CS-350 - Fundamentals of Computing Systems
Homework Assignment #4 - EVAL

Due on October 10, 2024 — Late deadline: October 11, 2024 EoD at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

EVAL Problem 1

In this EVAL problem, we will study the behavior of the system when multiple processors (a.k.a. threads in this case) are available to process the incoming workload. For this part, make sure to have at least $w+2$ processors available in your system, where w is the number of worker threads that we will start. The extra two threads are needed for the client and the parent thread, respectively.

a) First thing first, determine the maximum number of CPUs that you can use for this EVAL problem. If your machine has $w+2 = 4$ CPUs, you should consider $w = 2$, and so on. For this part, simply explain how you recovered that information for your machine. If you are using SCC, you can select required number of CPUs. However, still verify that you indeed received it, explain how you did it here.

b) Starting easy with $w = 2$, run the following command:

```
./server_multi -q 1000 -w 2 2222 & ./client -a 37 -s 20 -n 1500 -d 0 2222
```

Now, measure the utilization of each individual worker thread. To do that, first classify the printouts by the thread ID (`T0`, `T1`, ...) and then re-use the same approach to compute their utilization as in HW1, but independently for each thread.

What do you notice about the utilizations of the two threads? Can you conclude that the load is balanced between them?

c) Now run a similar experiment as above by considering the following template command:

```
./server_multi -q 1000 -w <workers> 2222 & ./client -a 37 -s 20 -n 1500 -d 0 2222
```

where `<workers>` is first set to 4, then to 6, and finally to 8. Note, to run the command with `-w 8`, you need at least $w + 2 = 10$ CPUs on your machine. If you do not have that many CPUs, use SCC or post-process the output produced by CodeBuddy as these are exactly the parameters used by CodeBuddy to run your code on the first 3 test cases of `server_multi`.

Produce a plot for the average response time of any request (y-axis)—thus, without differentiating between threads—as a function of the number of threads (value of the `-w` parameter) from 2 to 8 (in increments of 2). Is the improvement in response time linear or super-linear as the number of threads increases?

d) Last but not least, let us study what happens to the rejection rate as more workers become available. Run two cases and compare the output.

First, re-run or reuse the output from HW3 for the equivalent command:

```
./server_multi -w 1 -q 10 2222 & ./client -a 18 -s 20 -n 1500 -d 0 2222
```

Then, run and post-process the output of command:

```
./server_multi -w 2 -q 10 2222 & ./client -a 18 -s 20 -n 1500 -d 0 2222
```

Now look at the rejection rate and reason about the following: *if X is the rejection rate with 1 worker, then X/W is the rejection rate with W workers.* Is this true or not? Motivate your answer. You are not oblibged, but welcome to run additional relevant experiments to further sustain your conclusion.