COMP9334 Project, Term 1, 2019: Fog/cloud Computing

Version 1.0
Due Date: 11:00pm Friday 26 April 2019.

This version: 20 March 2019

Updates to the project, including any corrections and clarifications, will be posted on the subject website. Make sure that you check the course website regularly for updates.

Change log

Nothing for Version 1.0.

1 Introduction and learning objectives

This project is inspired by the research work reported in the article *FogQN: An Analytic Model for Fog/Cloud Computing* [2] on modelling a computing system that makeg use of both fog and cloud computing. The key question studied in the paper is on how to distribute work between two computing resources, namely the fog and the cloud. In this project, you will use simulation to try to answer a similar research question.

In this project, you will learn:

1. To use discrete event simulation to simulate a computer system
2. To use simulation to solve a design problem
3. To use statistically sound methods to analyse simulation outputs

2 Support provided

If you have problems doing this assignment, you can post your question on the message board on the subject website. **We strongly encourage you to do this as asking questions and trying to answer them is a great way to learn. Do not be afraid that your question may appear to be silly, the other students may very well have the same question!**

3 Description of the fog/cloud computer system

3.1 Introduction to fog/cloud computing

You are probably aware that there are many more devices with Internet connectivity nowadays. These new devices include sensors, vehicles, Internet-of-Things (IoT) etc. In general, these devices are referred to as the edge devices. The fog and the cloud are two places to process the data coming from these edges devices, see Figure 1.

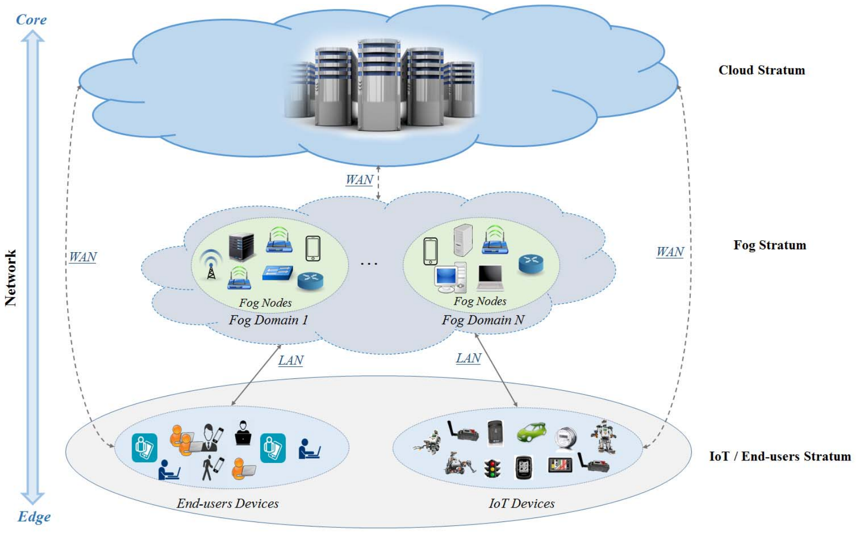

Figure 1: This figure depicts the three layers of edge devices, fog and cloud. The edge devices are the end-user devices and IoT, which is short for Internet-of-things. Note that LAN and WAN stand, respectively, for local area network and wide area network. This diagram shows that the edge devices are closer to the fog computing resource than the cloud computing resource. This figure is taken from [1].

- The **fog** is a computing facility located closer to the edge devices. The network latency to reach the fog computing facility is therefore low. However, the fog has a lower processing speed.
- The **cloud** is a computing facility farther away from the edge devices. The network latency to get to the cloud is higher. However, the processing speed at the cloud is higher than that in the fog.

Given that there are two facilities to process the data from the edge devices, a question is which facility we should make use of in order to reduce the **overall response time**. Note that the **overall** response time of using a facility is the sum of two parts, which are the network latency to reach the facility and the response time of doing the processing at the facility. Now, let us consider the case of doing all the processing at the fog. The network latency will be low but the response time of doing the processing at the fog is high. On the other hand, if we do all the processing at the cloud, then the network latency will be high but the response time of doing the processing at the cloud is low. Neither one of these two options appears to be the best option.

In this project, you will investigate how to distribute work between the fog and the cloud in order to minimise the overall response time. Note that this is all about fog/cloud computing that we will tell you so that you get enough intuition to understand this project. If you want to learn more about the fog/cloud computing paradigm, which is an area of commercial and academic research at the moment, you can consult [1].

3.2 Fog/cloud setup for this project

Figure 2 depicts the fog/cloud system to be considered in this project. The system consists of three components: the fog, the network and the cloud. We assume that all the requests will go to the fog computing facility in the beginning. The fog will process these requests but imposes

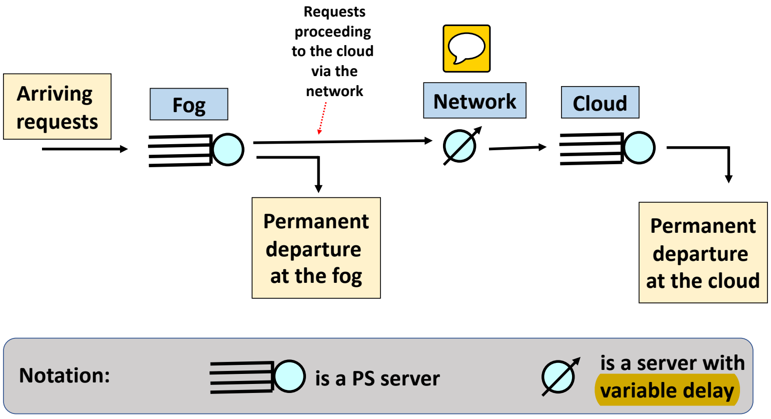
Requests proceeding to the cloud via the network
Arriving requests
Fog
Network
Cloud
Permanent departure at the fog
Permanent departure at the cloud
Notation:
is a PS server
is a server with variable delay

Figure 2: The fog/cloud computing system in this project.

an upper limit on the processing time spent on each request. If a request requires an amount of processing less than or equal to the upper limit, this request will be completed at the fog and leave the system permanently upon completion. On the other hand, if a request requires an amount of processing more than the upper limit, the request will be sent to the cloud via the network and the cloud will complete the rest of the processing. After the cloud finishes the processing, the request will leave the system permanently.

In this project, we model the fog as a processor sharing (PS) server. The same holds for the cloud.

The network is modelled as a server with variable delay. We use the term network latency to refer to this delay. Consider a request which is to be sent to the cloud. If this request leaves the fog at time t and the network latency for this request is x, then this request will arrive at the cloud at $t + x$. For a server with variable delay, the value of x for different requests can be different. Note that there are no queues at the variable delay server modelling the network.

4 Examples

We will now present three examples to illustrate the operation of the fog/cloud system that you will simulate in this project. In all these examples, we assume that the system is initially empty.

4.1 Example 1: Simulating a PS server

In this example, we assume there are 5 requests. The arrival times and service times required at the fog are given in Table 1.

We further assume that the fog will give each arriving request a processing time of no more than 5 time units. Since the service times of the five requests are all less than or equal to this upper limit, all these requests will be completed at the fog and will not be sent to the cloud. This effectively means that, for this example, we only need to simulate the PS server for the fog, and

Arrival time at the fog	Service time required at the fog
1	2.1
2	3.3
3	1.1
5	0.5
15	1.7

Table 1: Data for Example 1.

忽略

we can neglect the network and the cloud. The aim of this example is to show you how to simulate a PS server.

The PS server was discussed in Lecture 4A. When there are n jobs in the server, then each job receives $\frac{1}{n}$ of the service.

The events in a PS server are the arrival of a job to the server and the departure of a completed job from the server. You should convince yourselves that between two consecutive events, → 连续的 the number jobs in a PS server remains the same. The discrete event simulation advances from an event to the next one.

发展

The key data structure that you need to maintain is the list of jobs in the server. Each job is characterised by two attributes: ① the time the job arrives at the server and the ② remaining amount of service the job still needs. Each time when a job arrives or departs, this data structure should be updated. We will explain how this update can be done in a moment.

We will illustrate how the simulation of PS server works using "on-paper simulation". Three of the quantities that you need to keep track of are:

3个变量

- **Next arrival time** is the time of the next arrival
- **Next departure time assuming no more arrivals** is defined as the time of the next departure assuming that no more arrivals will come in the future. For simplicity, we will simply use the phrase next departure time. For example, if there are three jobs in the server at a certain time and these jobs still need 5, 6 and 10 units of service, then the next departure time will be 15 time units later.
- The list of jobs in the server. Each job is characterised by a 2-tuple. The first element of the 2-tuple is the arrival time of the job at the server and the second element is the amount of service it still needs.

tuple
(arrival time, still need service)

The "on-paper simulation" is shown in Table 2. The notes in the last column explain what updates you need to do for each event. Please note that there are more quantities that you need to keep track of than those three that are mentioned above.

A graphical representation of the PS server status over time is given in Figure 3. This graphical representation is the same as the one used in Lecture 4B to explain how you can calculate the departure time of jobs in a PS server. There are three plots in the figure, showing the arrival times, remaining amount of service for each job and the departure times. The figure is best viewed in colour because the quantities related to each job is shown in a specific colour. Note that in between two consecutive events, the remaining amount of service for each job is **not** a constant. What is constant in between two consecutive events is the number of jobs in the server. The number of jobs in the server determines the service rate of each job which is the slope of the remaining service curves. You will see that the slope stays constant in between two events and changes each time an event occurs.

→ 斜率

Arrival time at the fog	Service time required at the fog
1	2.1
2	3.3
3	1.1
5	0.5
15	1.7

⇒.

共2个job,
resource 减半.
时间变为2倍.

2+2×1.1=4.2

Master clock	Event type	Next arrival time	Next departure time	Job list	Notes
					updates
$t = 0$	–	1	∞	–	We assume the server is empty at the start of the simulation and this is indicated by departure time of ∞.
$t = 1$	Arrival	2	3.1 1+2.1 =3.1 t, 离开	(1,2.1) arrive still need	Since the event is an arrival, we need to update (i) Next arrival time; (ii) Job list; and (iii) Next departure time. There is one job in the list. The notation (1,2.1) means the job arrives at $t = 1$ and at the time of the master clock, it still needs 2.1 units of service. If there are no more arrivals, the next departure is expected to be at time 3.1 and this is the next departure time.
$t = 2$	Arrival	3	4.2 3.1−2=1.1 2+1.1×2 =2+2.2 =4.2.	(1,1.1), (2,3.3)	Since the event is an arrival, we need to update (i) Next arrival time; (ii) Job list; and (iii) Next departure time. Note that for the job list, you need to add the new job to the list and you also need to update the amount of service still needed by those jobs that were in the server before the arrival of this job. We now explain how the job (1.2.1) is updated to (1,1.1). At the time of the last event (which was $t = 1$), this job needed 2.1 units of service. Given that the current time is $t = 2$, which means 1 time unit has lapsed. Since there was only one job in the server between $t = 1$ and $t = 2$, the job received one unit of service hence its remaining service time is $2.1 - 1 = 1.1$. This means that your simulation needs to remember the time of the last event.
$t = 3$	Arrival	5	4.8 3+0.6×3 =4.8	(1,0.6), (2,2.8), (3,1.1)	One unit of time has lapsed since the last event and there were 2 jobs in the server, so the jobs in the server received 0.5 units of service each.
$t = 4.8$	Departure	5	5.8	(2,2.2), (3,0.5) 2.8−2.2 =0.6	Since the event is a departure, you will need to update the (i) Job list; and (ii) Next departure time.
$t = 5$	Arrival	15	6.2	(2,2.1), (3,0.4), (5,0.5)	
$t = 6.2$	Departure	15	6.4	(2,1.7) (5,0.1)	
$t = 6.4$	Departure	15	8	(2,1.6)	
$t = 8$	Departure	15	∞	–	

Table 2: Table illustrating the updates in a PS server.

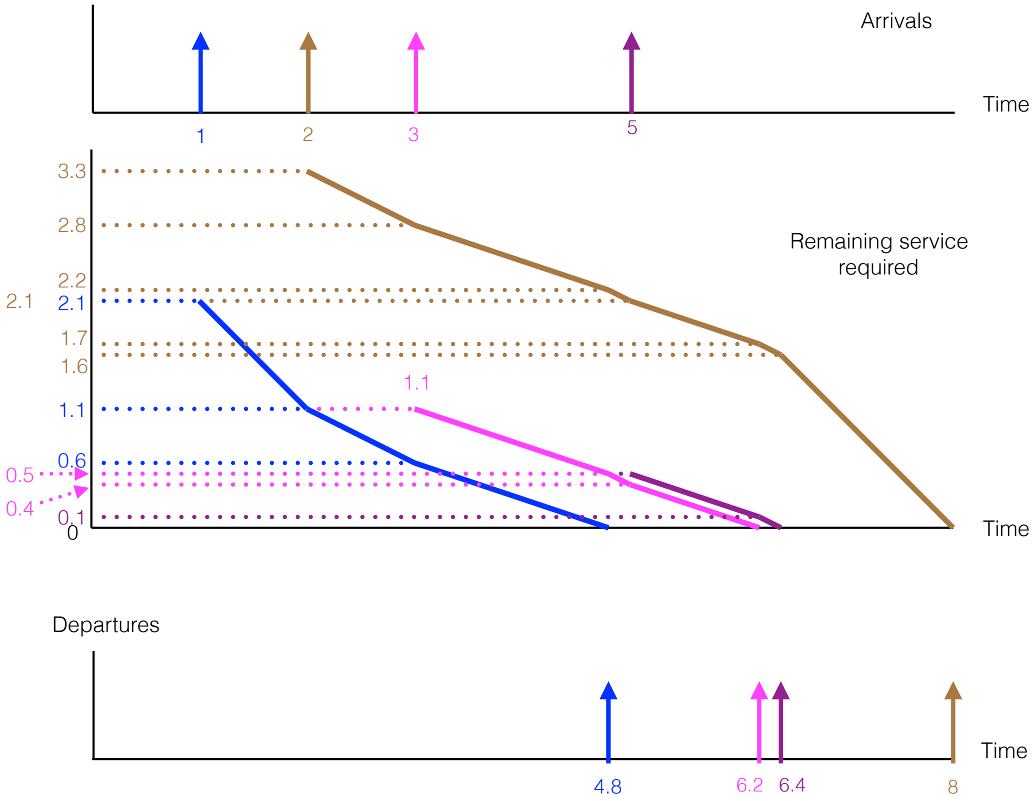

Figure 3: Figure illustrating PS.

4.2 Example 2: Simulating the fog, the network and the cloud

This example illustrates the simulation of the fog/cloud system. We label the requests with indices 1, 2, 3 and so on; these indices are in the first column of Table 3. The second column of the table shows the arrival times. The third column shows the amount of processing required and this amount is measured by the time the request will need if all the processing is done in the fog. Since the amount of time is measured with respect to the fog, we refer to it as the service time in the fog time unit.

索引

Request	Arrival time at the fog	Service time in the fog time unit
1	1.1	4.1
2	6.2	5.2
3	7.4	1.3
4	8.3	2.0
5	9.1	3.2
6	10.1	4.1

→ 完全在fog

Table 3: Arrival times and service times in the fog time unit for Example 2.

处理上限是2

For this example, we assume that the fog will only provide no more than 2 time units of service to each request. This means that:

- If a request requires 2 time units or less service from the fog, then the request will be completed entirely at the fog. For the requests in Table 3, both Requests 3 and 4 will be entirely completed at the fog.
- If a request requires more than 2 time units of service from the fog, the request will get 2 time units of service from the fog. After receiving a service of 2 time units, the request will be sent to the cloud via the network. The remaining work of the request will be completed at the cloud. For the requests in Table 3, Requests 1, 2, 5 and 6 will each receive 2 time units of service at the fog before they are sent to the cloud.

先在 fog 处理
2 time, 再传到
network → cloud 里

In general, we use the parameter fogTimeLimit to denote the maximum amount of service a request can receive from the fog. For this example, the value of the parameter fogTimeLimit is 2.

In order to simulate those requests that are sent to the cloud, we will need their network latency and the service time they need at the cloud. For this example, the network latency for Requests 1, 2, 5 and 6 are given in the fourth column of Table 4.

We will now explain how the service time at the cloud is to be computed. Consider a request whose service time in the fog time unit is x where $x > \text{fogTimeLimit}$, the amount of time required by this request at the cloud is given by:

$$\text{fogTimeToCloudTime} * (x - \text{fogTimeLimit})$$

→ 在 cloud 处理所需时间

where fogTimeToCloudTime is a parameter. In this example, we assume the value of the parameter fogTimeToCloudTime is 0.6. Therefore, Request 1 will need $0.6 * (4.1 - 2) = 1.26$ time units of service at the cloud. Note that the quantity $(x - \text{fogTimeLimit})$ represents the work that this request still needs. This work is measured in terms of the amount of time needed at the fog, but since this work is now to be completed at the cloud and the cloud is a faster computing facility, so we need to convert the amount work to the time needed at the cloud. We model this conversion by using the multiplication factor fogTimeToCloudTime. You can expect the value of fogTimeToCloudTime is less than 1 because the cloud has a higher processing speed than the fog.

By using the setup above, Table 4 shows the actual time each request gets at the fog and the service time needed at the cloud. Note that the actual time a request gets at the fog (third column

$$\text{fogTimeToCloudTime} * (x - \text{fogTimeLimit})$$

Request	Arrival time at the fog	Service time in the fog time unit
1	1.1	4.1
2	6.2	5.2
3	7.4	1.3
4	8.3	2.0
5	9.1	3.2
6	10.1	4.1

in Table 4) is the smaller of the service time in the fog time unit (third column in Table 3) and the value of the parameter fogTimeLimit. Note also that the service time needed at the cloud (fifth column in Table 4) is deduced from the service time in the fog time unit, together with the parameters fogTimeLimit and fogTimeToCloudTime.

推断

推断出的

Request	Arrival time at the fog	**Actual** service time provided at the fog	network latency	Service time required at the cloud
1	1.1	2	1.5	1.26
2	6.2	2	1.3	1.92
3	7.4	1.3		
4	8.3	2		
5	9.1	2	1.6	0.72
6	10.1	2	1.8	1.26

0.6 x (2.1) = 1.26.

Table 4: This table shows arrival time at the fog, actual service time provided at the fog, network latency and service time required at the cloud for Example 2.

In the simulation of the fog/cloud system, there are five events:

1. Arrival at the fog 到达 fog
2. Departure from the fog (without going to the network) 完全离开 fog. 结束.
3. Departure from the fog to the network 从 fog 离开去 network.
4. Departure from the network to the cloud 从 network 离开去 cloud.
5. Departure from the cloud 离开 cloud.

Table 5 shows the event times in ascending order with an explanation of the events.

Request	Arrival time at the fog	**Actual** service time provided at the fog	network latency	Service time required at the cloud
1	1.1	2	1.5	1.26
2	6.2	2	1.3	1.92
3	7.4	1.3		
4	8.3	2		
5	9.1	2	1.6	0.72
6	10.1	2	1.8	1.26

0.6 × (2.1

Request	Arrival time at the fog	Service time in the fog time unit
1	1.1	4.1
2	6.2	5.2
3	7.4	1.3
4	8.3	2.0
5	9.1	3.2
6	10.1	4.1

Time	Request	Event
1.1000	1	Arrival at the fog
3.1000	1	Departure from the fog to the network
4.6000	1	Departure from the network to the cloud. Note that Request 1 departed from the fog at time 3.1 and the network latency for this request is 1.5, therefore this request departs from the network at 3.1 + 1.5 = 4.6.
5.8600	1	Departure from the cloud. Note that Request 1 arrived at the cloud at time 4.6 and with no other jobs arriving at the cloud when it was served, it departed from the cloud at 4.6 + 1.26 = 5.86.
6.2000	2	Arrival at the fog
7.4000	3	Arrival at the fog
8.3000	4	Arrival at the fog
9.1000	5	Arrival at the fog
9.4333	2	Departure from the fog to the network. Note that by this time, Request 2 has received 2 (= 1.2 + 0.9 / 2 + 0.8 / 3 + 0.3333 / 4) units of service. Since the value of the parameter fogTimeLimit is 2, this request has to depart from the fog.
10.1000	6	Arrival at the fog
10.7333	2	Departure from the network to the cloud
11.2111	3	Departure from the fog
12.6533	2	Departure from the cloud
14.6611	4	Departure from the fog
15.1944	5	Departure from the fog to the network
15.5000	6	Departure from the fog to the network
16.7944	5	Departure from the network to the cloud
17.3000	6	Departure from the network to the cloud
17.7289	5	Departure from the cloud. Request 5 requires 0.72 units of service from the cloud. At time 17.7289, the amount of service it has received is (17.3000 − 16.7944) + (17.7289 − 17.3000)/2 = 0.72.
18.7744	6	Departure from the cloud

Table 5: The event times for Example 2.

3.1 = 1.1 + 2

5.86=4.6+1.26

等价完全处理完, departure.

1.9167

1.2+0.45+ 0.2667

+ 0.083825

= 2.000555

1.2

0.9

0.8

0.333

完全结束

1.82

event 2 at cloud

1.73000 后 有 5 和 6.

⇒ 除以 2.

追踪模拟

4.3 Example 3: Trace driven simulation

This example illustrate the data that you will be given if you are asked to do a trace-driven simulation. You will be given: cloud/fog

上限

- The values of fogTimeToCloudTime and fogTimeLimit.
- For each request, you will be given the arrival time at the fog, service time in the fog time unit and network latency. Note that for a request whose service time in the fog time unit is no more than the fogTimeLimit, we use a zero network latency to indicate this request will not be sent to the cloud.

当全在fog处理时，network latency为0.

The data for this example are:

Arrival time at the fog	Service time in the fog time unit	network latency
1	3.7	1.5
2	5.1	1.4
4	1.3	0
5	2.4	0
6	4.5	1.6

fogTimeLimit = 2.5
fogTimeToCloudTime = 0.7

With the data above, the events in the fog/cloud system are shown in the table below. Note that we have labelled the requests in the order that they arrive at the fog.

Time	Request	Event
1	1	Arrival at the fog
2	2	Arrival at the fog
4	3	Arrival at the fog
5	4	Arrival at the fog
5.6667	1	Departure from the fog to the network $1 + 2 \times \frac{1}{2} +$
6	5	Arrival at the fog
7.1667	1	Departure from the network to the cloud
8.0067	1	Departure from the cloud
8.7556	3	Departure from the fog
9.3556	2	Departure from the fog to the network
10.7556	2	Departure from the network to the cloud
11.8222	4	Departure from the fog
12.2	5	Departure from the fog to the network
12.5756	2	Departure from the cloud
13.8	5	Departure from the network to the cloud
15.2	5	Departure from the cloud

$1+1+ \frac{1}{3} + \frac{2}{12} = 2.5$

$1 \times \frac{1}{3} + \frac{2}{3} \times \frac{1}{4} =$

You can compute the system response times by using the arrival and departure times from the system. The following table shows the arrival and departure times that can be used to compute the system response times. The mean response time is 7.6720 without considering transient removal.

Arrival time	Departure time
1	8.0067
2	12.5756
4	8.7556
5	11.8222
6	15.2

→ mean Response 怎么算出来的?

5 Project description

This project consists of two main parts. The first part is to develop a simulation program for the fog/cloud system which is described in Section 3.2 and illustrated in Section 4. In the second part, you will use the simulation program that you have developed to solve a design problem.

5.1 Simulation program

You must write your simulation program in one (or a combination) of the following languages or numerical package: Python (either version 2 or 3), C, C++, Java or Matlab. All these languages and numerical packages are available on the CSE system. We will test your program on the CSE system so your submitted program **must** be able to run on a CSE computer. Note that it is possible that due to version differences, code that runs on your own computer may not work on the CSE system. It is your responsibility to ensure that your code works on the CSE system. In particular, if you plan to use Matlab, you need to know that CSE runs Matlab 2013, which is unfortunately older than expected.

比一般版本老!

7个输入参数.

We require you to write your simulation program as a **function** in your chosen language. Your function must have the following seven inputs:

mode :模式

random

trace driven

1. A parameter `mode` of string type. This parameter is to control whether your program will run simulation using randomly generated arrival times, service times and network latencies; or in trace driven mode. The value that the parameter `mode` can take is either `random` or `trace`.
2. A parameter `arrival` for supplying arrival information to the program. The meaning of `arrival` depends on `mode`. We will provide more information later on. 和参数有关
3. A parameter `service` for supplying service time in the fog time unit to the program. The meaning of `service` depends on `mode`. We will provide more information later on.
4. A parameter `network` for supplying network latency to the program. The meaning of `service` depends on `mode`. We will provide more information later on.
5. A parameter `fogTimeLimit` to input the value of fogTimeLimit. This parameter is a positive floating point number.
6. A parameter `fogTimeToCloudTime` to input the value of fogTimeToCloudTime. This parameter is a positive floating point number.
7. A parameter `time_end` which stops the simulation if the master clock exceeds this value. 总处理时间 This parameter is only relevant when `mode` is `random`. This parameter is a positive floating point number.

Note that your simulation program must be a general program which allows different values to be used. When we test your program, we will vary the parameter values. You can assume that we will only use valid inputs for testing. You can have additional parameters if you wish.

允许加入别的参数.

For the simulation, you can always assume that the system is empty initially.

一开始是空的.

Note that we assume that it takes negligible time to send the requests to the fog.

可忽略不计的

5.1.1 The random mode

When your simulation is working in the `random` mode, it will generate the **inter-arrival** times, service times in the fog time unit and network latencies in the following manner.

到达间隔.

Request	Arrival time at the fog	Service time in the fog time unit
1	1.1	4.1
2	6.2	5.2
3	7.4	1.3
4	8.3	2.0
5	9.1	3.2
6	10.1	4.1

指数分布

平均到达率是λ.

1. The inter-arrival probability distribution is exponentially distributed with parameter λ. This means the mean arrival rate of the jobs is λ. You will need to supply the value of λ to your program using the input parameter `arrival`. → λ

2. The ~~service time~~ in the fog time unit t is generated by the probability density function $g(t)$ where:

   概率密度方程.

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

   where

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

   Note that this probability density function has three parameters: α_1, α_2 and β.

   均匀分布

3. The network latency is uniformly distributed in the open interval (ν_1, ν_2) where $\nu_2 > \nu_1 > 0$. Note that network latency only applies to requests that are to be sent to the cloud.

5.1.2 The trace mode

When your simulation is working in the `trace` mode, it will read the list of arrival times, the list of service times in the fog time unit and the list of network latencies from three separate ASCII files.

参数确定，从文件中读取.

Let us use Example 3 in Section 4.3 for an illustration. The arrival times are $[1, 2, 4, 5, 6]$, the service times in the fog time unit are $[3.7, 5.1, 1.3, 2.4, 4.5]$ and the network latencies are $[1.5, 1.4, 0, 0, 1.6]$. Your program is required to simulate until all jobs have departed. For this example, the last departure occurs at time 15.2 and you can stop the simulation there.

模拟直至所有job都结束

We will supply the arrival times, service times in the fog time unit and network latencies to you using three ASCII files. The names of these three files have specific format, which we will discuss in Section 6 on testing. For Example 3 in Section 4.3, the service times in the fog time unit will be specified by a file whose contents are as follows:

```
3.700
5.100
1.300
2.400
4.500
```

占一行

where each service time takes up a line of the file. The same format is used for arrival times and network latencies. You can assume that the data we provide for `trace` mode are consistent in the following way: (1) The number of arrival times, the number of service times in fog unit and the number of network latencies are equal. (2) If the service time in fog unit for a request is not greater than fogTimeLimit, then the corresponding network latency is zero.

5.2 Determining a suitable value of fogTimeLimit

After writing your simulation program, your next step is to use your simulation program to do a design.

For this design problem, you will assume the following parameter values: $\lambda = 9.72$, $\alpha_1 = 0.01$, $\alpha_2 = 0.4$, $\beta = 0.86$, $\nu_1 = 1.2$, $\nu_2 = 1.47$ and fogTimeToCloudTime is 0.6.

一堆参数

Your aim is to determine the value (or a range of values) of the parameter fogTimeLimit that gives the best system response time. Recall from Section 3.1 that we will not get the best overall response time if we do all the processing at the fog or all the processing at the cloud. Doing all the processing at the fog corresponds to setting fogTimeLimit to infinity. Doing all the processing at the cloud corresponds to setting fogTimeLimit to 0. This means that for some non-zero value of fogTimeLimit, you can get a better overall response time.

In solving this design problem, you need to ensure that you use **statistically sound** methods to compare systems. You will need to consider parameters such as length of simulation, number of replications, transient removals and so on. You will need to justify in your report how you choose the value or range of fogTimeLimit.

什么意思,
?

重复

暂时的

6 Testing simulation program

As part of the assessment for this project, you are asked to **attend an interview** so that we can test your simulation program. During the interview, we will ask you to run a series of tests. Each test is specified by a number of configuration files. For each test, we require four output files of specific format so that we can verify the correctness of your simulation program. The aim of this section is to specify the format of the configuration files and output files.

The number of tests that you will need to run is specified in an ASCII file with the name `num_tests.txt`. If we want to run 12 tests in the interview, the file `num_tests.txt` contains the string `12` only.

Each test is specified by 5 configurations files. We will index the tests from 1. If 12 tests are used, then the indices for the tests are 1, 2, ..., 12. The names of the configuration files are:

- For Test 1, the configuration files are `mode_1.txt`, `para_1.txt`, `arrival_1.txt`, `service_1.txt` and `network_1.txt`. The files are similarly named for indices 2, 3, ..,9.
- For Test 10, the configuration files are `mode_10.txt`, `para_10.txt`, `arrival_10.txt`, `service_10.txt` and `network_10.txt`. The files are similarly named if the test index is a 2-digit number.

We will refer to these files using the generic names `mode_*.txt`, `para_*.txt` etc.

6.1 Wrapper file

When you submit your project, you must include a **wrapper** file which will loop through all the tests. We will use this wrapper file in your interview to run your simulation. The pseudo-code for your wrapper file is as follows.

```
Read the file num_tests.txt to determine the number of tests

FOR test_index from 1 to num_tests
    Read in the configuration files for the test_index
    Set the simulation mode and parameter values
    Call your simulation function
    Write the output files % This can be in the wrapper or your simulation function
```

The filename of this wrapper file must be called `wrapper` followed by the appropriate file extension for the language that you used, e.g. `wrapper.py` for Python.

Note that another purpose of the wrapper file is to show us how to run your code. We may need to run additional tests and the wrapper file will allow us to do that. You should have comments in the wrapper file to explain to us how to run your code.

6.2 Configuration file format

6.2.1 mode_*.txt

This file is to indicate whether the simulation should run in the random or trace mode. The file contains one string, which can either be random or trace.

6.2.2 para_*.txt

If the simulation mode is trace, then this file has two lines. The first line is the value of fogTimeLimit. The second line is the value of fogTimeToCloudTime. If the test is Example 3 in Section 4.3, then the contents of this file are:

```
2.5
0.7
```

If the simulation mode is random, then this file has three lines. The first two lines contain, respectively, the values of fogTimeLimit and fogTimeToCloudTime. The last line contains the value of time_end.

6.2.3 arrival_*.txt

The contents of the file arrival_*.txt depend on the mode of the test. If mode is trace, then the file arrival_*.txt contains the arrival times of the requests with one arrival time occupying one line. You can assume that the list of arrival times are strictly increasing. For Example 3 in Section 4.3, this file will be

```
1.000
2.000
4.000
5.000
6.000
```

If the mode is random, then the file arrival_*.txt contains a string for a floating point number. This number corresponds to the value of λ.

6.2.4 service_*.txt

For trace mode, the file service_*.txt contains one service time in the fog time unit per line.

For random mode, the file service_*.txt contains three lines, corresponding to the values of the α_1, α_2 and β.

6.2.5 network_*.txt

For trace mode, the file network_*.txt contains one network latency per line.

For random mode, the file network_*.txt contains two lines, corresponding to the values of the ν_1 and ν_2.

6.3 Output file format

In order to test your simulation program, we need four output files **per test**. One file containing the mean response time. The other three files contain the departure times from the fog, the network and the cloud.

The mean response time without considering transient removal (a scalar value) should be written to a file whose filename has the form `mrt_*.txt`. For Example 3 in Section 4.3, the contents of this file are:

```
7.6720
```

The second file contains the departure times of the requests from the fog. The filename should be of the form `fog_dep_*.txt`. For Example 3 in Section 4.3, the contents of the file are:

```
1.0000 5.6667
2.0000 9.3556
4.0000 8.7556
5.0000 11.8222
6.0000 12.2000
```

The third file contains the departure times of the requests from the network. The filename should be of the form `net_dep_*.txt`. For Example 3, the contents of the file are:

```
1.0000 7.1667
2.0000 10.7556
6.0000 13.8000
```

The fourth file contains the departure times of the requests from the cloud. The filename should be of the form `cloud_dep_*.txt`. For Example 3, the contents of the file are:

```
1.0000 8.0067
2.0000 12.5756
6.0000 15.2000
```

Note the following requirements for the files containing the departure times:

1. Each line contains the arrival time and departure time of a request. The arrival time is printed first, followed by a tab, then the departure time from that particular component of the system.

2. The requests must be ordered according to the ascending time of the arrival time at the fog. For example, for Example 3 in Section 4.3, the request that arrives at the fog at time 2.0000 will depart from the fog at 9.3556, from the network at 10.7556 and from the cloud at 12.5756. Note that the reason why we use the arrival time at the fog is because each request has a unique arrival time at the fog. Therefore we can use the arrival time at the fog as a unique identification for each request. Note that our way of simulating a PS server allows multiple jobs to leave the PS server at the same time, therefore the arrival times at the network may not be distinct.

3. If the simulation is in the `trace` mode, we expect the simulation to finish after all requests have left the system. Therefore, each departure file should contain all the requests that have left that particular component.

4. If the simulation is in the `random` mode, each departure file should contain all the requests that have left that particular component by `time_end`.

All numbers in `mrt_*.txt`, `fog_dep_*.txt`, `net_dep_*.txt` and `cloud_dep_*.txt` should be printed as floating point numbers to exactly 4 decimal places.

6.4 Sample files

The following sample files are available on the project website:

- The file `num_tests.txt` containing the string 4 for 4 tests.
- The files `mode_1.txt`, `mode_2.txt`, `mode_3.txt` and `mode_4.txt`. Note that Tests 1, 2 and 3 are for `trace` mode while Test 4 is for `random` mode.
- The files `para_*.txt`, `arrival_*.txt`, `service_*.txt`, `network_*.txt` for `*` from 1 to 4, as the input to the simulation.
- The files `mrt_*_ref.txt`, `fog_dep_*_ref.txt`, `net_dep_*_ref.txt` and `cloud_dep_*_ref.txt` for `*` from 1 to 4, as the reference files for the output. For Tests 1, 2 and 3, you should be able to reproduce the results in `mrt_*_ref.txt`, `fog_dep_*_ref.txt`, `net_dep_*_ref.txt` and `cloud_dep_*_ref.txt`. However, since Test 4 is in `random` mode, you will not be able to reproduce the results in the output files. They have been provided so that you can check the expected format of the file.
- The Python file `cf_output_with_ref.py` which shows you how we may compare your output against the reference output in `trace` mode.

Note that Tests 1 and 2 are the same as, respectively, Example 2 and Example 3 in Section 4.

6.5 Test for reproducibility

We require that your simulation experiments are **reproducible** when operating in `random` mode. We require you to submit at least three sets of sample output files. Each set of output files should include four files: a file similar to `mrt_*.txt` (which contains the mean response time) and a file similar to the files `fog_dep_*.txt`, `net_dep_*.txt` and `cloud_dep_*.txt` (which contains the departure times.) We will pick from these sample output files and ask you to reproduce the results in the interview.

7 Project requirements

This is an individual project. You are expected to complete this project on your own.

7.1 Submission requirements

Your submission should include the following:

1. A written report
   (a) Only soft copy is required.
   (b) It must be in Acrobat pdf format.
   (c) It must be called ”report.pdf”.
2. Program source code:
   (a) For doing simulation
   (b) The wrapper file, see Section 6.1
3. Sample outputs for reproducibility testing, see Section 6.5
4. Any supporting materials, e.g. logs created by your simulation, scripts that you have written to process the data etc.

The assessment will be based on your submission and the tests conducted at your interview. Note that the interview is based on the code that you submitted and will be run on a CSE computer. It is important that you submit the right version of the code and make sure that it runs on the CSE system. In the interview, we do **not** accept code that is not the submitted version and we do **not** accept code running on your computer.

It is important that you write a clear and to-the-point report. You need to aware that you are writing the report to the marker (the intended audience of the report) not for yourself. Your report will be assessed primarily based on the quality of the work that you have done. You do not have to include any background materials in your report. You only have to talk about how you do the work and we have provided a set of assessment criteria in Section 7.3 to help you to write your report. In order for you to demonstrate these criteria, your report should refer to your programs, scripts, additional materials so that we are aware of them.

7.2 Interview

You are also required to demonstrate your work in an interview in Week 11. Further details on the interview will be available later. In the interview, we will ask you to run a number of tests and to demonstrate that your work is reproducible.

7.3 Assessment criteria

We will assess the quality of your project based on the following criteria:

1. The correctness of your simulation code. For this, we will:
   (a) Test your code using test cases in an interview and through additional testing if necessary
   (b) Look for evidence in your report that you have verified the correctness of the inter-arrival probability distribution, service time distribution and network latency distribution. You can include appropriate supporting materials to demonstrate this in your submission.
   (c) Look for evidence in your report that you have verified the correctness of your simulation code. You may derive test cases such as those in Section 4 to test your code. You can include appropriate supporting materials to demonstrate this in your submission.
2. You will need to demonstrate that your results are reproducible. We will ask you to do that at your interview. In addition, you should provide evidence of this in your report.
3. For the part on determining a suitable value of fogTimeLimit, we will look for the following in your report:
   (a) Evidence of using statistically sound methods to analyse simulation results
   (b) Explanation on how you choose your simulation and data processing parameters, e.g lengths of your simulation, number of replications, end of transient etc.

7.4 How to submit

You should “tar”, “rar” or “zip” your report, programs and supporting materials into a file called “project.tar”, “project.rar” or “project.zip”. The submission system will only accept one of these filenames.

You should submit your work via the course website. Note that the maximum size of your submission should be no more than 20MBytes.

You can submit multiple times before the deadline. The latest submission overrides the earlier submissions, so make sure you submit the correct file. Do not leave until the last moment to submit, as there may be technical or communication error and you will not have time to rectify.

8 Plagiarism

You should be aware of the UNSW policy on plagiarism. Please refer to the course web page for details.

References

[1] Carla Mouradian et al. "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges", IEEE Communications Surveys and Tutorials, Vol 20, No 1, 1st Quarter, 2018. https://ieeexplore.ieee.org/document/8100873

[2] Uma Tadakamalla and Daniel A. Menasce. "FogQN: An Analytic Model for Fog/Cloud Computing", Proc. 1st Workshop on Managed Fog-to-Cloud (mF2C), 2018. https://ieeexplore.ieee.org/document/8605797.