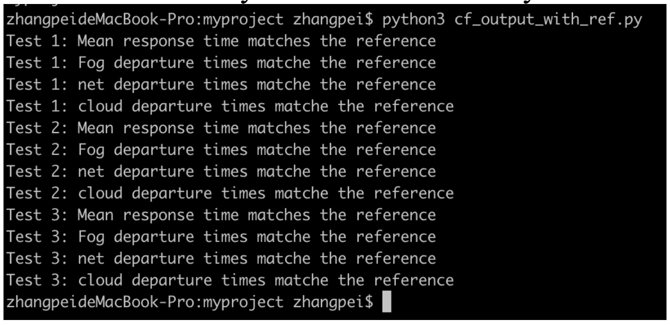
COMP9334 Project report
z5146286

1. Simulation Program (Running on Python)

1. Trace Mode.

I use three test cases to verify the correctness of my simulation program



```
zhangpeideMacBook-Pro:myproject zhangpei$ python3 cf_output_with_ref.py
Test 1: Mean response time matches the reference
Test 1: Fog departure times matche the reference
Test 1: net departure times matche the reference
Test 1: cloud departure times matche the reference
Test 2: Mean response time matches the reference
Test 2: Fog departure times matche the reference
Test 2: net departure times matche the reference
Test 2: cloud departure times matche the reference
Test 3: Mean response time matches the reference
Test 3: Fog departure times matche the reference
Test 3: net departure times matche the reference
Test 3: cloud departure times matche the reference
zhangpeideMacBook-Pro:myproject zhangpei$ 
```

All results of test case(mrt, fog_depart, net_depart,cloud_depart) match the reference txt(mrt_ref.txt, fog_depart_ref.txt, net_depar_ref.txt t,cloud_depart_ref.txt).

2. Random Mode.

The following chart are the parameters for this test case:

Arrival rate(λ)		5.720
Service time	α1	0.050
	α2	0.300
	β	0.740
FogtimeLimit		0.2
FogTimeToCloudeTime		0.6
Network latency	ν1	1.200
	ν2	1.470
Time end		1000
seed		1

- The inter-arrival probability distribution is exponentially distributed with parameter. For each job, its arrival time is the last arrival + inter-arrival time. Thus I use numpy.random.exponential() with parameter to generate the inter-arrival time.
- The service time in the fog time unit t is generated by the probability density function g(t):

$$g(t) = \begin{cases} 0 & \text{for } t \le \alpha_1 \\ \frac{\gamma}{t^\beta} & \text{for } \alpha_1 \le t \le \alpha_2 \\ 0 & \text{for } t \ge \alpha_2 \end{cases}$$

Where

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

Calculate service time:

$$g(t) = \frac{\gamma}{t^\beta} \quad \alpha_1 \le t \le \alpha_2$$

let $G'(t) = g(t)$

since $g(t) = r t^{-\beta}$

$$\Rightarrow G(t) = \frac{r}{1-\beta} t^{1-\beta} + C$$

$$G(\alpha_1) = \frac{r}{1-\beta} \alpha_1^{1-\beta} + c \, / \, G(\alpha_2) = \frac{r}{1-\beta} \alpha_2^{1-\beta} + C$$

$$\Rightarrow G(\alpha_2) - G(\alpha_1) = 1$$

$$\Rightarrow \text{random} = \begin{cases} G(\alpha_2) - G(\alpha_1) & t \ge \alpha_2 \\ G(t) - G(\alpha_1) & \alpha_1 < t < \alpha_2 \\ G(\alpha_1) - G(\alpha_1) & t \le \alpha_1 \end{cases}$$

$$\Rightarrow \text{random} = \begin{cases} 1 & t \ge \alpha_2 \\ G(t) - G(\alpha_1) & \alpha_1 < t < \alpha_2 \\ 0 & t < \alpha_1 \end{cases}$$

where

$$G(t) - G(\alpha_1) = \frac{t^{1-\beta}}{\alpha_2^{1-\beta} - \alpha_1^{1-\beta}} - \frac{\alpha_1^{1-\beta}}{\alpha_2^{1-\beta} - \alpha_1^{1-\beta}}$$

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

since $\text{random} \in (0, 1)$

$$\text{random} = \frac{t^{1-\beta} - \alpha_1^{1-\beta}}{\alpha_2^{1-\beta} - \alpha_2^{1-\beta}}$$

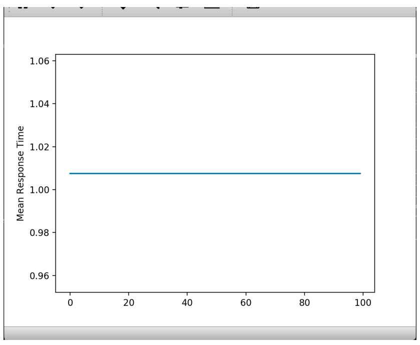
$$t^{1-\beta} = \text{random} * (\alpha_2^{1-\beta} - \alpha_1^{1-\beta}) + \alpha_1^{1-\beta}$$

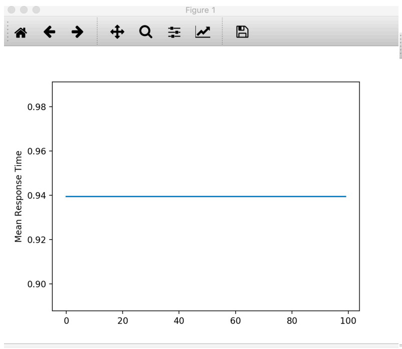
$$t = \sqrt[1-\beta]{\text{random} * (\alpha_2^{1-\beta} - \alpha_1^{1-\beta}) + \alpha_1^{1-\beta}}$$

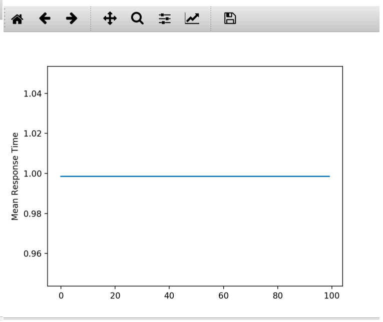
- The network latency is uniformly distributed in the open interval (v1, v2) where v2 > v1 > 0. I use the numpy.random.uniform() to generate latency.

2. Reproducibility

I use seed(0), seed(1), seed(2) to run simulation 100 times respectively. And each of seed have same result. The output of three seed store in documents.

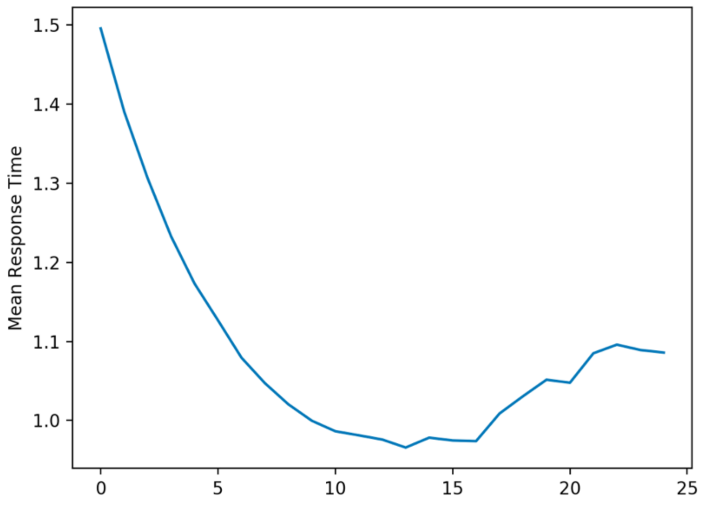






3. Determine FogTimeLimit

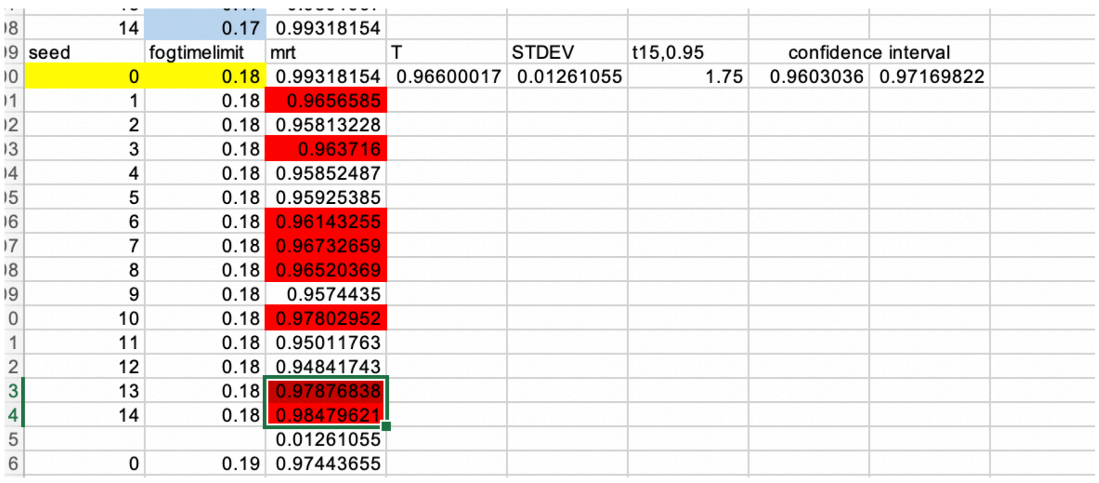
At first, I run the simulations to try different fogTimeLimit from 0.05 to 0.3 with same seed. The length of each simulation is 10000 units of time, which is long enough for transient removal.
The following graph show the result: (x-axis * 0.01 + 0.05 = fogTimeLimit)



We can easily see, between 10 an 15, the value of MRT is in deep .

Then I use different seed to run the simulations for 15 times, which is also the number of replications. In each experiment, program will try different value of fogTimeLimit from 0.050 to .0300. The length of each simulation is 10000 units of time, which is long enough for transient removal. The result store in data.xls.

After comparing the MRT in ‘ data.xls ‘with same seed and different fogTimelimit, the value of MRT almost decrease until fogTimelimit = 0.18.



8	14	0.17	0.99318154					
9	seed	fogtimelimit	mrt	T	STDEV	t15,0.95	confidence interval	
0	0	0.18	0.99318154	0.96600017	0.01261055	1.75	0.9603036	0.97169822
1	1	0.18	0.9656585					
2	2	0.18	0.95813228					
3	3	0.18	0.963716					
4	4	0.18	0.95852487					
5	5	0.18	0.95925385					
6	6	0.18	0.96143255					
7	7	0.18	0.96732659					
8	8	0.18	0.96520369					
9	9	0.18	0.9574435					
0	10	0.18	0.97802952					
1	11	0.18	0.95011763					
2	12	0.18	0.94841743					
3	13	0.18	0.97876838					
4	14	0.18	0.98479621					
5			0.01261055					
6	0	0.19	0.97443655					

Accounting computing confidence interval ,I prefer to let fogtimelimit be 0.18.