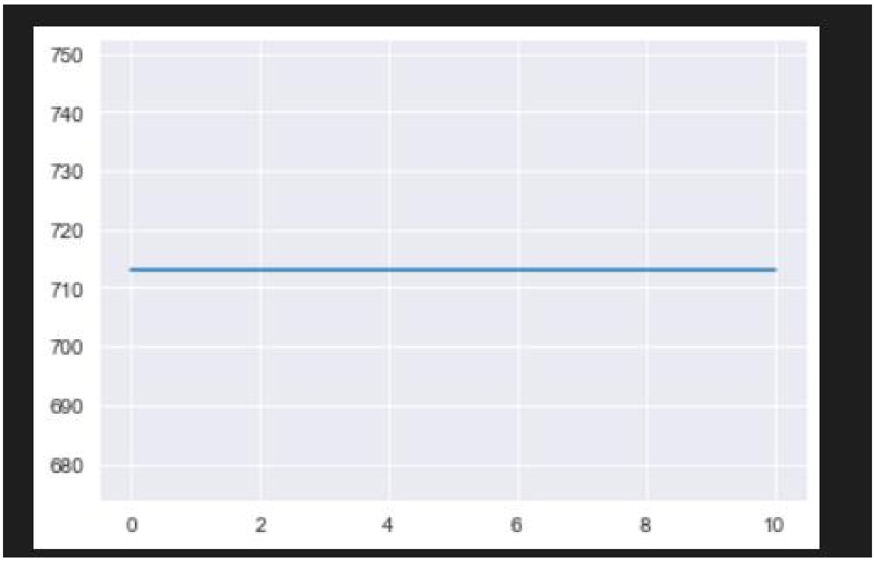
2.1 reproducible

In the code, I used random.expovariate() and random.uniform(), its seems that the value generated everytime cannot be same,but to make sure my code is reproducible, I added a random seed=1 into random function, which change it to pseudo-random,so even multiple time used(with the same input) ,the output will remain the same.

Here is a function that plot the changes of jobs in ten different random test with same input.（all the reference are from test case 7 which already shows before)

```
def tentimes(cumulative_T,seed):
    random.seed(seed)
    nb_of_jobs = 0
    while cumulative_T < time_end:
        random_arrival = random.expovariate(Lambda)*random.uniform(alpha2l,alpha2u)
        if random_arrival == 0:
            continue
        cumulative_T += random_arrival
        if cumulative_T > time_end:
            break
        inter_arrival_time_list.append(random_arrival)
        #arrival_time_list.append(cumulative_T)
        nb_of_jobs += 1
    return nb_of_jobs
list = []
cumulative_T = 0.0
for i in range(10):
    list.append(tentimes(cumulative_T,seed))
    cumulative_T = 0.0
x = np.linspace(0, 10, 10)
y = list
plt.plot(x, y,label='tentimes')
```

The plot show that it didnt change in 10 time test,so its reproducible



3.1 determining a suitable value of the threshold h

First of all,the 5.2 part of project description want me to assume that

- Number of servers: $n = 6$

16

- For inter-arrival times: $\lambda = \cancel{3.9}\ 3.1$, $a_{2\ell} = 0.91$, $a_{2u} = 1.27$
- For the number of server visits required for each job: the sequence p_1, p_2, p_3, p_4, p_5 is ~~0.52, 0.21, 0.15, 0.08, 0.04~~ 0.32, 0.21, 0.15, 0.08, 0.24.
- For the service time per server visit: $\beta = 3.4$, $\alpha = 0.3$.

On this basis, I think the best way to find the suitable threshold h is to use the control variables in the statistics， do not change any parameters other than h and observe the change in mean response time for different h

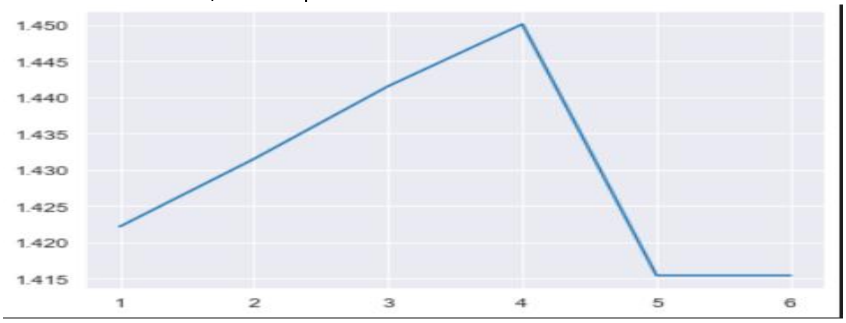
```
beta, alpha = 3.4, 0.3
n, end_time = 6, 1000.0
p = np.array([0.32, 0.21, 0.15, 0.08, 0.24])
Lambda, alpha2l, alpha2u = 3.1, 0.91, 1.27
```

For h from 1 to n， calculate the mrt of simulation

```
interArrival, service = randomsimulator(Lambda, alpha2l, alpha2u, p, end_time, alpha, beta)
for h in range(1, n+1):
    allJob = []
    nextArrival = 0.0
    for i, interval in enumerate(interArrival):
        nextArrival += interval
        idx = i + 1
        totalVisitTime = (~np.isnan(service[i])).sum()
        serviceTime = service[i]
        allJob.append(Job(index=idx, arrivalTime=nextArrival, serviceTime=serviceTime, totalVisitTime=totalVisitTime))

    dispatcher = Dispatcher(h, n)
    mrt.append(loadvalue(allJob,11,dispatcher))
print(mrt)
plt.plot([i for i in range(1,n+1)],mrt)
```

In random seed = 1, the output is



We can find when h=5 to h=6, mrt becomes to the minimum and remain the same.

END of Report