Engineering &
Computer Science
Syracuse University

CIS 657 – Principles of Operating Systems

Topic: Process – Policies
(Scheduling – MLFQ)

Endadul Hoque

Acknowledgement

- Youjip Won (Hanyang University)
- OSTEP book – by Remzi and Andrea Arpaci-Dusseau (University of Wisconsin)

Revisit: Process Scheduling

- Workload assumptions:
  1. ~~Each job runs for the **same amount of time.**~~
  2. ~~All jobs **arrive** at the same time.~~
  3. ~~All jobs only use the **CPU** (i.e., they perform no I/O).~~
  4. The **run-time** of each job is known.

Revisit: Process Scheduling

- Workload assumptions:
  1. ~~Each job runs for the **same amount of time.**~~
  2. ~~All jobs **arrive** at the same time.~~
  3. ~~All jobs only use the **CPU** (i.e., they perform no I/O).~~
  4. ~~The **run-time** of each job is known.~~

Let's relax assumption 4:
Scheduler has **no prior knowledge** on the run-time of each job

How to schedule without perfect knowledge?

Multi-Level Feedback Queue (MLFQ)

- A Scheduler that learns from the past to predict the future.
- Two-fold objective:
  - Optimize **turnaround time** → Run shorter jobs first
  - Minimize **response time** without *a priori knowledge of job length*.

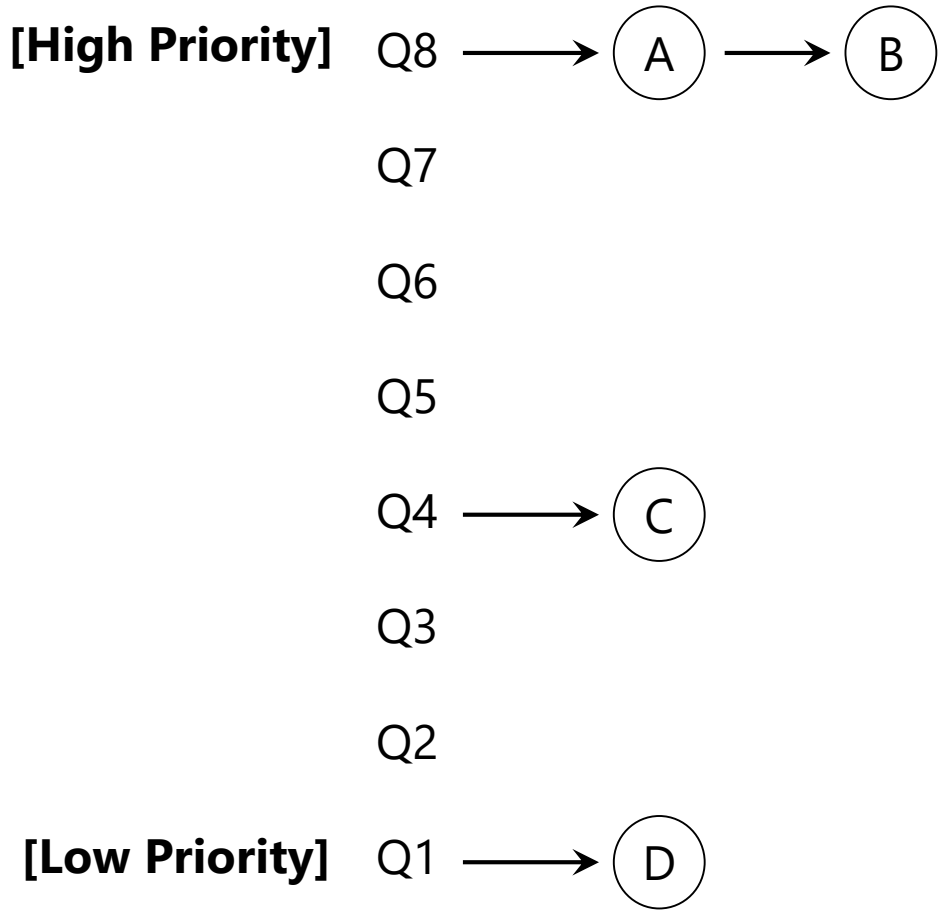
MLFQ: Basic Rules

- MLFQ has a number of distinct **queues**.
  - Each queues is assigned a different priority level.

- A job that is ready to run is on a single queue.
  - A job **on a higher queue** is chosen to run.
  - Use round-robin scheduling among jobs in the same queue

Rule 1: If Priority(A) > Priority(B), A runs (B doesn't).
Rule 2: If Priority(A) = Priority(B), A & B run in RR.

MLFQ: Example



MLFQ: Job Priority

- MLFQ **varies** the priority of a job based on its observed behavior.
- Example:
  - A job repeatedly relinquishes the CPU while waiting for I/Os → **Keep** its *priority high*
  - A job uses the CPU intensively for long periods of time → **Reduce** its *priority*.

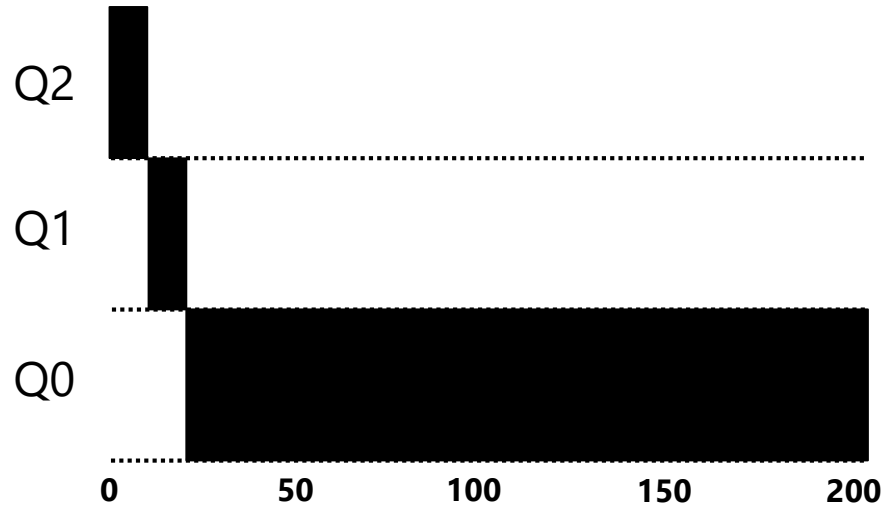
MLFQ: How to Change Priority

- MLFQ priority adjustment algorithm:
  - **Rule 3**: When a job enters the system, it is **placed** at the highest priority
  - **Rule 4a**: If a job uses up an entire time slice while running, its priority is **reduced** (i.e., it moves down on queue).
  - **Rule 4b**: If a job gives up the CPU before the time slice is up, it **stays** at the **same** priority level

In this manner, MLFQ approximates **SJF**

Example 1: A Single Long-Running Job

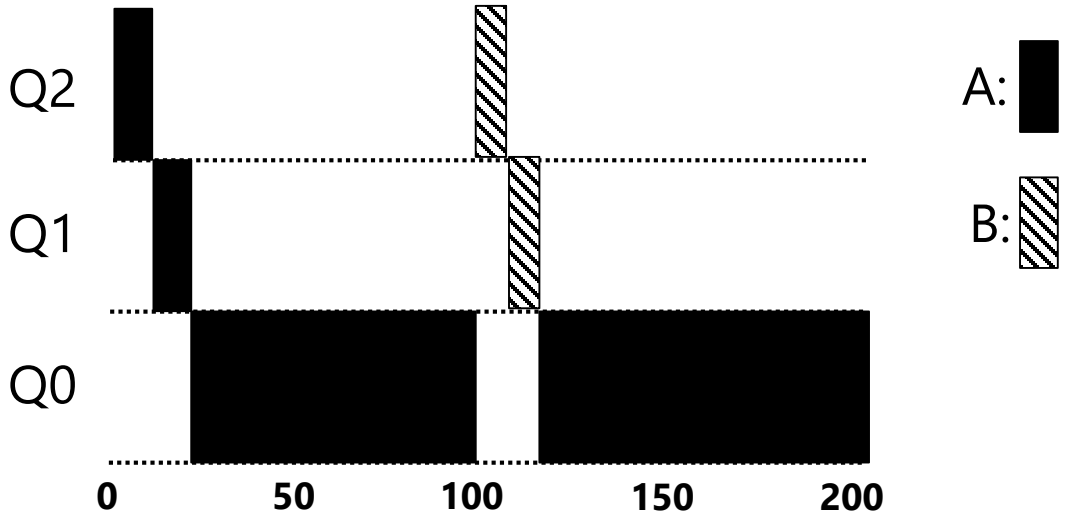
- A three-queue scheduler with time slice 10ms



Long-running Job Over Time (msec)

Example 2: Along Came a Short Job

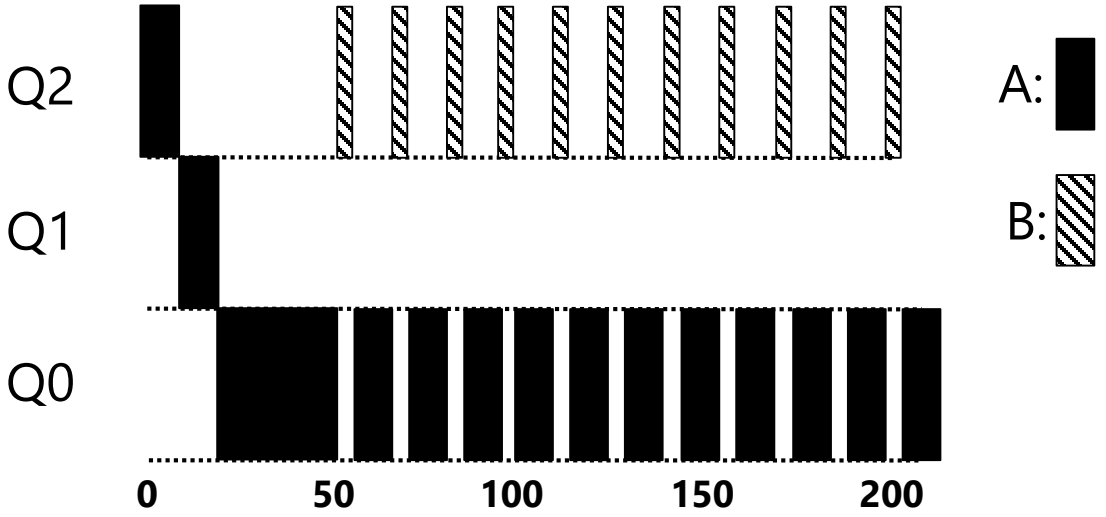
- Assumption:
  - **Job A**: A long-running CPU-intensive job
  - **Job B**: A short-running interactive job (20ms runtime)
  - **A** has been running for some time, and then **B** arrives at time T=100.



Along Came An Interactive Job (msec)

Example 3: What About I/O?

- Assumption:
  - **Job A**: A long-running CPU-intensive job
  - **Job B**: An interactive job that need the CPU only for 1ms before performing an I/O



A Mixed I/O-intensive and CPU-intensive Workload (msec)

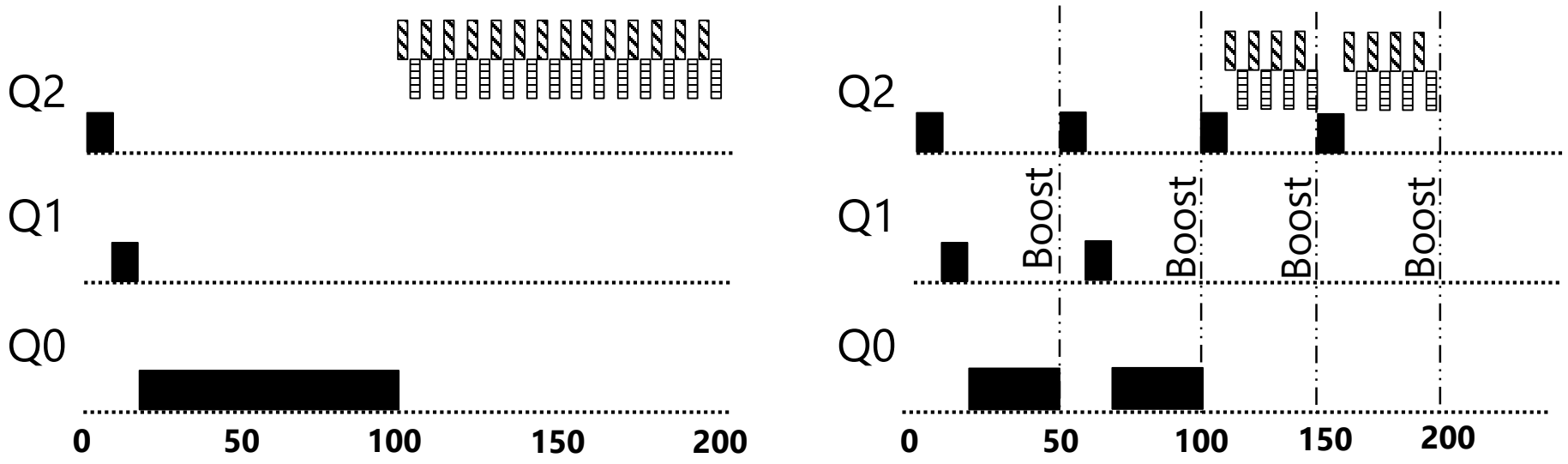
MLFQ keeps an interactive job at the highest priority

Problems with the Basic MLFQ

- Starvation
  - If there are “too many” interactive jobs in the system.
  - Long-running jobs will never receive any CPU time.
- Game the scheduler
  - After running 99% of a time slice, issue an I/O operation.
  - The job gains a higher percentage of CPU time.
- A program may change its behavior over time.
  - CPU bound process → I/O bound process

The Priority Boost

- **Rule 5:** After some time period **S**, move all the jobs in the system to the top-most queue.
  - Example: a long-running job(A) with two short-running interactive jobs (B, C)

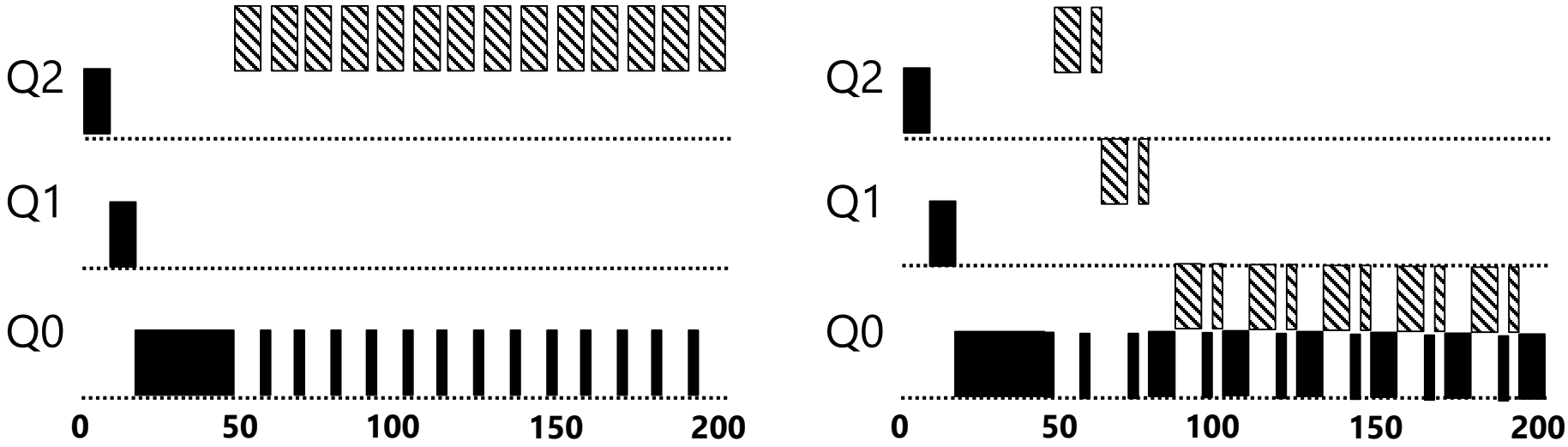


Without(Left) and With(Right) Priority Boost

A: B: C:

How to prevent gaming of our scheduler?

- Better Accounting
- **Rule 4** (Rewrite Rules 4a and 4b): Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), **its priority is reduced**(i.e., it moves down on queue).



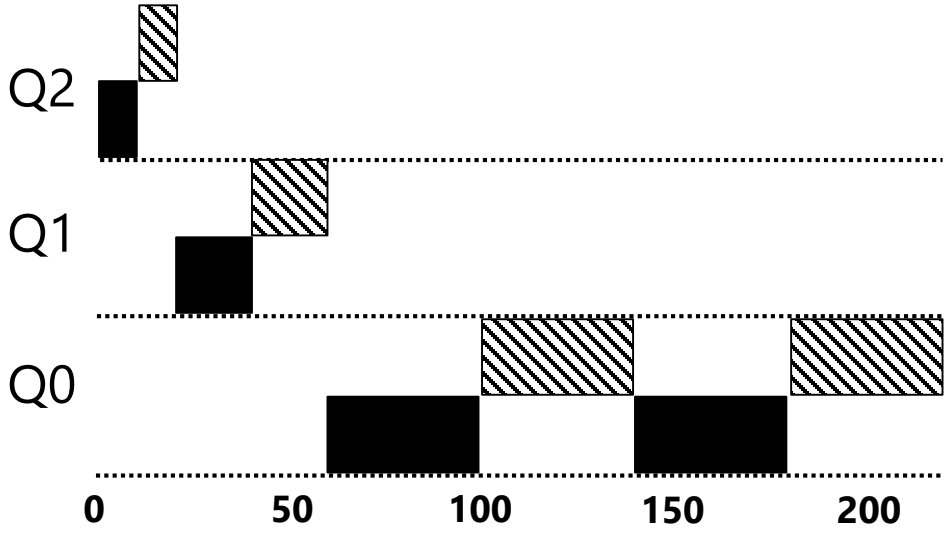
Without(Left) and With(Right) Gaming Tolerance

Tuning MLFQ And Other Issues

Lower Priority, Longer Quanta

- The high-priority queues → Short time slices
  - E.g., 10 or fewer milliseconds
- The Low-priority queue → Longer time slices
  - E.g., 100 milliseconds

Example: 10ms for the highest queue, 20ms for the middle, 40ms for the lowest



MLFQ: Summary

- The refined set of MLFQ rules:
  - **Rule 1:** If Priority(A) > Priority(B), A runs (B doesn't).
  - **Rule 2:** If Priority(A) = Priority(B), A & B run in RR.
  - **Rule 3:** When a job enters the system, it is placed at the highest priority.
  - **Rule 4:** Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced(i.e., it moves down on queue).
  - **Rule 5:** After some time period S, move all the jobs in the system to the topmost queue.

Reading Material

- **Chapter 8** of OSTEP book – by Remzi and Andrea Arpaci-Dusseau (University of Wisconsin) http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-mlfq.pdf

Questions?