COMP9334 Project, Session 1, 2016: Simulation of a multi-server system

Version 1.0
Due Date: 11:59pm Sunday 22 May 2016

This version: 28 Apr 2016

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

1 Introduction and learning objectives

In capacity planning and performance evaluation, there are three major classes of techniques that one can use to understand the performance of a system, namely:

1. Building analytical models (e.g. queueing models)
2. Building simulation models
3. Building the actual system and test it!

In COMP9334, you have learnt how to build queueing models and simulation programs to investigate the performance (in terms of waiting time or response time) of computer networks and systems. What you have learnt by now is that not all queueing models are analytically tractable and simulation may be used in these situations to analyse the performance of a computer system.

In this project, you will learn how to build a simulation model of a multi-server system and use rigorous statistical techniques to analyse your simulation data. The system to be modelled is what is known as a fork-join system in queueing literature. Fork-join systems can be used to model web services, see [1].

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!**

You may also attend the LiC's consultation on Fridays 1:30-2:30pm.

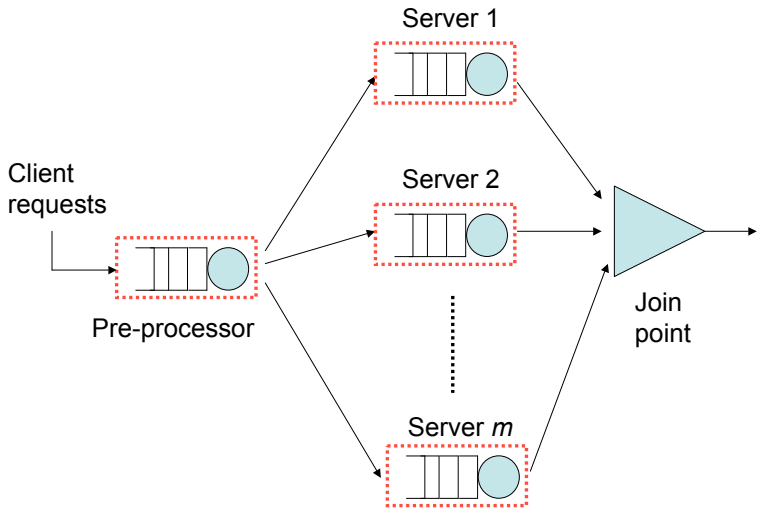

Figure 1: The multi-server system to be simulated in this project.

3 System description

Figure 1 depicts the computer system which you will be simulating in this project. The system consists of a pre-procesor, m servers (labelled as Server 1, Server 2, ..., Server m in Figure 1) and a join point (which is shown as a triangle in Figure 1).

The entry point of the client request to the system is the pre-processor. The pre-processor has a few functions:

- The pre-processor breaks a client request into n sub-tasks where n is a positive integer less than or equal to m.
- In case $n < m$, the pre-processor randomly selects n distinct servers out of m servers to process the n sub-tasks that it has generated. In case $n = m$, all m servers will be selected.
- The pre-processor sends one sub-task to each of the n selected servers for further processing. You may assume that it takes negligible time to send the sub-tasks from the pre-processor to the selected servers.

The function of the servers is to process to sub-tasks assigned to them by the pre-processor. Once a server has finished processing a sub-task, it will send the result to the join point. You may assume that it takes negligible time to send the results of the sub-tasks from the servers to the join point.

Example 1 *Let us assume that* $m = 10$ *and* $n = 3$. *Let us assume a client request* `Request(1)` *arrives at the pre-processor. The pre-processor breaks* `Request(1)` *into 3 sub-tasks, which we will refer to as* `Sub-task(1,1)`, `Sub-task(1,2)` *and* `Sub-task(1,3)`. *Let us assume that the pre-processor has chosen Servers 3, 5 and 10 to process these three sub-tasks. The pre-processor then sends* `Sub-task(1,1)` *to Server 3,* `Sub-task(1,2)` *to Server 5 and* `Sub-task(1,3)` *to Server 10 for processing. (Remark: You will see later that it does not really matter which sub-task will be sent to which server, it will not affect the work that you will do. The key set-up here is one sub-task to each of the n selected servers.) After Server 3 has finished processing* `Sub-task(1,1)`, *it will send the result to the join point. Servers 5 and 10 behave in a similar way.* □

The aim of the join point is to assemble all the results from the sub-tasks of a request into a final result. However, a join point can only perform (or start) the assembling when all the results from all the sub-tasks of a client request has been received by the join point.

Example 2 *Following on Example 1 above. Let us assume that the system is empty when* `Request(1)` *arrives. (The system is empty means there are no jobs in the pre-processor, all the servers and the join point.)*

Let us assume that `Request(1)` *arrives at time* $t = 0$ *and it requires* 4 *time units for processing at the pre-processor. Note that this processing time includes the time for the pre-processor to break a client request into sub-tasks and to select 3 servers for processing the sub-tasks. We assume that it takes negligible time to send the sub-tasks from the pre-procesor to the selected servers. Based on these assumptions, all the three sub-tasks from* `Request(1)` *leave the pre-processor at time* $t = 4$ *and are promptly received by Servers 3, 5 and 10.*

Let us assume that Server 3 needs 6 time units to process `Sub-task(1,1)`, *Server 5 needs 2 time units to process* `Sub-task(1,2)` *and Server 3 needs 8 time units to process* `Sub-task(1,3)`. *Since we assume the system is empty, that means the results of* `Sub-task(1,1)`, `Sub-task(1,2)` *and* `Sub-task(1,3)` *will be sent to the join point at time* $t = 10$, *6 and 12 time units, respectively. We assume that it takes negligible time for a server to send results from the server to the join point. This means that the join point receives the results from* `Sub-task(1,1)`, `Sub-task(1,2)` *and* `Sub-task(1,3)` *also at* $t = 10$, *6 and 12 time units, respectively.*

Since the join point can only do the assembling to obtain the final result of a request when it has received the results of all the sub-tasks of a request, the join point can only start the assembling of the final result of `Request(1)` *at time* $t = 12$ *units.* □

We assume that it takes negligible time for the join point to assemble the results of the sub-tasks of a request into the final result. Once, the final result of a request is obtained, the join point delivers it to the client. We also assume that it takes negligible time for the join point to deliver the final result of a request to the client. We consider a request has completed and left the system when its final result has been delivered by the join point to the client. The response time of a request is defined as the difference between the time at which it leaves the system and the time at which it enters the system.

Example 3 *Following on Example 2 and according to the assumptions that it takes the join point negligible time to assemble and deliver the final result, the join point delivers the final result of* `Request(1)` *at time* $t = 12$ *units. This is also the time at which* `Request(1)` *leaves the system. The response time of* `Request(1)` *is therefore* $t = 12$ *units since it enters the system at* $t = 0$. □

4 Simulation and system design

In this project, you will develop a simulation model to investigate the performance of the system described in Section 3.

In this simulation, you will assume the following:

1. The inter-arrival time sequence $\{a_1, a_2, \ldots, a_k, \ldots, ...\}$ of the client requests has the following properties:
   (a) Each a_k is the sum of two random numbers a_{1k} and a_{2k}, i.e $a_k = a_{1k} + a_{2k}$ $\forall k = 1, 2, ...$
   (b) The sequence a_{1k} is exponentially distributed with a mean arrival rate 0.85 requests per time unit.
   (c) The sequence a_{2k} is uniformly distributed in the interval $[0.05, 0.25]$.

   Note: The easiest way to generate the inter-arrival time sequence is to sum an exponentially distributed random number with the given rate and a uniformly distributed random number in the given range. It would be more difficult to use the inverse transform method in this case, though it is doable.

2. The service time required by a client request at the pre-processor is exponentially distributed with a mean service time of $\frac{n}{10}$ per request where n is the number of servers selected to process the request.

3. The service time t_s required by each sub-task at each server is independently and identically distributed. The probability density function $f(t_s)$ of the service time t_s at the server is given according to the Pareto distribution and is given by:

$$f(t_s) = \begin{cases} 0 & \text{for } t_s < \frac{t_m}{n^{1.65}} \\ \frac{1}{n^{1.65k}} \frac{k t_m^k}{t_s^{k+1}} & \text{for } t_s \geq \frac{t_m}{n^{1.65}} \end{cases} \quad (1)$$

   where $t_m = 20\frac{k-1}{k} \approx 10.3846$, $k = 2.08$ and n is the number of servers chosen to serve the client request. Note that with the above probability density function, t_s can only take values in the range $[\frac{t_m}{n^{1.65}}, \infty)$.

4. You can assume that all the time variables given above are measured in the same time unit.

5. The pre-processor and the servers use first-come-first-serve queueing discipline.

6. The pre-processor, the servers and the join point has infinite buffer.

7. The pre-processor randomly chooses n out of m servers to process the sub-tasks. There is an equal probability that the pre-processor will choose each of these m servers.

8. You can assume $m = 10$.

We will refer to the system with the above assumptions as the **reference system**. You may have noticed that the system parameters are such that if you break the requests into many sub-tasks (i.e. large n), it requires more processing time at the pre-processor to do the breaking down but subsequently each sub-tasks requires less time if n is bigger. The converse happens for n being small. Consequently, when n is small, the performance bottleneck of the reference system is the servers. However, when n is large, the bottleneck is the pre-processor. Therefore, the reference system is not expected to perform well, i.e have poor response time, when n is small nor large. The reference system is probably going to have better response time for some intermediate value of n. Your goal for this project is to determine the value (or values) of n to be used in the **reference system** to give it the smallest mean response time.

In order to achieve this goal, you need to do the following:

1. Write a computer program or routine that generate the required probability distributions.

2. Write a computer program that can simulate the reference system for different choices of n. By using this program, you can test out the effect of different values of n on the mean response time of the reference system. This will allow you to find suitable value(s) of n.

□

In doing this project, you should bear in mind that a learning objective is to learn how to do rigorous simulation, especially in choosing the right parameters in simulation. You will need to decide on how long you should run the simulation for, how many replications you should do etc. You will need to use **statistically sound method** to compare two systems. Of course, you will be required to justify your choices of simulation parameters and explain why you think the value (values) of n that you have chosen is (are) good.

This project requires you to write some computer programs. You can do this project using any computer language (C,C++,Jave etc.) or any mathematical package (Excel, Matlab, Scilab, Octave etc.) you perfer. You may also use discrete simulation packages such as OMNet++, OPNET etc. The choice is entirely yours.

You are also allowed to re-use or use any programs that you can find in the public domain to do this project. For example, if you find a Matlab programs that can simulate the first-come first-serve queue without even doing any modifications, you can use it on two conditions: you acknowledge the source of your program and you can demonstrate that the program does what you need. As another example, you find a C++ program and modify it to suit your need, this is also acceptable provided that you acknowledge the source and demonstrate that your modified program does what you need. One last example, say if you find a Java simulation program and your friend also wants to use the program, it is perfectly acceptable for you to tell your friend where the program can be found. You can also pass the *original* program to your friend. However, if you have modified your program to do the project, you should **not** pass the modified program to your friend because it contains

your work. If you want to clarify any of these issues, please check with the Lecturer-in-charge.

Note that whether you are writing your own program or using a public domain program, you must **verify** that the program indeed simulates what is required. You will need to demonstrate in your report that your programmes are correct. If you intend to use test cases to prove that your programs are correct, you will need to derive your own test cases and ensure that they are sufficient to verify the correctness of your programs.

An additional requirement that we require is your simulation experiments be **reproducible**. Let us illustrate what we mean by an example. Assume that you perform 5 replications of a simulation and get the mean response time of, say, 5.1456, 5.1892, 5.7865, 4.1234 and 5.6789. We may ask you to reproduce some of these simulation in your interview and you should be able to show us that your simulations do indeed give these figures as the mean response time. This is perfectly doable since pseudo-random generators are in fact deterministic machines. Here are some hints on how you can make sure that your experiment is reproducible:

- If you use C and you use `srand(time(0))` to try to randomise the seed. In order to make your experiment reproducible, you should instead do

  ```
  seed = time(0);
  srand(seed);
  ```

  You should then make sure that you store or make a record of the seed that you have used. This will allow you to reproduce your experiment if you know the seed.

- For Matlab, you will need to save the states of the random number generators.

To sum up, project marks are allocated for (1) Demonstrating that your simulation programmes are correct; (2) Sound statistical analysis; (3) Reproducibility.

5 Scope and assessment criteria

This is an individual project. You are expected to complete this project on your own. However, this project does not preclude you from using any publicly available program to do your work, provided that you acknowledge the source. If you do not acknowledge your source, it is considered as **plagiarism**.

5.1 What and how to submit

You are required to submit

1. A written report detailing the work that you have done. Note the following:
   (a) Only soft copy is required.
   (b) It must be in Acrobat "pdf" format.

(c) It must be called "report.pdf".

2. The programs, scripts etc. that you use to do your work. **You need to make sure that you submit all your programs etc. since the assessment will be based on the programs that you have submitted.**

3. Any other supporting materials, e.g. a log created by your simulation, scripts that you have written to process the data, the seeds that you have used for your simulation etc.

It is important for you to refer to any of the programs, scripts, additional materials in your written report so that we are aware of them.

You can use zip, tar or rar to archive your report, programs and supporting materials into a single file called "project.zip", "project.tar" or "project.rar" depending on the utility that you have chosen to archive your work.

When you are ready to submit,

1. Log onto your CSE account and make sure that the current directory contains the file to be submitted.

2. Type 9334 at the bash prompt and then the following command:

   ```
   give cs9334 project project.tar
   ```

   if the file to be submitted is project.tar. Otherwise, replace it by name of your submission file. Please note that the system will only accept "project.zip", "project.tar" or "project.rar" as the filename for submission.

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 communications error and you will not be time to rectify that.

5.2 Assessment

Your assessment will be based on the written report. (However, you may be asked to attend an interview if required). Thus, it is important that you write a clear and to-the-point report. It is important that you realise 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, how you choose your simulation parameters and why you think they are good enough, how you design your simulation program etc.

Although the project has a stated goal, you are welcome to go beyond the project descriptions above and do some in depth study of the topic, we will encourage you to do that by offering bonus marks for innovative and creative investigations. For example, the project assumes that the pre-procesor randomly chooses a number of servers to process the sub-tasks.

You can attempt to come out with a more efficient mechanism.

If you do work beyond our expectations and your work is deserving, you may be awarded bonus mark for your project. With the bonus mark, it may mean that you score more than 20% (the nominal weighting for the project) for the project. The marks beyond 20% will be added towards your raw final mark for the entire course. This means that your raw final mark for the entire course can exceed 100% and if this happens, the maximum you will receive is 100%.

6 Plagiarism

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

In particular, if you use a program to do your work, do make sure that you acknowledge the source.

References

[1] Daniel A. Menasc. Response-Time Analysis of Composite Web Services. *IEEE Internet Computing*, 2004.