CS-350 - Fundamentals of Computing Systems
Homework Assignment #1 - EVAL

Due on September 19, 2024 — Late deadline: September 21, 2024 EoD at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

EVAL Problem 1

In this EVAL problem, you will evaluate the result of the clock measuring functions developed in the corresponding BUILD assignment. In particular, we will estimate which method (SLEEP vs. BUSYWAIT) works best for stable CPU speed estimates, and use the power of clock measurements to understand a bit how the SCC works under the hood.

a) First, estimate the CPU clock speed on your own machine using the SLEEP method. To do this, collect the results returned by the `clock` executable you implemented by running it 10 times. Start with a wait time of 1 second, all the way up to (and including) 10 seconds, increasing the wait time by 1 second at each step.

   From the 10 CPU clock speed measurements you have obtained, compute and report the max, min, average, and standard deviation.

b) Repeat the same procedure above where instead of the SLEEP method, you use the BUSYWAIT method for all the measurements. Then, compute and report the same statistics on the measured CPU clock speed.

c) Now let us this capability to answer the following question: is the SCC comprised of identical machines? In other words, does the SCC always allocate you a physical machine with the same type of CPUs? To answer this question, conduct 10 clock frequency measurements of 10 seconds each, but start each experiment in a different SCC session, with some time elapsing in between. For instance, conduct the first 10-seconds measurement in a SCC session started on Tuesday morning, the second in the afternoon, etc. Each time, make sure to start a new SCC session. In your report, produce a table with one column being the date/time of the experiment and resulting measured clock speed. Next, draw your conclusions about how the SCC works under the hood in terms of machine provisioning.

EVAL Problem 2

In this EVAL problem, we will start to examine the behavior of the `server` process in response to variable traffic conditions generated by the `client`.
HINT: many of the questions below, will ask you to determine a lapse of time in which the server was active. You can compute that time window in post-processing as the lapse of time between the `<receipt timestamp>` of the very first request and the `<completion timestamp>` of the last request processed by the server.

a) First, launch the client with the following parameters: `client -a 10 -s 12 -n 500 <some port>` where `<some port>` is the same port (of your choice) used for the `server`. Allow the client and server to exchange data and terminate, then post-process the output produced by the server.

   From the server output, compute the average throughput of the server in terms of requests per second. Walk us through the steps you performed to accomplish this.

b) Follow the same procedure above and once again post-process the output of the server. This time, compute the utilization of the server as a percentage. Walk us through the steps you performed to accomplish this.

c) Repeat the experiment above but by running the server and client back-to-back, while at the same time extracting some statistics about the server run from the OS. To do so, move to the build directory and use the following command line:

   ```
   /usr/bin/time -v ./server 2222 & ./client -a 10 -s 12 -n 500 2222
   ```

   NOTE: by doing so, BOTH client and server will run in the same terminal and their output will be interleaved on the console. If you want to suppress the output of the client, redirect the stdout of the client to the null device using:

   ```
   /usr/bin/time -v ./server 2222 & ./client -a 10 -s 12 -n 500 2222 > /dev/null
   ```

   Also note that the server will run in background. You can use the following command to kill it if it does not terminate:

   ```
   killall server
   ```

   After the server has run successfully, the `time` utility will print a host of statistics acquired from the OS. Take a look at the entry "Percent of CPU this job got:". Does it match with what you computed in the previous question?

d) Now let us repeat the computation of the utilization of the server as in Qb) but this time sweep through the `-a` parameter passed to the client. In particular, run the first experiment for a value of 1; then a second time with a value of 2; and so on until and including the case where the value is 12. Thus, you will run 12 experiments in total.

   Produce a plot that depicts the trend of the resulting server utilization (y-axis) as a function of the arrival rate (`-a`) parameter (x-axis). Is there any correlation between the two values?

e) By reusing the same exact output files produced as part of the previous question, let us reason about the response time of the various requests. First, compute the average response time for all the 500 requests as observed in the case when the parameter `-a` was set to 10. Also compute max, min, and std. deviation.

f) Now, repeat the average response time calculation individually for the various runs with the `-a` parameter from 1 to 12. Finally, produce a plot that depicts the trend of the average response time y-axis as a function of the server utilization x-axis. What relationship do you discover between the two quantities? Are they directly or inversely proportional to each other? Is the relationship linear?