COMP9334 Project, Session 1, 2017: Getting the best response time out of your power bill

Version 1.0
Due Date: 11:00pm Sunday 21 May 2017.

This version: April 11, 2017

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

This project is inspired by the research work reported in the article *Optimal Power Allocation in Server Farms* [1]. Server farms are a common part of many corporations' computing infrastructure but they consume a lot of energy. The key question asked in this research is: Is it possible to reduce the energy consumption of server farms while maintaining good response time of the computing system. In this project, you will use simulation to try to answer a similar research question.

In this project, you will learn:

1. To apply discrete event simulation to a performance analysis problem
2. 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 you question may appear to be silly, the other students may very well have the same question!**

3 Background

A typical architecture of a server farm is shown in Figure 1. The server farm consists of a high-speed router at the front end and a number of computing servers at the back end. A job arriving at the server farm will first reach the high-speed router. The function of the router is to send the job to one of the servers in the back end. The job will then be processed by the server and once it is completed, the job leaves the server. Most servers use Processor Sharing (PS) rather than first-come-first-serve to process the jobs. The PS in Figure 1 indicates the servers are using PS for job processing.

A key performance issue in a server farm is its response time. In this project, we will assume that the high-speed router takes negligible time to distribute the jobs to the servers. Therefore, the response time of a job is due entirely to the processing in the server.

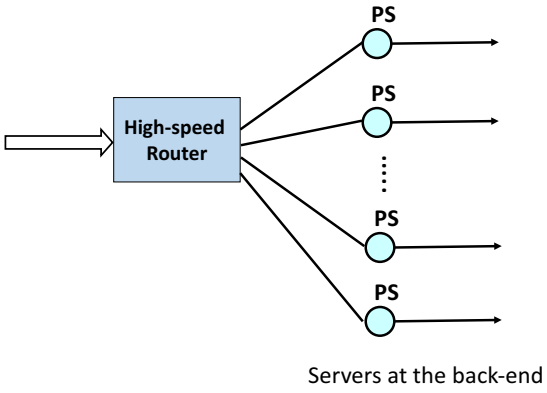


Figure 1: Typical architecture of a server farm. The front-end consists of a high-speed router and the back-end has a number of computer servers. Each server uses processor sharing (PS).

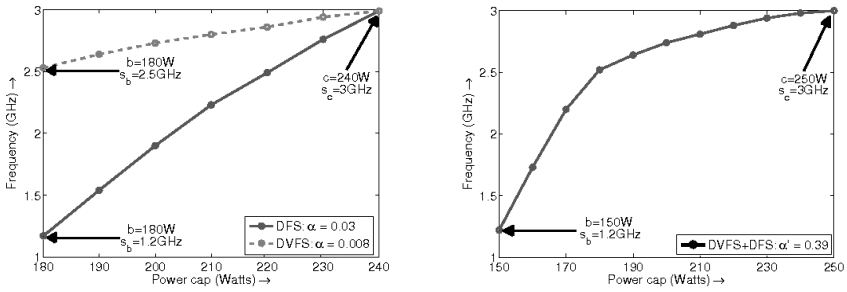
Energy cost in operating servers is high. Each server consumes typically hundreds of Watts of power. The article [1] stated that "*Server farms today consume more than 1.5% of the total electricity in the U.S. at a cost of nearly $4.5 billion.*" (Note: the article was published in 2009 so these are past figures.) Given the high operating cost of server farms, a question is whether it is possible to reduce the energy consumption while maintaining the same level of system performance in terms of response time.

Modern servers are equipped with power management technologies such as Dynamic Frequency Scaling (DFS) or Dynamic Voltage and Frequency Scaling (DVFS). These technologies allow a system administrator to operate the servers at different power levels. If the server is operating at a higher (resp. lower) power level, then it is operating at a higher (resp. lower) clock frequency or higher (resp. lower) computing speed. Figure 2 shows the relationship between clock frequency and power consumption for a particular type of server under a specific workload. You can clearly see that the clock frequency is an increasing function of the power consumption.

The power management technologies have provided us with a method to reduce the power consumption while maintaining the performance. You can approach this problem in two different ways. You can fix the performance (i.e. response time) and see how you can reduce power consumption. Alternatively, you can fix the power consumption and see how you can maximise the performance. We will take the latter approach in this project.

Let us consider an example which was partially described on Page 26 in Week 1's lecture. Let us assume that you have a power budget of 3000W. You can operate each server at 150W, 200W and 250W, which will give you a clock frequency of 1.2 GHz, 2.7 GHz and 3 GHz. (These clock frequencies correspond roughly to the curve in Fig. 2b.) You can use your 3000W power budget in a variety of configurations, for example:

- Operate 20 servers at 150W at clock frequency of 1.2 GHz (the slowest speed).
- Operate 15 servers at 200W at clock frequency of 2.7 GHz (the medium speed).
- Operate 12 servers at 250W at clock frequency of 3.0 GHz (the highest speed).



(a) DFS and DVFS

(b) DVFS+DFS

Power-to-frequency curves for DFS, DVFS, and DVFS+DFS for the CPU bound LINPACK workload. Fig.(a) illustrates our measurements for DFS and DVFS. In both these mechanisms, we see that the server frequency is linearly related to the power allocated to the server. Fig.(b) illustrates our measurements for DVFS+DFS, where the power-to-frequency curve is better approximated by a cubic relationship.

Figure 2: Figure illustrating DFS and DVFS. This figure is taken from [1].

A performance analysis question is which one of these configurations will give you the minimum response time for your workload. This is the question that you will be considering in this project. Note that for simplicity, in this project, we will limit ourselves to configurations where all servers are operating at the same clock frequency.

We will now describe the project in greater details.

4 Project description

In this project, you will develop simulation models to decide how to allocate power to servers to minimise the response time.

4.1 Description of workload and server farm

In a real design problem, you will need to collect data from the server to find out what the inter-arrival time distribution and the service time distribution are. In this project, you will be given these distributions but in real-life, you will need to fit a probability distribution against the data that you have collected.

In this simulation, you will assume that:

1. The server farm has 10 servers.
2. Each server uses Processor Sharing (PS) to process the requests.
3. The power budget is 2000 Watts. This means that if you switch on s servers (and leave the other $(10 - s)$ servers off), then each running server is operating at a power level of $\frac{2000}{s}$ Watts.
4. If a server is running at a power level of p Watts, then its clock frequency f (measured in GHz) is a function of p and is given by

$$f = 1.25 + 0.31(\frac{p}{200} - 1) \quad (1)$$

For example, if 8 servers are switched on, then each server operates at a power level of 250 Watts and its clock frequency f is $1.25 + 0.31(\frac{250}{200} - 1) = 1.3275$ GHz.

Note: For simplicity, we assume in this project that the servers can operate at any power level but in reality, a server can only operate at a discrete number of power levels.

5. We use $\{a_1, a_2, \ldots, a_k, \ldots, ...\}$ 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 $a_k = a_{1k}a_{2k}$ $\forall k = 1, 2, ...$

   (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 $[0.75, 1.17]$.

   Note: The easiest way to generate the inter-arrival times is to multiply 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.

6. The high-speed router distributes the requests using round robin job assignment. For example, assuming s servers that are switched on and the servers that are on are Server 1, Server 2, ..., Server s. The round robin job assignment means tha Request 1 is assigned to Server 1, Request 2 to Server 2, ..., Request s to Server s, Request $(s + 1)$ to Server 1, Request $(s + 2)$ to Server 2, ..., Request $2s$ to Server s, Request $(2s + 1)$ to Server 1, and so on. You can assume the time to distribute a request to a server is negligible.

7. 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 $\alpha_1 = 0.6$, $\alpha_2 = 8$, $\beta = 0.86$, and

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

   Note that t is the service time when the request is processed by a server operating at 1 GHz. You can assume that if the same job is to be processed by a server operating at f GHz, then the service time required is $\frac{t}{f}$. For example, if a job requires a service time of 8 time units when processed by a server operating at 1 GHz, then the same job will take $\frac{8}{1.86}$ time units to complete when it is processed by a server operating at 1.86 GHz.

4.2 Project goal and requirements

Your goal is to determine the number of servers s you should operate in order to minimise the mean response time of the requests. In order to achieve this goal, some of the tasks that you need to do are:

1. Computer programs or routines that generate the required probability distributions. You need to provide evidence that you have done this correctly.

2. A computer program that can simulate a PS server. The simulation program should take the number of operating servers s as an input variable. By using this program, you can test the effect of different values of s on the mean response time of the sever farm. This will allow you to find suitable value(s) of s.

3. You need to provide evidence that you have verified the correctness of the simulation code. We will not provide you with test cases so you will need to come out with the test cases yourselves.

4. You need to explain how you choose the simulation parameters. Examples of simulation parameters are how long you should run the simulation for, how many replications you should do, where the transient ends etc.

5. You will need to use statistically sound method to compare two systems. You will need to justify why you think the value(s) of s that you have chosen is (are) good.

6. You need to ensure that your simulation results are reproducible.

4.3 Language and use of public domain programs

This project requires you to write some computer programs. You can do this project using any language (C, C++, Java, Python etc.) or any mathematical package (Matlab, Scilab, Octave etc.) you prefer. You may also use discrete simulation packages such as OMNet++ 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 a PS server, 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 to simulate PS server 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 classmate also wants to use the program, it is perfectly acceptable for you to tell your classmate where the program can be found. You can also pass the *original* program to your classmate. However, if you have modified your program to do the project, you should **not** pass the modified program to your classmate 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.

4.4 Reproducibility

We require that your simulation experiments are **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 results 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 setting of the random number generator. This is discussed in Week 6's lecture.

4.5 Computational resources

In case you need computing resources to do simulation, the CSE server `williams.cse.unsw.edu.au` can be used for simulation. See `http://taggi.cse.unsw.edu.au/FAQ/CSE_Server_Information/#williams` for details.

5 Hints on doing simulation

There are three hints on simulation.

The first hint is that you can do this project by simulating only one of the s servers. This can be done by using the arrival times of the Requests 1, (s+1), (2s+1), (3s+1) etc because each server gets one out of s requests. In general, you can choose a random integer k (where $1 \leq k \leq s$) and use Requests k, (s+k), (2s+k), (3s+k) etc.

The second hint is that the minimum response time will not occur at $s = 1$ nor $s = 2$. You do not have to simulate these two values of s. You will need to simulate s from 3 to 10 inclusively.

The third hint is on how to simulate a PS server and you can find it in Appendix A.

6 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**.

6.1 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 that you use to do your work.
3. Any supporting materials, e.g. a log created by your simulation, scripts that you have written to process the data 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 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.

6.2 Assessment

Your assessment will be based on the written report and if required, a demonstration/interview. Thus, it is important that you write a clear and to-the-point report. It is important that you 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, how you choose your simulation parameters and why you think they are good enough, how you design your simulation program, how you carry out the statistical analysis, 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. 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%.

7 Plagiarism

You should be aware of the UNSW policy on plagiarism. Please refer to the course web page for details. See also Section 4.3 on the use of public domain software.

References

[1] A. Gandhi, M. Harchol-Balter, R. Das and C. Lefurgy. "Optimal power allocation in server farms", ACM Sigmetrics 2009. `http://dl.acm.org/citation.cfm?id=1555349.1555368`.

A Simulating a PS server

The PS server was discussed in Week 5's lecture. 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, the number jobs in a PS server remains the same. The discrete event simulation advances from an event to the next one.

The key data structure that you need to maintain is the list of jobs in the server. Each job is characterised by two attributes: the time the job arrives at the server and the remaining amount of service the jobs will still need. Each time when a job arrives or departs, this data structure should be updated. We will use an example to explain this. For this example, we consider a PS server with the following job arrival times and service times.

Arrival time	Service time
1	2.1
2	3.3
3	1.1
5	0.5
15	1.7

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:

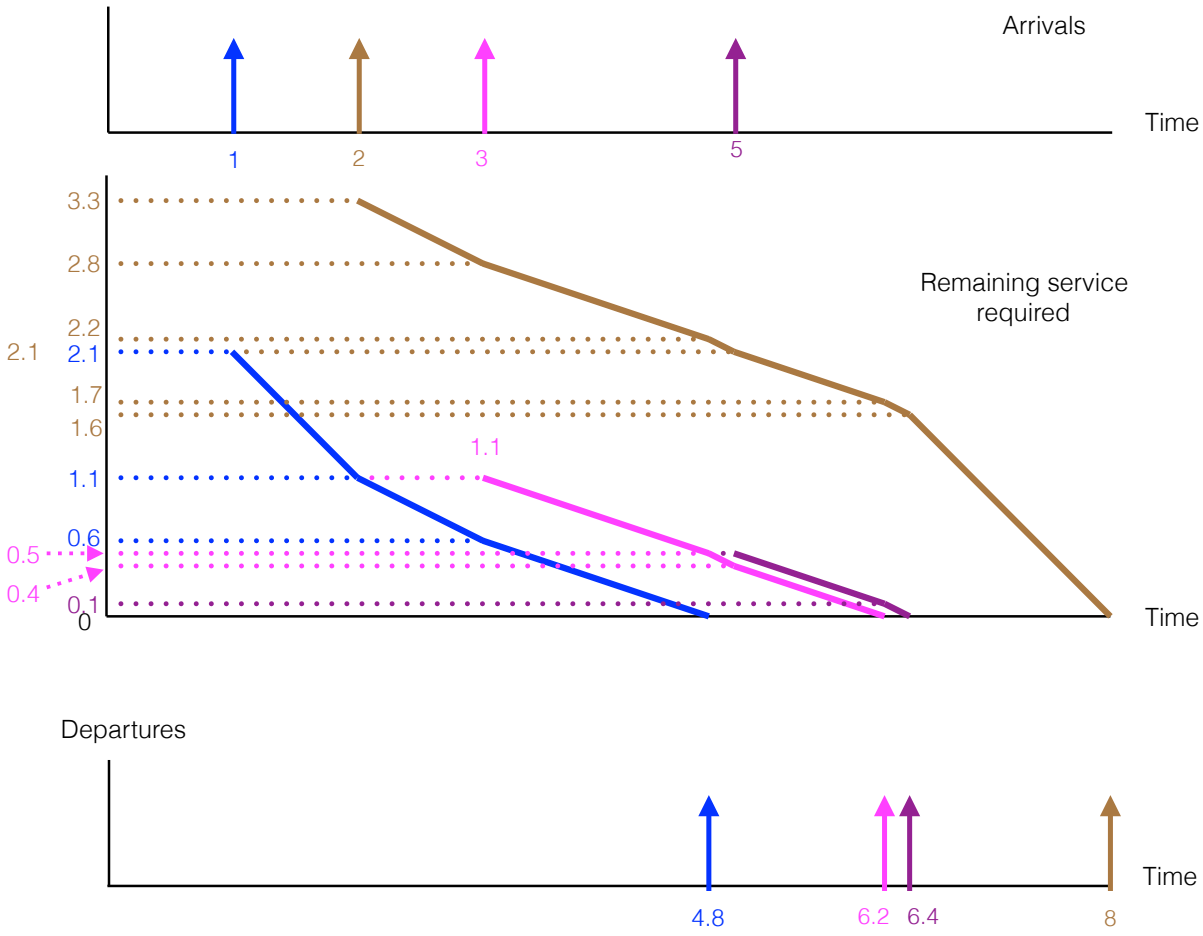
- **Next arrival time** is the time of the next arrival
- **Next departure time assuming no more arrivals** is defined as the time of the next departure assuming that no more arrivals will come in the future. For simplicity, we will simply use the phrase next departure time. For example, if there are 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.

The "on-paper simulation" is shown in Table 1. 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. 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 curve. You will see that the slope stays constant in between two events and changes each time an event occurs.

Master clock	Event type	Next arrival time	Next departure time	Job list	Notes
$t = 0$	–	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)	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	(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	(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)	Since the event is a departure, you will need to update the (i) Job list; and (ii) Next departure time. You should also update the response time and the number of jobs completed; the updating of these two quantities is done in the same way as first-come-first-serve.
$t = 5$	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 1: Table illustrating the updates in PS


Arrivals
Time
1
2
3
5
3.3
2.8
2.2
2.1
2.1
1.7
1.6
1.1
0.6
0.5
0.4
0.1
0
Remaining service
required
1.1
Time
Departures
Time
4.8
6.2
6.4
8