COMP28112 – Lab exercise 3
'The Quest for Performance'

Introduction

When engineering a new distributed system, it is often the case that one needs to build a model of the system first in order to guarantee a certain level of performance. For example, when designing the server for exercise 2, we had to assess various performance related aspects, such as response time to the multiple requests during a typical lab session, or capacity requirements, such as the disk space needed to maintain a log file. Assuming, in the latter case, that the average size of the logs generated during one lab session is 10MB, it wouldn't be a good idea to use a disk with only 10MB free space: after every lab session we would have to clear the logs!

It is true that the server for exercise 2 does not have to cope with particularly high loads. The messages are short, the requests do not require lots of computational time, and the number of users is not that high. However, in a more demanding, real-world scenario, the server side will face higher loads; still the system designers need to ensure that the overall performance will meet a certain level. To do this, the designers will have to make several decisions: how many servers shall be used, what bandwidth must be available and so on.

These are typical capacity planning decisions. According to one definition, "*capacity planning is the process of predicting when future load levels will saturate the system and of determining the most cost-effective way of delaying system saturation as much as possible*" [1].

By definition, capacity planning implies the existence of some ability to assess the performance of a system under different circumstances. Since such an assessment often needs to take place before the system has been built (it would be painful to find out about the failure of a system after it has been built! In addition, an extensive range of assessments might be too time-consuming to carry out in a real system.) the only way to perform the assessment is to use an abstract model of the system. With this model, one can make use of either *mathematical analysis* or *computer simulation*[1] to obtain an answer to a range of questions. With mathematical analysis, one expects to derive closed formulae answering these questions. With computer simulation one develops a computer program that is trying to evaluate the state of a system (often the question is how the state varies over time, in which case the simulation may need to consider discrete time intervals) using a large number of different input scenarios (which may be randomly generated).

[1] http://en.wikipedia.org/wiki/Computer_simulation

The Exercise

In this exercise, you are going to use computer simulation to evaluate the performance of the server-side of a distributed system under different circumstances.

1. In this system, requests keep arriving from the clients; these requests involve some computation at the server (we don't care what is computed). The average time between the arrival of two successive requests is an input parameter (expressed in time units, which is the unit of time used in the simulation – see below).

2. When these requests arrive, they are initially stored in a queue. The maximum queue size is an input parameter. Any requests that arrive when the queue is full are rejected.

3. A number of servers (or CPUs, if you like) may fetch requests from the queue for execution. Only one server can fetch requests at a time. Clearly, a single server cannot execute multiple requests at the same time.

4. The time needed by the server to complete every request is also an input parameter (expressed again in time units).

Taking as an input the four parameters described above, you are going to write a program to simulate the system and show, over a period of 1,000,000 time units:

- The percentage of requests that was rejected.
- The average queue size (that is, how many requests were in the queue on average) during the simulated period of time.
- The average response time of the system to each request that was processed during this period of time (the response time is computed as the difference between the time the request has completed its execution on one of the servers and the time the request arrived to the queue).

Thus, the time period you are going to simulate is 1,000,000 time units. Remember that during one time unit, only one new request may arrive to the system and only one request can be assigned from the queue to a server for execution. If, during one time unit, the queue is empty and a new request arrives, this request can be assigned to an idle server (if there is one).

Hint

The key to write the simulation is to model what happens during each time unit and keep track of how what happens affects the values of all the parameters you need to compute. For example, if at some point in time a new request arrives, you need to check if there is space in the queue, record its arrival, update the queue size, etc… After this check, you need also to check if there is an idle server and assign a request for execution… Throughout these checks, you need of course to update all the information used to produce the statistics you need to generate at the end of the simulation!

You can use any programming language you like. You will need to use an appropriate library function/method in this language to produce random numbers to simulate the arrival of requests over time. It is suggested that this is your first step and you start by generating those (random) points in time when a new request arrives (based on the value of the arrival rate – your first input parameter) – you may want to check the notes from the relevant lecture too.

Regarding the input parameters of the program (which might be entered at command line or read from a file – whatever you prefer – but they should not be hard-coded), you can assume that the maximum size of the queue may be any number between 10 and 10000, the number of servers may be any number between 1 and 50, and the execution time of a request may be any number between 1 and 100 time units. Apparently, the mean time between successive arrivals may be any number between 1 (that is, requests arrive all the time) and 1000000 (the latter meaning that, on average, one request is expected throughout the simulation).

If you run on a linux machine the executable

`/opt/info/courses/COMP28112/ex3/lab3` (or `$COMP28112/ex3/lab3`)

with command line parameters

```
queue_size arrival_rate number_of_servers execution_time
```

(e.g., `lab3 10000 3 2 25`) you will get answers which are similar to what you should be getting with your program (due to the randomness introduced in the program, there will be small variations between different runs with the same parameters; in practice, for each set of parameters, one would expect that multiple runs would take place to obtain a better idea about the behaviour of the system).

Marking Scheme

Correct simulation of time	1
Correct simulation of random arrivals	2
Correct simulation and computation of queue size	2
Correct simulation of all the rest	2
Correct computation of output statistics	2
Quick completion of each simulation run (no more than 5 sec)	1

NB: You should put your solution in a file called `solution3` in a directory called `ex3` in your `COMP28112` directory.

References

[1] D. A. Menascé, V. A. F. Almeida. Capacity Planning for Web Performance: Metrics, Models & Methods. 1998. Prentice-Hall.