CS-350 - Fundamentals of Computing Systems
Homework Assignment #5 - EVAL

Due on October 30, 2023 — Late deadline: November 1, 2023 EoD at 11:59 pm at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

EVAL Problem 1

In this EVAL problem, we will study the behavior of a system with a configurable FIFO vs. SJN queue dispatch (a.k.a. scheduling) policy. The goal of this EVAL is to demonstrate that despite no changes are made to the workload, some key system metrics might change if the system is able to meaningfully reorder the pending workload for processing.

a) First thing first, let us understand the overhead of using SJN as the queue scheduling policy. Measuring overheads is always hard, but luckily for you when things are implemented on a real machine like these assignments, measuring the true machine behavior becomes possible!

We are going to measure overheads in two situations: (1) at low utilization, and (2) at high utilization. In both cases, you will run the commands provided below 10 times and extract the average server runtime. We will then compare the average obtained under FIFO vs. those obtained under SJN.

(1) Low Utilization. In the low-utilization test, run 10 times the following command for the FIFO experiment:

```
/usr/bin/time -v ./server_pol -w 2 -q 100 -p FIFO 2222 & ./client -a 10 -s 20 -n 1500 2222
```

As you can see the `/usr/bin/time` utility is prefixed to the server to obtain the final runtime statistics. You should parse the 10 results from this command that are found in the "Elapsed (wall clock) time" statistic reported by the `/usr/bin/time` utility at the end of the run, and compute an average of these values.

Now, repeat the same thing by replacing `-p FIFO` with `-p SJN` in the command above. Collect the new 10 values of wall-clock times and compute the average. Finally, compare the average runtime computed for FIFO vs. SJN.

(2) High Utilization. In the high-utilization test, we follow an identical procedure as above, with the only difference that we change the amount of workload sent by the client. The new command template will be:

```
/usr/bin/time -v ./server_pol -w 2 -q 100 -p <policy> 2222 & ./client -a 40 -s 20 -n 1500 2222
```

where `<policy>` should be set to FIFO in the first 10 runs, and to SJN in the last 10 runs.

Compare the average runtime under FIFO and SJN at low and high utilization, reasoning on the extra per-request time spent by the server under the two policies.

Solution.

For the **low utilization** scenario:

FIFO Average:

$$\begin{aligned}\text{Average}_{\text{FIFO, Low}} &= \frac{150.76 + 150.76 + 150.74 + 150.76 + 150.74}{5} \\ &= 150.752 \text{ seconds}\end{aligned}$$

SJN Average:

$$\begin{aligned}\text{Average}_{\text{SJN, Low}} &= \frac{150.74 + 150.75 + 150.75 + 150.77 + 150.77}{5} \\ &= 150.756 \text{ seconds}\end{aligned}$$

For the **high utilization** scenario:

FIFO Average:

$$\begin{aligned}\text{Average}_{\text{FIFO, High}} &= \frac{37.96 + 37.97 + 37.96 + 37.96 + 37.96}{5} \\ &= 37.962 \text{ seconds}\end{aligned}$$

SJN Average:

$$\begin{aligned}\text{Average}_{\text{SJN, High}} &= \frac{38.00 + 38.00 + 37.97 + 38.00 + 37.97}{5} \\ &= 37.988 \text{ seconds}\end{aligned}$$

At **low utilization**, the average runtimes for both FIFO and SJN are similar, with FIFO being 150.752 seconds and SJN being 150.756 seconds. This subtle difference suggests that, at low server utilization, the scheduling policy (whether FIFO or SJN) doesn't significantly influence the runtime.

However, at **high utilization**, there is a slight increase in the average runtime for SJN (37.988 seconds) compared to FIFO (37.962 seconds). The increased time under SJN at high utilization might be attributed to the overhead associated with constantly determining the shortest job next, whereas, in FIFO, tasks are simply processed in the order they arrive without additional decision-making.

In conclusion, while the differences are minuscule and might be negligible for many practical applications, the extra per-request time spent by the server under SJN at high utilization suggests that FIFO might be slightly more efficient under heavy loads, due to its straightforward task scheduling.

b) Okay, we have understood the cost of SJN with the measurements above. Now let us quantify the benefits. In a way similar to what we did in HW1, we can measure the average response time as a function of the server utilization. Recall that the utilization is computed as the sum of the length of all the processed requests over the total server runtime. With only one worker, that should be a number less than or equal to 1, with two workers it should be less than or equal to 2, and so on.

For this experiment, we will always set `-w 2` to spawn two workers. Next, do a series of experiments where you run the following template command 10 times:

```
./server_pol -w 2 -q 100 -p FIFO 2222 & ./client -a <arr_rate> -s 20 -n 1500 2222
```

Where the parameter `<arr_rate>` goes from a value of 22 (first run) to a value of 40 (10th run), thus increasing by 2 at every run. Use these experiments to construct a curve of the request response time as a function of the server utilization.

Now, do 10 more runs with the same strategy as described above, but where the server policy is set to SJN, thus the new command template will be:

```
./server_pol -w 2 -q 100 -p SJN 2222 & ./client -a <arr_rate> -s 20 -n 1500 2222
```

Plot the two curves (one from the FIFO experiments, and the other from the SJN experiments) in the same plot and try to quantify which queueing policy works better and by how much.

Solution. the SJN curve being consistently below the FIFO curve indicates that SJN is more efficient, resulting in shorter average response times. This efficiency stems from SJN's strategy of prioritizing shorter jobs, thereby minimizing wait times for a large number of requests. In contrast, FIFO processes jobs in their arrival order, leading to potential inefficiencies when short jobs are queued behind longer ones. Thus, the plot suggests that, across various server utilizations, SJN offers better performance in terms of average response time compared to FIFO.

Problem 1 continued on next page...

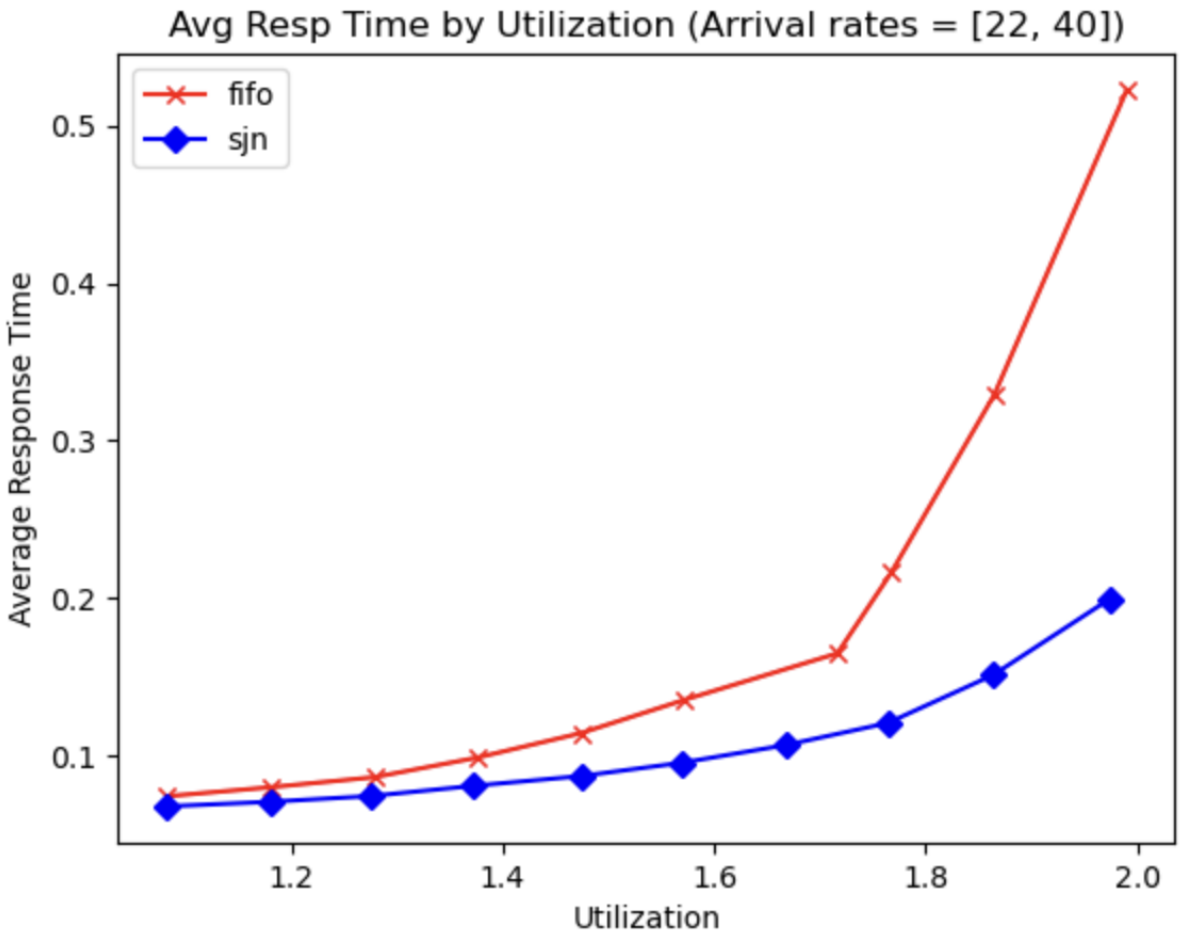

Figure 1: 2: Avg Resp Time by Utilization

Problem 1 continued on next page...

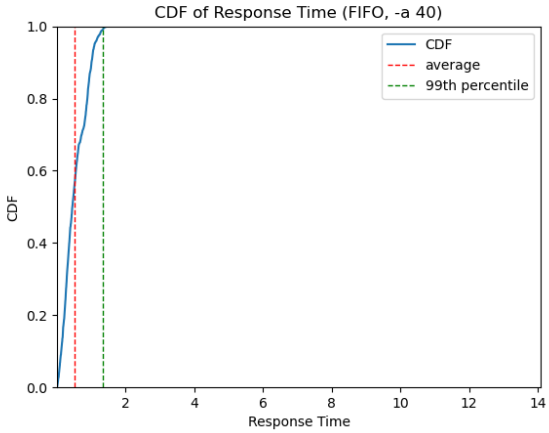

Figure 2: 3: CDF of Response Time (FIFO, -a 40)

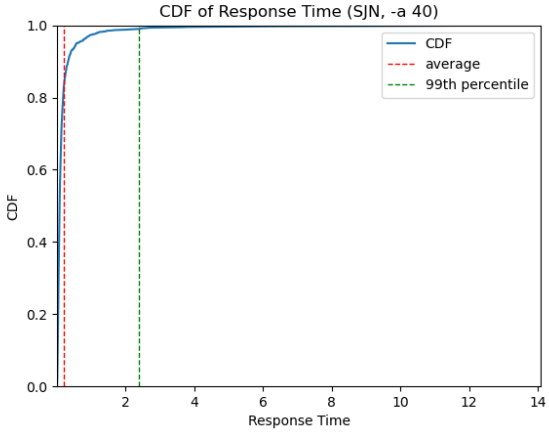

Figure 3: 3: CDF of Response Time (SJN, -a 40)

c) In the engineering world, we say that there is no free lunch. If one of the two policies works better than the other, where is the catch? We are set to discover that with the following experiment. The response time average only tells part of the story, so let us study the full distribution of response times.

Focus on the outputs produced in the previous experiment by the 10th run (the one where you set `-a 40`) of each policy (FIFO and SJN). Produce TWO separate plots, each depicting the CDF of the response times observed in the two runs under analysis. Make sure that in both plots, the range of the x and y axis are the same, so that they are visually comparable.

In the CDF plots, add two vertical bars: (1) one marking the value of the average response time, and (2) another marking the 99th percentile response time value.

d) Now focus on the differences that you see between the two CDF plots. Reason on the spread of the response time distribution and on the overall predictability of the system when evaluated by looking response time metric. *HINT:* see possible definitions of predictability in the CS350 book.

Finally, provide a motivated answer to the question: *which queuing policy leads to a more predictable system behavior?*

Solution.

FIFO displays an average response time situated in the middle of its CDF, indicating a balanced spread of response times around this average. Its 99th percentile being less than 2 suggests most jobs complete in less than 2 units of time.

In contrast, SJN has its average near 0, revealing that a majority of jobs have very low response times. This is expected given SJN's priority for shorter jobs. However, the 99th percentile being more than 2 indicates that while many jobs complete swiftly, some longer jobs take significantly more time, creating a long tail in the distribution.

FIFO provides more consistent response times since jobs are processed in the order they arrive. However, there's potential for long jobs to delay subsequent ones. SJN, while boasting improved average response times, poses unpredictability for longer jobs. These jobs, given the lowest priority, can endure extensive waiting if shorter jobs continuously arrive.

Therefore, while SJN may outperform FIFO in terms of average response times, FIFO offers a more predictable system behavior due to its consistent processing of jobs in their order of arrival.