Performance Engineering

We are going to upgrade a software system that executes web requests from users from the Internet.

We have observed the running system for 2 hours. During these two hours, we have seen that 6000 execution requests have been completed. Actually, we have seen that requests arrived at a rate of 50 requests per minute; and therefore we can deduce that all requests that arrived were completed.

The system is composed of six service centers: A *webServer,* a *UserAppServer*, a *UserLogServer*, an *AdminAppServer,* an *AdminLogServer,* and a *Database.* We are interested in knowing the service time S_k of each service center, but we could not directly measure it. However, we have been able to measure the following:

- Each user's request executes once in the *WebServer*. There is only one resource for the *WebServer.* We have seen that the *WebServer* has been busy 1504.8 seconds. After the WebServer executes, a request can follow two different paths, depending on whether the requester was a normal user or an administrator.

- If the requester was a normal user, which happened 90% of times, the request executed in the *UserAppServer*, iterating 4 times its execution in this server. Each time that a request executed each iteration in the *UserAppServer*, it also left a log trace by executing in the *UserLogServer* (therefore, the number of visits to the *UserLogServer* is the same as to the *UserAppServer*). There are two resources for executing the *UserAppServer* and one resource for the *UserLogServer.* The Utilization of the *UserAppServer* is 29.88%, and the Utilization of the *UserLogServer* is 29.88% too.

- If the requester was an administrator, which happened the 10% of times, the request executed in the *AdminAppServer,* iterating 2 times. Each time that a request executed each iteration in the *AdminAppServer*executed, it also left a log trace by executing in the *AdminLogServer* (therefore, the number of visits to the *AdminLogServer* is the same as to the *AdminAppServer*). There is a single resource for executing the *AdminAppServer,* which was busy during 360 seconds, and a single resource for the *AdminLogServer* which was busy during 120 seconds.

- Finally, each request (regardless it was an administrator or a normal user request) needed to execute some calls to a *Database*. There is a single resource for executing the *Database,* and we measured that it was busy for 900 seconds. We could not measure the number of times each request executed in the *Database*. However, we were able to measure that there were, in average, 0.14325 jobs in the *Database*; and that, for each execution of the *Database*, the time between the job arrived and the execution was completed (including the time that it spent waiting for service in a queue) was 0.0573 seconds.

You have to submit a PDF document with your answers to the following 3 exercises:

A) Use the operational laws to calculate the Service Time S_K of each of the six service centers.

B) We would like to upgrade our previous system by adding a *SecurityCheck* service, which will execute for all requests, its Service time will be 300 milliseconds, and will execute in a single resource. Model the upgraded System using Queueing Networks (In JMT or in your preferred Queueing Network simulation engine). Add screenshots of your model (including screenshots of the values of service times S_k that you calculated in the previous exercise, number of resources, arrival rates, and routing probabilities). Simulate the model to calculate the System Response Time.

Hint1: In the cases that, from a service center (e.g., WebServer) a job can go to more than one service center, use Probabilistic Routing.

Hint2: When a request iterates X times in a group of service centers, you can model it adding an additional loopback arc (see slide 34 of the first Performance lesson), having the loopback arc probability (X-1)/X and the arc that leaves the service center probability 1/X.

Hint3: Use the exponential distribution for all times and rates (frequencies) you need to model.