Simulation Analysis of a Web Application

Simulation Program Design (Classes, Program Logic - events, event handlers,etc) due:
March 25th, 2018, in class. <u>Submit a slide presentation, not a report.</u>

<u>Simulation Program Functional Demo: April 4th.</u>

- ***Submit all your main code and scripts, and input files if any.***
- ***Code should be well-commented***
- ***Code should produce a clean, somewhat readable trace.***
- ***In your demo you will have to point out (reason-through) a few examples (from trace output) of why you think the code is correct***
- ***You need not show metrics calculation at this point.***

Final Submission (Code, Analysis in slides format) due: Sunday, April 28th, 2018, 11:59pm
Presentations: April 29th- 1st May, 2018

- ***Submit a file called assmt2_final_name1_name2.tar. Tarball should have:***
  - *All code (please delete junk files), scripts, all input files used.*
  - *Slides which show the graphs, observations and conclusions*

<u>Assignment is to be done by TWO team members.</u>

Overall Goal of the Assignment

In this assignment you will continue the study of the performance of a Web Application, this time through a discrete event simulation model. You will take advantage of the power of simulation analysis and model behaviours that are difficult to capture using theoretical queuing systems and Markov chain models. You will write a program, and then run it to ask various interesting questions. For all metrics you will perform proper statistical soundness analysis.

Detailed Specifications

Write a simulation program in any general purpose simulation language (except C. It is not a good habit in today's times to program in C - please use every opportunity to program in object oriented languages. C++, Java or python is fine.)

System Characteristics

Your system should have the characteristics of the Web server environment that you measured. The following is recommended, you can add/modify.

- Multi-core server machine
    - *Assume thread-to-core affinity. Once a thread is "assigned" to a core, it will remain there*
- Multi-threaded Web server
- Thread-per task model - until max is reached, after which queuing for threads starts.
- Round-robin scheduling, with context-switching overhead
- Request time-outs
- Users are in a standard closed loop - issue request, wait for response, think then issue request again.
    - *Think time should have a mode not close to zero - don't assume exponential think time*
- Have options of various request execution time distributions, such as - constant, uniform, exponential.
- Request timeout should also have a well-chosen distribution
    - *Have some sort of a minimum, and then a variable component*

Performance Metrics

Metrics/graphs will be similar to what you measured for assignment

- Average Response Time vs number of users
    - Generate confidence intervals for this metric. Point estimates (averages of estimates from independent runs) are ok for the remaining metrics
- Throughput, Goodput, Badput vs Number of users
    - Badput = rate of completion of requests which were already timed out.
    - Goodput = rate of completion of requests that were not timed out
    - Throughput = goodput + badput
- Request Drop rate vs number of users
- Average core utilization vs Number of users
- Some additional graphs representing your own curiosity regarding system performance vs some system parameter

Ensure that the transient is determined and discarded in each of your simulation runs. You can do this informally, or follow Welch's procedure. .

Self-Proposed DES Project

- ***Sunday March 17th, 11:59pm: Draft Proposal submission***
- ***Tuesday March 18th: Proposal finalization after feedback/discussion***
- ***March 25th, 2018, in class: design plan***
- ***<u>Simulation Program Functional Demo: April 8th,in class.</u>***
- ***Final submission: Sunday, April 28th, 2018, 11:59pm***
- ***Project Presentations: April 29th- 1st May, 2018***