Scheduling: The Multi-Level Feedback Queue

Annalise Tarhan

January 29, 2021

1 Run a few randomly-generated problems with just two jobs and two queues; compute the MLFQ execution trace for each. Make your life easier by limiting the length of each job and turning off I/Os.

```
Job 0: runTime 33
Job 1: runTime 17

T0 - J0 starts at priority 1
T10 - J1 starts at priority 1
T20 - J0 starts at priority 0
T30 - J1 starts at priority 0
T37 - J1 finishes
T37 - J0 starts at priority 0
T47 - J0 starts at priority 0
T50 - J0 finishes
```

```
Job 0: runTime 27
Job 1: runTime 21

T0 - J0 starts at priority 1
T10 - J1 starts at priority 1
T20 - J0 starts at priority 0
T30 - J1 starts at priority 0
T40 - J0 starts at priority 0
T47 - J0 finishes
T47 - J1 starts at priority 0
T48 - J1 finishes
```

2 How would you run the scheduler to reproduce each of the examples in this chapter?

Figure 8.2: -n 3 -q 10 -l 0,200,0
Figure 8.3: -n 3 -q 10 -l 0,180,0:100,20,0
Figure 8.4: -n 3 -q 10 -l 0,175,0:50,25,1
Figure 8.5 Left: -n 3 -q 10 -i 1 -l 0,101,0:100,50,1:100,50,1 -S -c
Figure 8.5 Right: I don't understand this one. Why does the first job's first block cause its priority to drop, but its first block after the second two jobs arrive doesn't? Also, the text says the priority is boosted after 50 ms, but the illustrated example doesn't show it being boosted until 100 ms.
Figure 8.6 Left: -n 3 -q 1 -a 10 -i 1 -l 0,170,0:80,90,9 -S -c
Figure 8.6 Right: -n 3 -q 1 -a 10 -i 1 -l 0,170,0:80,90,9 -c
Figure 8.7: -Q 10,20,40 -a 2 -l 0,140,0:0,140,0 -c

3 How would you configure the scheduler parameters to behave just like a round-robin scheduler?

Set the number of queues equal to one.

4 Craft a workload with two jobs and scheduler parameters so that one job takes advantage of the older Rules 4a and 4b to game the scheduler and obtain 99% of the CPU over a particular time interval.

-n 2 -q 100 -i 1 -S -l 0,1000,99:0,1000,0 -c

The two jobs each run roughly equally at first while they both still have priority 1, but after each of their first blocks, the first job monopolizes the CPU until it is finished running.

5 Given a system with a quantum length of 10 ms in its highest queue, how often would you have to boost jobs back to the highest priority level in order to guarantee that single long-running job gets at least 5% of the CPU?

In order for the job's single 10 ms block in the highest queue to be enough to guarantee 5% of the CPU, it would need one at least once every 200 ms, so it must be boosted at least that often.

6 One question that arises in scheduling is which end of a queue to add a job that just finished I/O; the -I flag changes this behavior for this scheduling simulator. Play around with some workloads and see if you can see the effect of this flag.

Using the -I flag causes a job to restart immediately after its I/O call completes, interrupting whichever job is running. Its cumulative effect is typically to increase the average turnaround time, but there are exceptions.