COMP9334 PROJECT

z5141180 Yizheng Ying

1. The reproducible result is achieved by seed = 135

The reproducible result is achieved by seed = 135 to create seed_list which include 5 random numbers by $\text{random.sample(range(100,5000),5)}$.

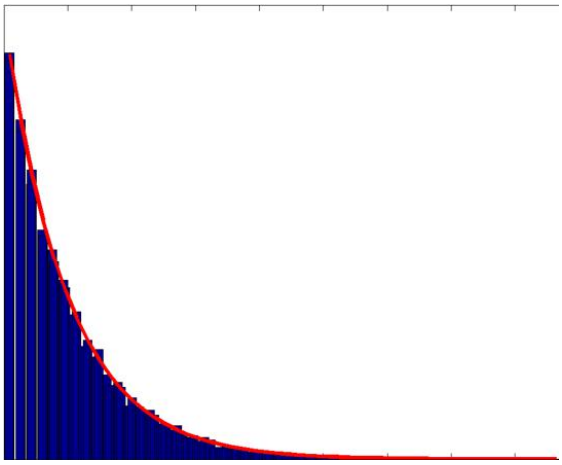
All random number generators for replications are attached in the submission and can be reproduce by using the seeds in the seed_list.

2. The function parameter explanation

The main simulation function is simulation2(mode, arrival, service, network, fogTimeLimit, fogTimeToCloudTime, time_end, num,seed)

mode : random or trace mode

arrival : in trace mode, the list of arrival times. in random mode, it contains a string for a floating point number. This number corresponds to the value of λ. It is use to create inter arrival probability distribution which is exponentially distributed.



We can get the uniformly distributed in(0,1) and use -log(1-u)/lamda to get the result.

service : in trace mode, it is the list of service time in the fog time. in random mode, it contains three numbers, corresponding to the values of the $\alpha_1, \alpha_2, \beta.$

network : in trace mode, it is the list of network latency. in random mode it contains two numbers, corresponding to the values of the ν_1 and ν_2.

fogTimeLimit :the service time in the fog time unit is no more than the fogTimeLimit.

fogTimeToCloudTime : used to calculate the time of each job in the cloud.

time_end : it is only used in random mode, which stops the simulation if the master clock exceeds this value.

num : represent the number of the test.

seed : the random seed.

3. The required probability distribution of inter-arrival time and service time

The inter-arrival probability distribution is exponentially distributed with parameter λ. In the code, we use

next_arrival_time = round(random.expovariate(lamda),6)

to get the arrival interval.

For the service time, we have the probability density function g(t) where:

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

And

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

So we can get:

$$G(t) = \begin{cases} 0 & for\ t \le \alpha_1 \\ \frac{\gamma}{1-\beta}(t^{1-\beta} - \alpha_1^{1-\beta}) & for\ \alpha_1 \le t \le \alpha_2 \\ 1 & for\ t \ge \alpha_2 \end{cases}$$

The inverse function of G(t) is :

$$invG(\mu) = [\frac{1-\beta}{\gamma}\left(\mu + \frac{\gamma}{1-\beta}\alpha_1^{1-\beta}\right)]^{\frac{1}{1-\beta}} \quad where\ \mu \in [0,1]$$

μ is uniformly distributed between [0, 1]. So we can use μ to get a distribution based on the invG(μ).

4. Implementation of simulation function:

1) Firstly we need to judge the mode is trace mode or random mode. In random mode, generate arrival, service distribution at fog, and network delay distributions. In trace mode we get arrival time list, service time list and get task_list and cloud_list where cloud_service_time is calculated by fogTimeToCloudTime * (service_time - fogTimeLimit), and latency_list, there are two attributes in each tuple, i.e.(arrive_time,network_latency_time). And we set global variables :

next_arrival_time = first element in arrival time list

next_departure_from_fog_without_cloud = 99999999

next_departure_from_fog_with_cloud = 99999999

next_departure_from_network = 99999999

next_departure_from_cloud = 99999999

99999999 means infinity. And set mast_clock=0, totalResponseTime=0, totalNumberOfJobsCompleted = 0

2) while next_arrival_time or next_departure_fog_time or next_network_cloud_time or next_cloud_time not equal to 99999999 or min(next_arrival_time, next_departure_fog_time, next_network_cloud_time, next_cloud_time) < time_end:

a. We set mast_clock_previous = mast_clock and then we set t equal to the minimum element in (next_arrival_time,next_departure_from_fog_without_cloud,next_departure_from_fog_with_clou d, next_departure_from_network, next_departure_from_cloud).

If the mast_clock = next_arrival_time: we set next_arrival_time = first element in arrival_list and then delete the element from the arrival_list.Update job list at fog, job list at network, job list of cloud.

b. If mast_clock = next_departure_from_fog_without_cloud: we update

next_departure_from_fog_without_cloud . Update job list at fog, job list at network, job list of cloud. Remove the job with 0 remaining service time. Update totalResponseTime =totalResponseTime + master_clock - arrival_time_at_fog. And update totalNumberOfJobsCompleted if the remaining service time in fog = 0. Add the job to fog_departure array.

c. If mast_clock = next_departure_from_fog_with_cloud: we update next_departure_from_fog_with_cloud. Update job list at fog, job list at network, job list of cloud. Add the job to network_departure array.

d. If mast_clock = next_departure_from_network: Update next_departure_from_network, job list at fog, job list at network, job list of cloud.

e. If mast_clock = next_departure_from_cloud: Update next_departure_from_cloud job list at fog, job list at network, job list of cloud. Remove the job with 0 remaining service time. Update totalResponseTime =totalResponseTime + master_clock - arrival_time_at_fog and totalNumberOfJobsCompleted. Add the job to cloud_departure array.

5. Justification of correct implementation

In order to prove that such implementation of my code produces correct results, I use the example in the specification of the project. Since we use the example 2 and example 3 with limited length of input vectors of arrival time and serive time, we have to limit the observation time to a fix interval to avoid the situation such that there are no more events and master clock can not advance, leading to infinite loop. In this case, we can get the results:

Time	Next_arrival_time	Next_fog_dep_time	Next_net_dep_time	Next_clo_dep_time
0	1.1	∞	∞	∞
1.1	6.2	3.1	∞	∞
3.1	6.2	∞	4.6	∞
4.6	6.2	∞	∞	5.86
5.86	6.2	∞	∞	∞
6.2	7.4	8.2	∞	∞
7.4	8.3	9	∞	∞
8.3	9.1	9.35	∞	∞
9.1	10.1	9.433	∞	∞
9.433	10.1	10.933	10.733	∞
10.1	∞	11.2111	10.733	∞
10.733	∞	11.2111	∞	12.6533
11.2111	∞	14.6611	∞	12.6533
12.6533	∞	14.6611	∞	∞
14.6611	∞	15.1944	∞	∞
15.1944	∞	15.0003	16.7944	∞
15.0003	∞	∞	16.7944	∞
16.7944	∞	∞	17.3	17.5144
17.3	∞	∞	∞	17.7288
17.7288	∞	∞	∞	18.7744
18.7744	∞	∞	∞	∞

mean_response_time =

$$\frac{(5.86-1.1)+(12.6533-6.2)+(11.2111-7.4)+(14.6611-8.3)+(17.7289-9.1)+(18.7744-10.1)}{6}$$

$$\approx 6.4481$$

This proves the system works correctly for the example 2 mentioned in the project.

In order to double check the system works correctly, we check the example 3.

Time	Next_arrival_time	Next_fog_dep_time	Next_net_dep_time	Next_clo_dep_time
0	1	∞	∞	∞
1	2	3.5	∞	∞
2	4	5	∞	∞
4	5	5.5	∞	∞
5	6	5.6667	∞	∞
5.6667	6	8.0667	7.1667	∞
6	∞	8.7996	7.1667	∞
7.1667	∞	8.7556	∞	8.0067
8.0067	∞	8.7556	∞	∞
8.7556	∞	9.3556	∞	∞
9.3556	∞	11.8222	10.7556	∞
10.7556	∞	11.8222	∞	12.5756
11.8222	∞	12.2	∞	12.5756
12.2	∞	∞	13.8	12.5756
12.5756	∞	∞	13.8	∞
13.8	∞	∞	∞	15.2
15.2	∞	∞	∞	∞

Mean_response_time=

$$\frac{(8.0067-1)+(12.5756-2)+(8.7556-4)+(11.8222-5)+(12.2-6)}{5}$$

$$\approx 7.6720$$

This simulation result is in accordance with the theoretical result, which proves that the codes can generate correct result in both example.

In conclusion, the fact that my codes are correctly implemented is verified by the two examples above.

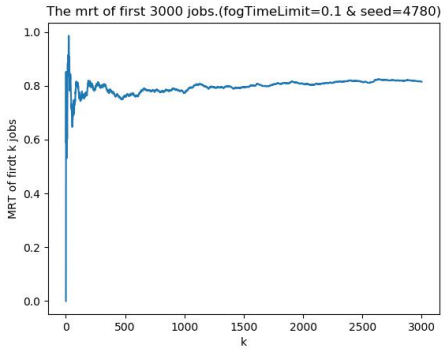
6. Observation Time and Transient State Selection

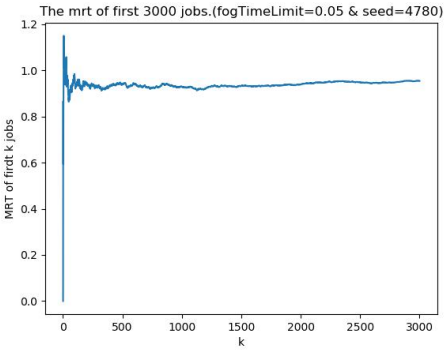
I use Welch method described in the lecture material to decided the transient state.

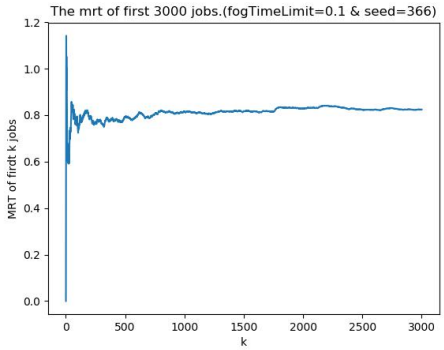
For observation time, I calculate the mean response time of ten replications to decide the observation time.

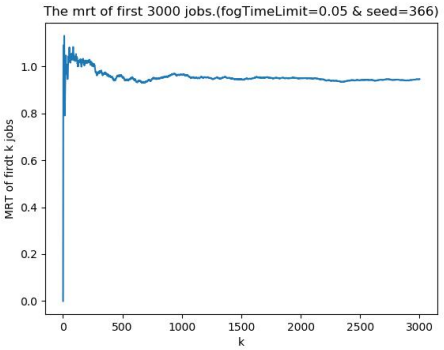
$$\lambda = 9.72, \alpha_1 = 0.01, \alpha_2 = 0.4, \beta = 0.86, \upsilon_1 = 1.2, \upsilon_2 = 1.47, time\_end = 1000$$

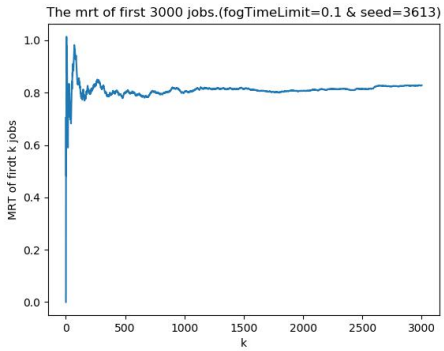
and fogTimeCloudTime is 0.6.

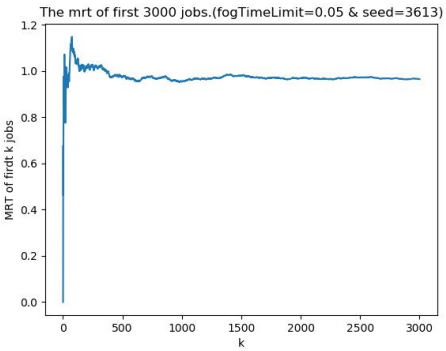

The mrt of first 3000 jobs.(fogTimeLimit=0.1 & seed=4780)
MRT of firdt k jobs
k

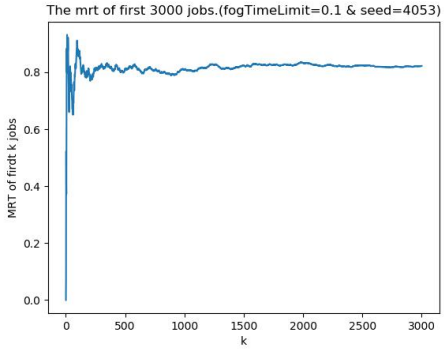

The mrt of first 3000 jobs.(fogTimeLimit=0.05 & seed=4780)
MRT of firdt k jobs
k

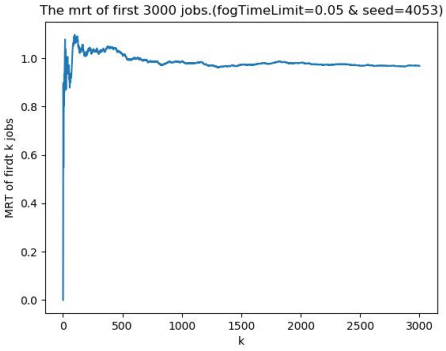

The mrt of first 3000 jobs.(fogTimeLimit=0.1 & seed=366)
MRT of firdt k jobs
k


The mrt of first 3000 jobs.(fogTimeLimit=0.05 & seed=366)
MRT of firdt k jobs
k


The mrt of first 3000 jobs.(fogTimeLimit=0.1 & seed=3613)
MRT of firdt k jobs
k


The mrt of first 3000 jobs.(fogTimeLimit=0.05 & seed=3613)
MRT of firdt k jobs
k


The mrt of first 3000 jobs.(fogTimeLimit=0.1 & seed=4053)
MRT of firdt k jobs
k


The mrt of first 3000 jobs.(fogTimeLimit=0.05 & seed=4053)
MRT of firdt k jobs
k

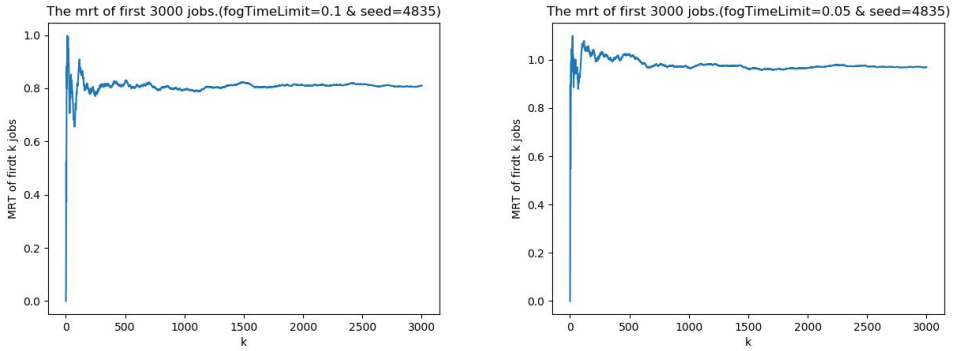


Figure. The mean response time of 5 replications

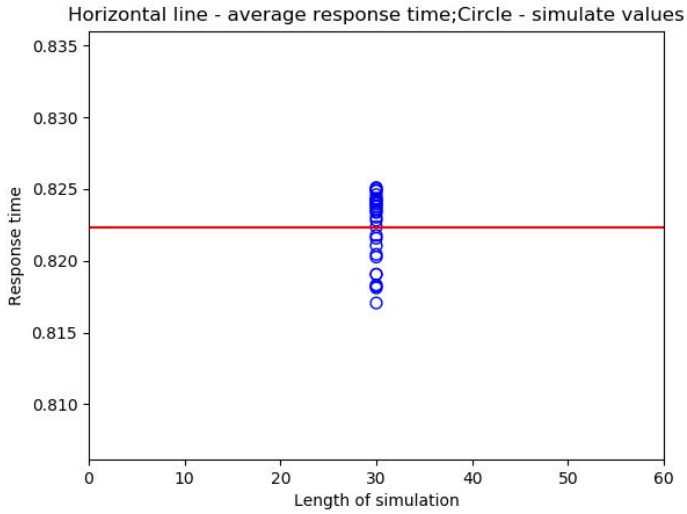
(X-axis: first k jobs Y-axis: Mean Response Time of first k jobs)

The 5 figure on the left are created by fogLimitTime = 0.1 and the 5 figure on the right hand are created by fogLimitTime = 0.05. Comparing the figures above we can see that after the first 1500 jobs, the curve will become converged. It means that the time from 0 to 1500 is transient state.

In conclusion, the transient state is 0 to 1500 jobs and the observation time is set to 3000 jobs.

7. Replication Number Selection

I set replications as 30 and can get the response time like below:



I first set the number of replications to 5 replications, we can get:

The numbers of mean and the confidence intervals of different fogLimitTime are shown in the table below. The confidence interval is 95% confidence interval with alpha = 0.05. We use the formula in the lecture to calculate the confidence interval.

$$[\hat{T} - t_{n-1,1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}, \hat{T} + t_{n-1,1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}]$$

Table 1 Mean response time and confidence interval with 5 replications

fogLimitTime	Mean response time	Confidence Interval
0.05	0.9861	[0.9839, 0.9883]

0.06	0.9308	[0.9276, 0.934]
0.07	0.8873	[0.8844, 0.8902]
0.08	0.8594	[0.8561, 0.8626]
0.09	0.8392	[0.8348, 0.8436]
0.1	0.8215	[0.8184, 0.8247]
0.11	0.8231	[0.8189, 0.8274]
0.12	0.8281	[0.8226, 0.8337]
0.13	0.8628	[0.858, 0.8675]
0.14	0.9167	[0.9029, 0.9304]

We compare the systems in pairs from 0.05 to 0.14 seconds. We can notice that fogLimitTime 0.05, 0.06,0.07,0.08,0.09,0.13,0.14,perform worse than farms with 0.1, 0.11 and 0.12 seconds because these mean response times are longer than the other three and their confidence intervals are not overlapping with these three as well.

The system of 0.1 seconds is in the confidence interval of the 0.11 seconds. It means it is hard to tell the performs yet. So I increase the number of replications to 20.

Table 2 Mean response time and confidence interval with 20 replications

fogLimitTime	Mean response time	Confidence Interval
0.05	0.9865	[0.9857, 0.9874]
0.06	0.931	[0.9301, 0.9318]
0.07	0.8885	[0.8875, 0.8895]
0.08	0.8581	[0.857, 0.8591]
0.09	0.8364	[0.835, 0.8378]
0.1	0.8214	[0.8203, 0.8225]
0.11	0.8228	[0.8216, 0.824]
0.12	0.8305	[0.829, 0.832]
0.13	0.8615	[0.8599, 0.8631]
0.14	0.9113	[0.9075, 0.915]

Based on Table 2, we can find that the confidence intervals of 0.1 and 0.11 seconds are still overlapping so we still need to add more replications.

Table 3 Mean response time and confidence interval with 80 replications

fogLimitTime	Mean response time	Confidence Interval
0.05	0.9896	[0.9891, 0.9901]
0.06	0.933	[0.9326, 0.9333]
0.07	0.8902	[0.8899, 0.8906]
0.08	0.8581	[0.8578, 0.8584]
0.09	0.8368	[0.8364, 0.8373]
0.1	0.823	[0.8226, 0.8233]
0.11	0.8213	[0.8209, 0.8218]
0.12	0.8289	[0.8284, 0.8294]
0.13	0.855	[0.854, 0.8561]
0.14	0.9024	[0.9007, 0.9042]

Based on Table 3, we can find the result we want. In these results, all confidence intervals are not

overlapping so which means if we compare all these systems in pairs, the one with the smallest mean response time is the optimal solution of all models.

In particular, for the systems with 0.1 seconds,0.11 seconds, the fogLimitTime of 0.1 seconds has the response mean time of 0.823 with 95% confidence interval [0.8226, 0.8233] while the fogLimitTime of 0.11 seconds has the mean response time is 0.8213 with 95% confidence interval [0.8209, 0.8218]. Therefore, the fogLimitTime of 0.11 seconds performs better than 0.1.

In conclusion, we can get that the system with the

$$\text{fogLimitTime} = 0.11\,\text{seconds}$$

performs best by giving the least mean response time when

$$\lambda = 9.72, \alpha_1 = 0.01, \alpha_2 = 0.4, \beta = 0.86, \upsilon_1 = 1.2, \upsilon_2 = 1.47, time\_end = 1000$$

and fogTimeCloudTime is 0.6.

.