

Student Number: z5050331

Student Name: Jincheng Liu

Course: COMP 9334

Lab: Project

Simulation Goal:

Develop simulation models to decide the number of servers, which could minimise the mean response time of the request. Use some statistical methods to analyse the result.

Simulation Tools:

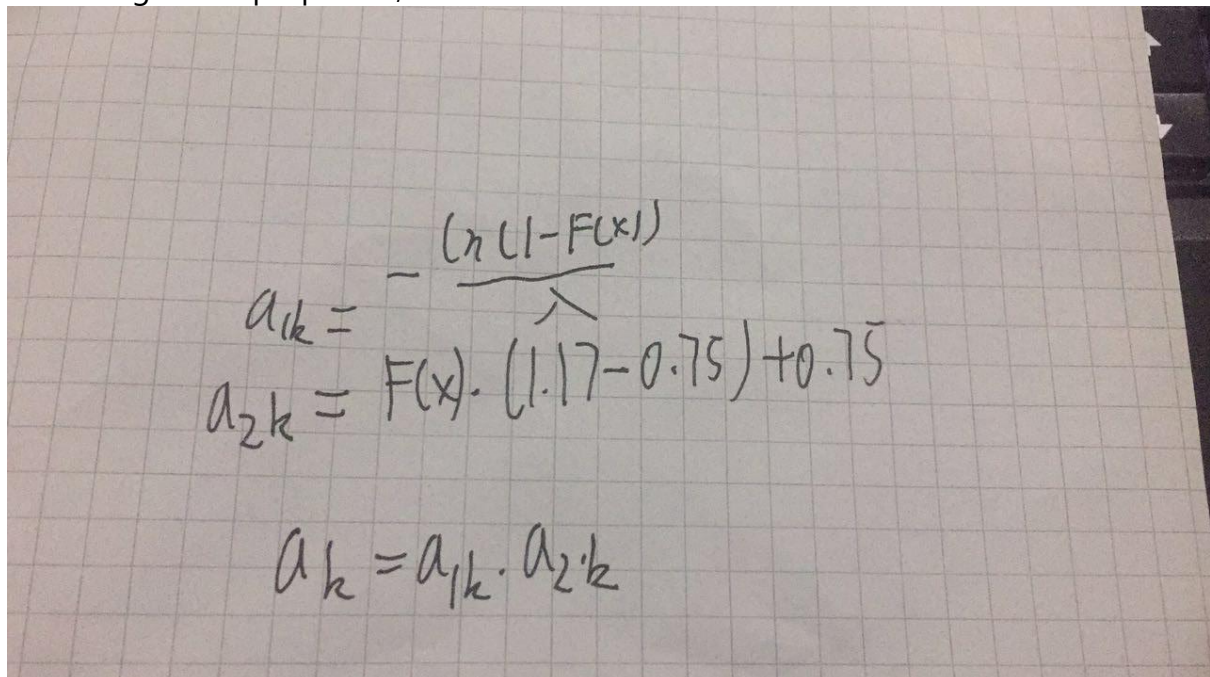
This simulation is developed by Java, the way how to run the code would be introduced at the last page. The analysis figures are created by JFreeChart, which is also written by Java.

The required probability distributions:

We use $\{a_1, a_2, \dots, a_k, \dots\}$ to denote the inter-arrival times of the requests arriving at the high-speed router. These inter-arrival times have the following properties:

- (a) Each a_k is the product of two random numbers a_{1k} and a_{2k} , i.e. $a_k = a_{1k}a_{2k} \forall k = 1, 2, \dots$
- (b) The sequence a_{1k} is exponentially distributed with a mean arrival rate 7.2 requests/s.
- (c) The sequence a_{2k} is uniformly distributed in the interval $[0.75, 1.17]$.

According to the properties, we can write the formula of the inter-arrival:


$$a_{1k} = -\frac{\ln(1-F(x))}{\lambda}$$
$$a_{2k} = F(x) \cdot (1.17 - 0.75) + 0.75$$
$$a_k = a_{1k} \cdot a_{2k}$$

, $F(x)$ is the random number (0~1) created by Java.

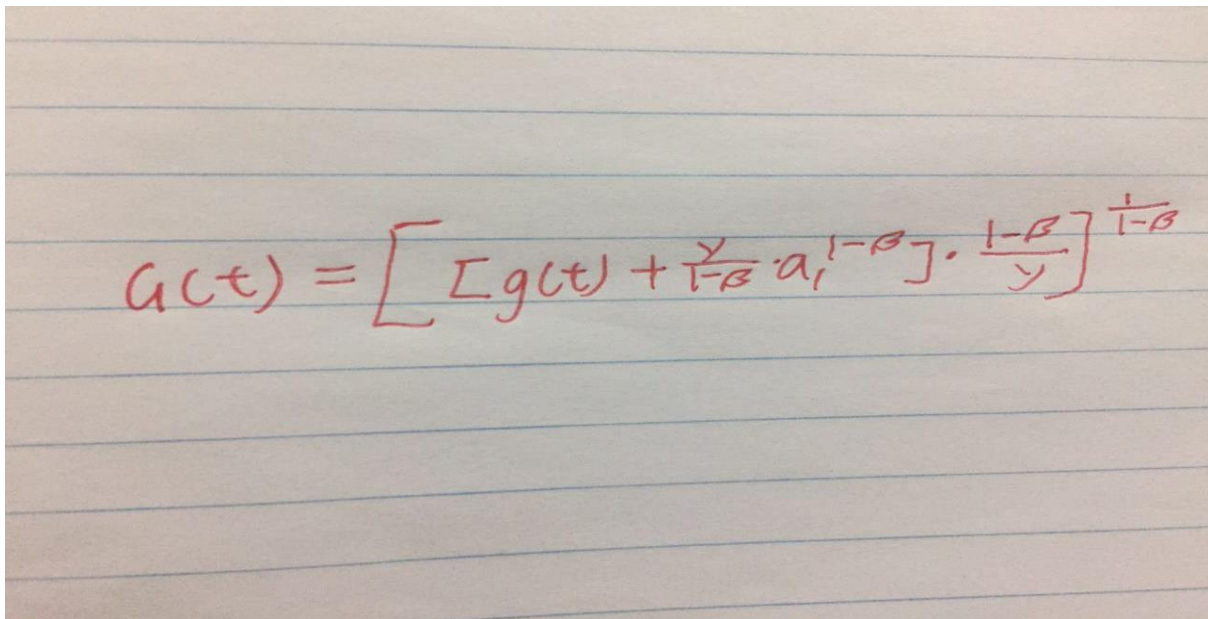
If the server is operating at 1 GHz, then the probability density function $g(t)$ of the service time t of the requests is:

$$g(t) = \begin{cases} 0 & \text{for } t \leq \alpha_1 \\ \frac{\gamma}{t^\beta} & \text{for } \alpha_1 \leq t \leq \alpha_2 \\ 0 & \text{for } t \geq \alpha_2 \end{cases} \quad (2)$$

where $\alpha_1 = \text{0.6 0.43}$, $\alpha_2 = \text{8 0.98}$, $\beta = 0.86$, and

$$\gamma = \frac{1 - \beta}{\alpha_2^{1-\beta} - \alpha_1^{1-\beta}}$$

According to the probability density function $g(t)$, we can write the formula of the service time:



$$t = \left[\int g(t) + \frac{\gamma}{1-\beta} \alpha_1^{1-\beta} \right] \cdot \frac{1-\beta}{\gamma}^{\frac{1}{1-\beta}}$$

, $g(t)$ is the random number (0~1) created by Java.

I have write the RandomNumberTest.java to test the numbers created by these functions. For example, in RandomNumberTest.java, it would create service time 5000000 times, which should be in the correct interval. If the number is in the wrong interval, it would print wrong message. But my result is "Service Time Function is correct", which shows my function is correct.

Simulation Reproducible:

To ensure my result is reproducible, I have the random seed to 1. When I re-stimulate the project, it would give me the same result.

```
this.SERVER_LIST = new ArrayList<Server>();  
this.random = new Random(1);  
}
```

Simulation Parameters:

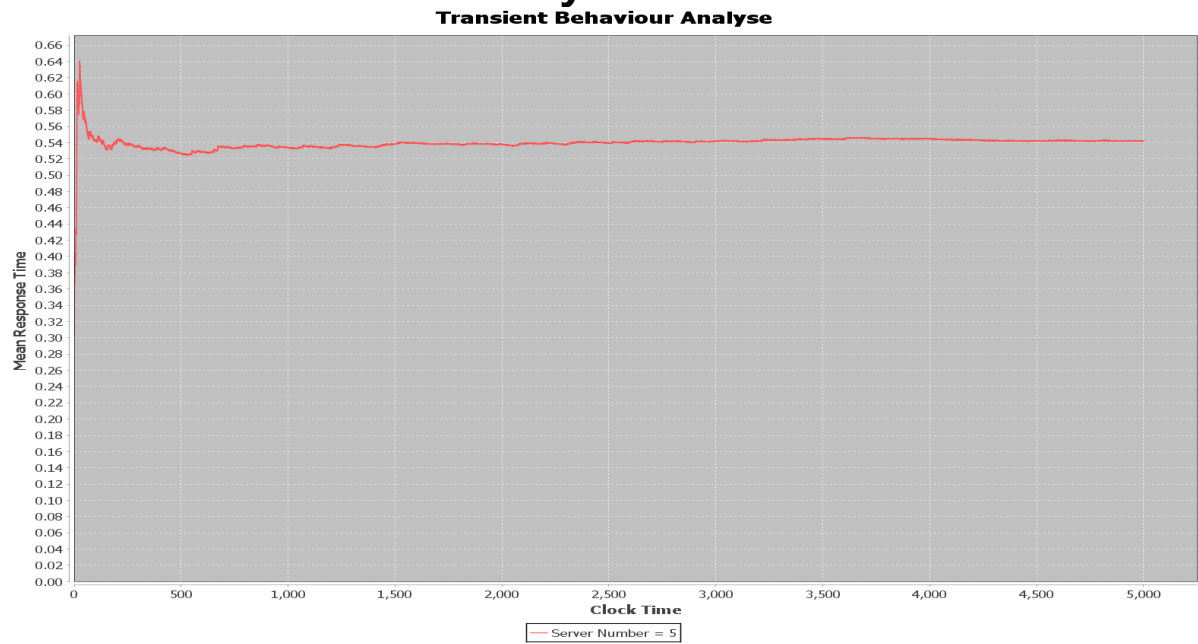
In this simulation, I have used the number of server as the parameter. The number of server is used to find under how many open server, the mean response time would have the minimum value. The stimulation result is the mean response time of the system.

```
for(int serverNumber = 3; serverNumber<= Simulation.SERVER_NUMBER;serverNumber++){  
    Simulation simulation = new Simulation(serverNumber);  
    Result result = simulation.start();  
}
```

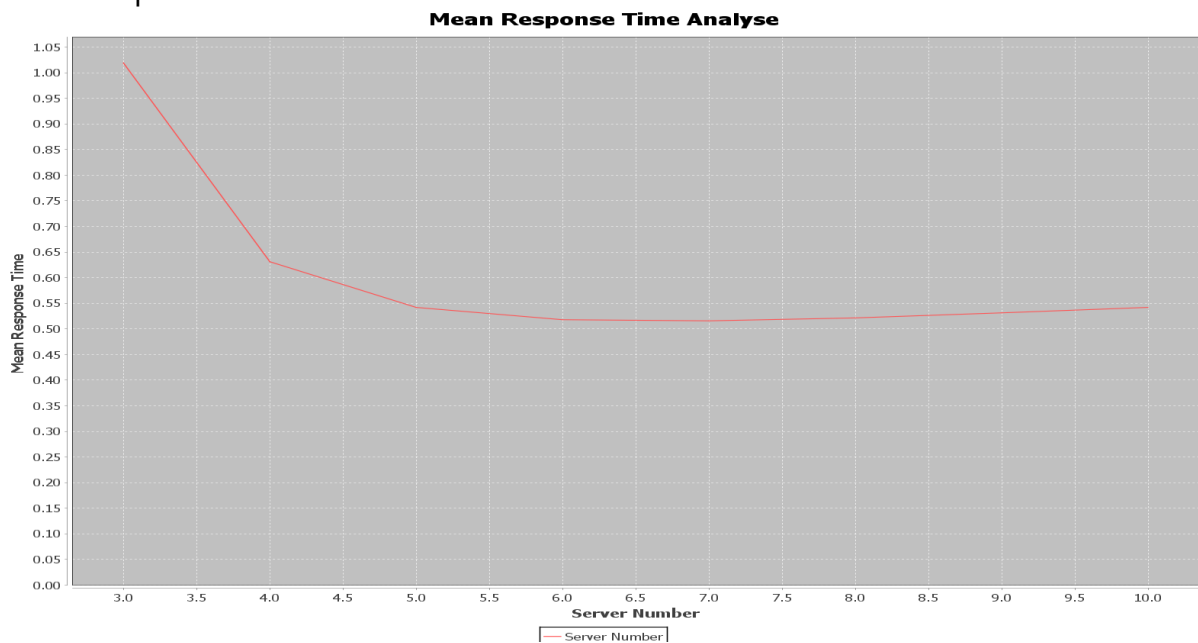
Simulation Way:

In this simulation, I have write the Server.java, which has the list of jobs. Each job has the arrival time and the remain service time of the job. In the Simulation.java, it has the number of server. In different simulations, it could use different number of servers. The server is PS server. When there are N jobs in the server, then each job receives 1/N of the service. So, in the simulation, when one event occurs, each job should update remain service times. There are two events in the system: arrival event and departure event. When one job departure, it's response time is equal to its arrival time minus its departure time. The average response time is equal to all response time divide the number of job.

Stimulation Result and Analyse:



From the 'Transient Behaviour Analyse'(TransientPartTest.java), we can know at the beginning, the mean response time is not in steady state. After 1100 jobs, the mean response time has been in the steady time, so I cut off the first 1100 jobs as the transient part.



AppData\Local\Program Files\Java\jdk-1.8.0_101\bin\java.exe [2017/6/16 14:14:11]

```
When 3 servers are on, the mean response time is 1.0191382172929297
The 95% confidence interval =[1.0191382172929295, 1.01913821729293].

When 4 servers are on, the mean response time is 0.6311568724895589
The 95% confidence interval =[0.6311568724895589, 0.6311568724895589].

When 5 servers are on, the mean response time is 0.5419257859441717
The 95% confidence interval =[0.5419257859441717, 0.5419257859441717].

When 6 servers are on, the mean response time is 0.5179507113320517
The 95% confidence interval =[0.5179507113320517, 0.5179507113320517].

When 7 servers are on, the mean response time is 0.5157481371613162
The 95% confidence interval =[0.5157481371613162, 0.5157481371613162].

When 8 servers are on, the mean response time is 0.5214603568907014
The 95% confidence interval =[0.5214603568907014, 0.5214603568907014].

When 9 servers are on, the mean response time is 0.5313419369923578
The 95% confidence interval =[0.5313419369923578, 0.5313419369923578].

When 10 servers are on, the mean response time is 0.5419855039879432
The 95% confidence interval =[0.5419855039879432, 0.5419855039879432].
```

From the result, we can know when 7 servers are on, the mean response time is the minimum value.

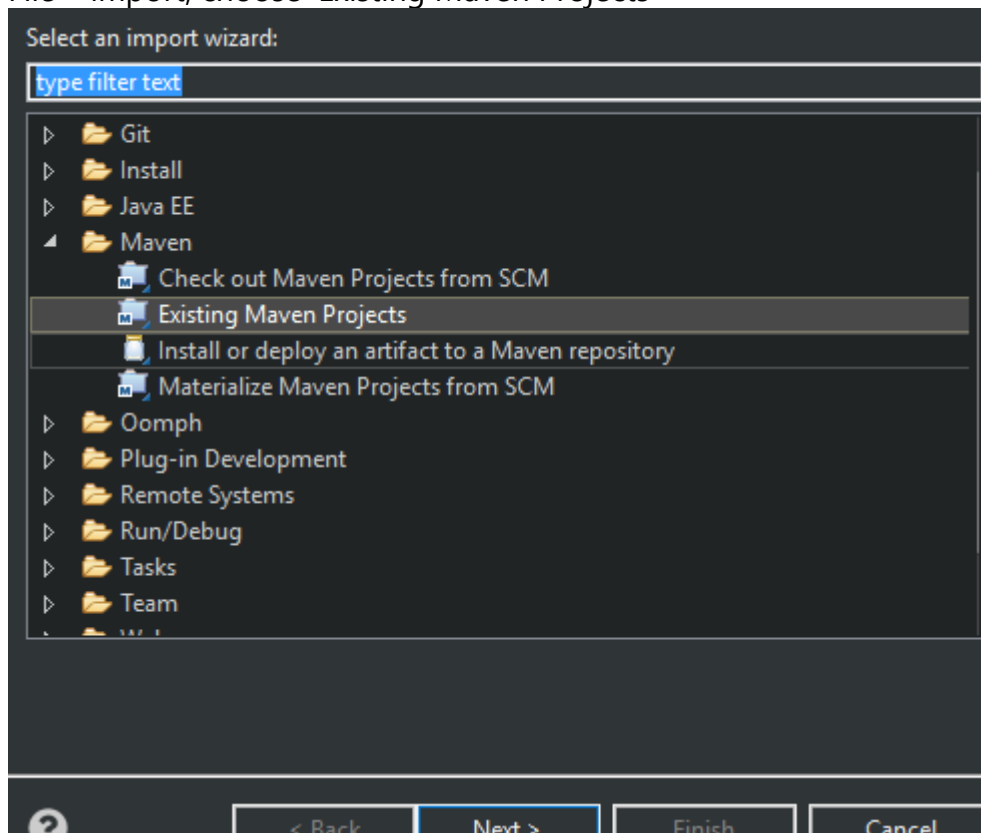
In our stimulation, the replication is 15. So according to the t-distribution table, when the p is 0.95, n is 15, the α is 1.753. And I can calculate the 95% confidence interval of each mean response time.

From the result, n=6 and n7, they almost have the same response time, but the 6th mean response time is larger than the 7th 's. And from the 95% confidence interval, we also can know the response time of 6th is larger than 7th 's.

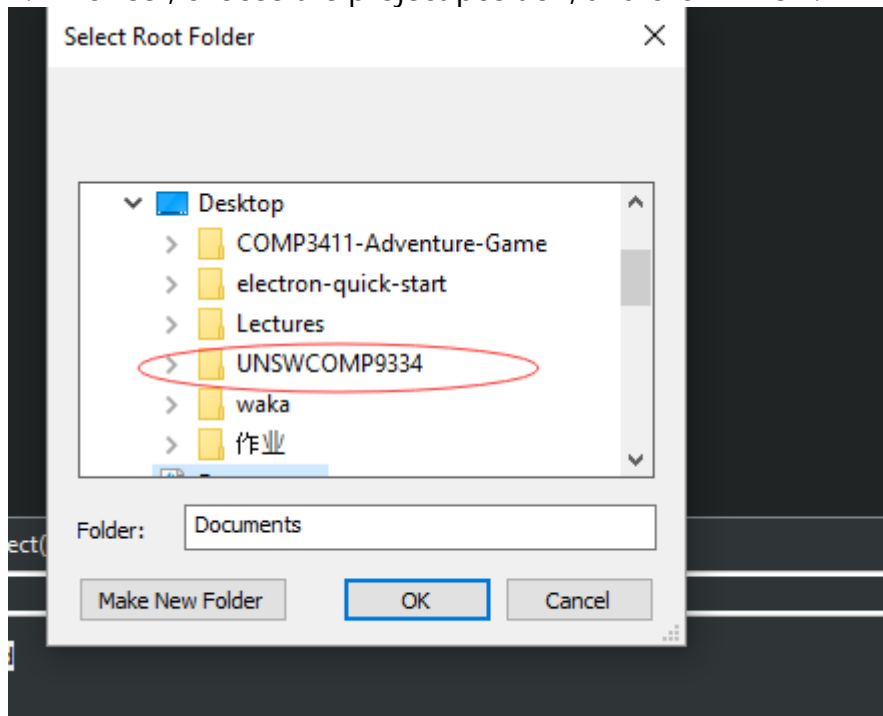
How to use my code:

My simulation is using maven to control the project, we can use the eclipse to import the project.

1. File->import, choose 'Existing Maven Projects'



2. Browser, choose the project position, and click 'Finish'.



3. If you want to see the Transient Part Figure, run the TransientPartTest.java

4. If you want to check the probability, run RandomNumberTest.java
5. If you want to see the Mean Response Analyse Figure, run the App.java