

KTH
2G1126 Distributed Computer Systems

Lab1 report

Server Waiting Times
- a simulation

2003-04-06

By:
Christoffer Cederwall, cegc@kth.se, 800705-0332
Olof Paulsen, op@kth.se, 780304-0497
Peter Stolt, d99-pst@nada.kth.se, 790523-2117
Andreas Öhrvall, d99-aoh@nada.kth.se, 800710-6274

Group 4

Table of contents

1. INTRODUCTION	3
2. MODIFICATION PROCESS	3
3. THE EXPERIMENT	4
4. RESULTS	4
5. ANALYSIS	4
6. WHO DID WHAT	5
7. WHAT HAVE WE LEARNED.....	5
8. POSSIBLE IMPROVEMENTS.....	5

1. Introduction

We were given a simulator that simulated the arrivals and the related departures of requests to a web server. The time of these events were calculated statistically using mathematical formulas. The task was to re-write the simulation tool so that it instead read the given log file from a more or less real web server and use that data instead of the calculated times in the first version.

In this log, a time stamp was given with the information of when the request arrived. In addition to that we found the size of the file that was requested. The idea was to assume that the time taken for the web server to deliver that file was somehow related to the size of the request at hand.

Two formulas for calculating this work time for the server were also provided. The idea was that a parameter indicates if it is a high performance architecture based system, or if the server operates on less powerful components. Then, by running the simulator with these different formulas, the workload of the server can be simulated relative to each other and then compare if the better computer performs better in from of waiting times etcetera for the arriving requests. The final idea could then be if it is worth investing in the higher performance server or not.

2. Modification process

The first part of the lab was to modify the given code for the simulator. Since the program worked for the existing M/M/1 model, we basically only needed to change the calculations of the estimated arrivals to what we found in the text file (the log from the web server) and then calculate the time for that server to complete the task of sending the requested data.

The first major thing was to read the file, which was done simply by a file pointer in c and then operating on it with fscanf commands. The other important stage then was to re-dimension the queue of arriving events so that it also stored the departure times which we calculated by the two different formulas depending on if it was the high performance server or the old fashion machine operating on the data.

Finally, some manipulation of this data prints the results to the screen. Here the main issue was to keep track of the units on each variable, so that the unix time stamps in the log was used in the correct way.

3. The experiment

The huge log file contained a lot of log entries. We used this in order to simulate two different scenarios on each of the two different web servers (i.e. different parameters on the calculation stage explained above). First we ran the first five thousand lines as input for the simulation. Then we fed the simulator with the next five thousand lines (i.e. lines 5000-10000) instead in order to see if the result was affected by the change of input. We also performed these simulations twice on each dataset.

In order to investigate the two different web servers, we simply had a variable called “level” in the program and if that was set to 1 we used the first version and if it was 2 the other choice was used. The formula to simulate these behaviours was ‘service time = 0.1 + level * (size in kilobytes)’. The level was either 0.6 or 0.3 respectively.

4. Results

When we ran the program twice (or more) with the same settings, we got the same results every time. Gathering the output data to a table gives this result:

	Slow server		Fast server	
	0-5000	5000-10000	0-5000	5000-10000
Average delay in queue	0.173	0.276	0.036	0.042
Average number in queue	2.665	9.833	0.551	1.499
Server Utilization	0.226	0.365	0.113	0.189
Time simulation ended	325.109	140.218	325.109	140.213
Max que length	85	200	53	88
Max delay in que	3.855	4.792	1.747	2.276
Number of custs waited over 0.5	1293	1863	570	870

5. Analysis

First of all, the fact that the result was the same when the simulator simulated with the same settings was expected. Since we use the same data as input and also have a fixed formula for calculating what to do, the outcome is defined on beforehand. Since the data could be generated randomly, or as in our case from real events which could be considered to be stochastic, the simulation itself is indirectly stochastic.

In both cases when comparing the two data sets, the results differ from each other, but it differs a lot more in the slower server. That indicates that the 5000 lines might not be big enough a dataset in order to use as an asymptotic reference. In the slower version the average delay in the queue is twice as long in the second input of data and

the average number in the queue is roughly three times as high. The maximum queue length etcetera is also higher with the second data set.

When turning to the 0.3 parameter (the faster web server), we see that these differences still exist, but they are smaller. That indicates that when having more powerful equipment, the data is processed fast enough in order to lower the number of requests in the queue in the more demanding case.

Comparing the two servers, we see that the performance is increased dramatically in the more demanding dataset and it is also better in the first 5000 loglines. It is also better in the first five thousand lines, but the relative difference is not as high.

Summarizing these observations, we realize that investing money on the faster server will generate better performance and might be money well spent if you know you will have a lot of demanding requests to the server. Of course it is better and faster in every situation, but could be “unnecessarily” powerful if you don’t expect such a demanding activity on the web server.

6. Who did what

Finally we document what we did and how we did it. First we sat down all four and planned the work and roughly deciding what we needed to modify and what we wanted to analyze. Then, Andreas and Peter fixed the modification and Christoffer and Olof went about analyzing what was given in the simulator.

Then we sketched an outline together on what we wanted to include in this report and then we circulated the report, like the token in a token ring, giving everybody a chance to write and modify the report before handing it in to the course staff.

7. What have we learned

This small program has given us an insight on how to make a (M/M/1) simulation in practice. The homework gave life to the theory we have studied and motivates us to continue study the theory of simulations since we now feel that it can be useful in practice. The switch to a trace-driven simulation was an interesting approach of the assignment. We had to understand every line of the code in order to adjust it, but we didn't have to spend many days on developing the code. The time felt very well spent.

8. Possible improvements

Our general opinion of the homework is that it was well prepared. The only thing we feel was a little inconvenient was the fact that the small log file and the big log file were so differently formatted. It would have been easier if there was a link to both the long and short log file and they could also have had exactly the same format. Both files should have had “unix-time” instead of formatted time.