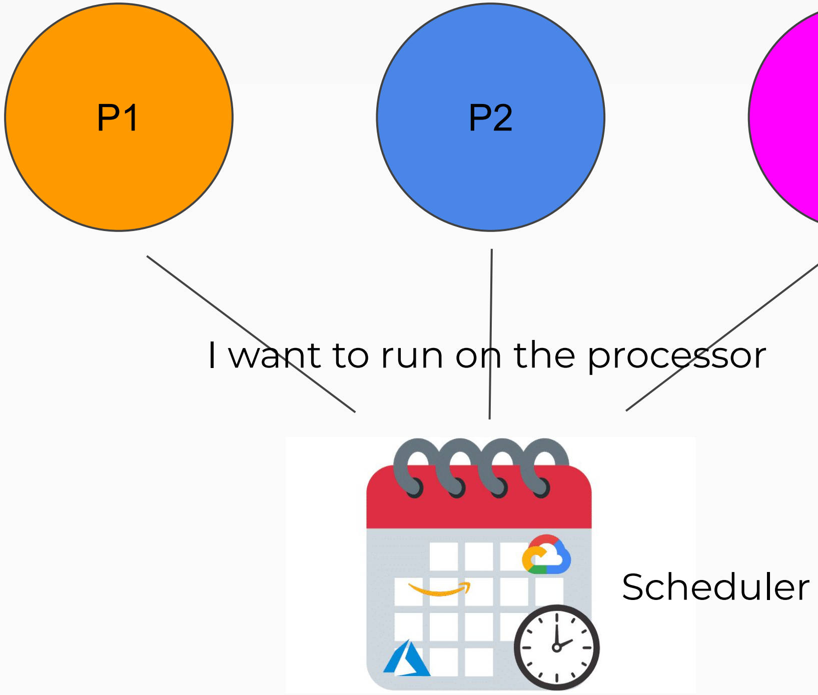
CS 330 - Operating Systems

Scheduling

14-08-2025

Scheduling

What is a Scheduler?



Scheduling Policies

- How should we develop a basic framework for thinking about scheduling policies?
- What are the key assumptions about the **workload**?
- What **metrics** are important?
- What basic approaches have been used in the earliest of computer systems?

Scheduling Parameters

- When the process is created?
- Time-taken for the job to complete
- Interrupting a process
- Does the process use CPU and also do I/O operations?

Designing a Scheduling Policy

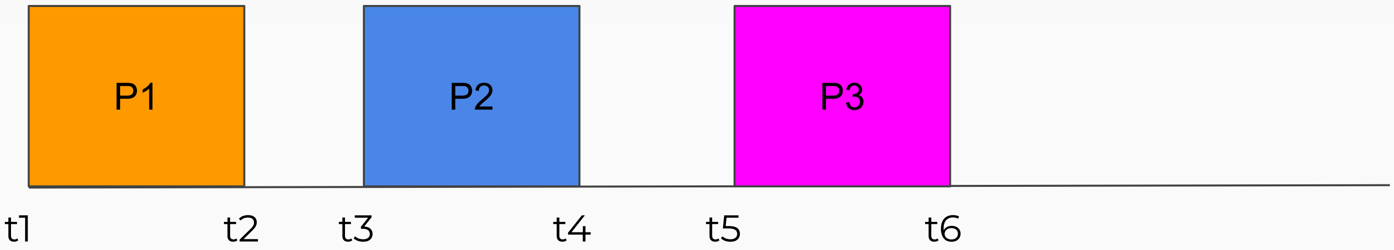
- Parameters
- Scheduling metrics

Designing a Scheduling Policy

- Parameters
- Scheduling metrics
    - Performance
    - Fairness

Scheduling Metrics

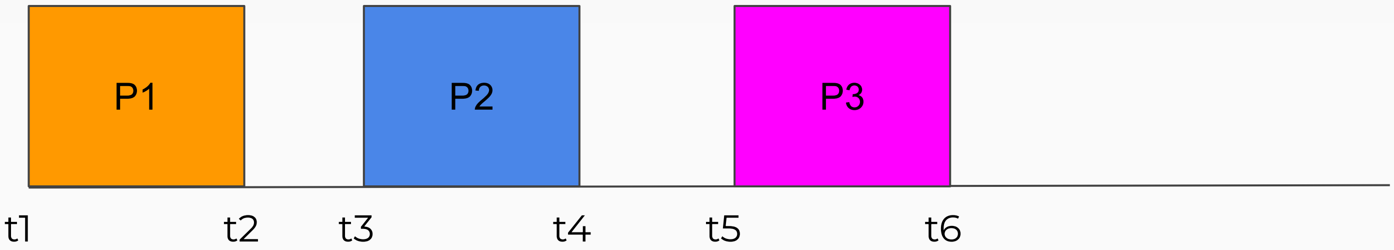
Turnaround time



Turnaround-time(p) = Completion-time(p) - Arrival-time(p)

Scheduling Metrics

Turnaround time



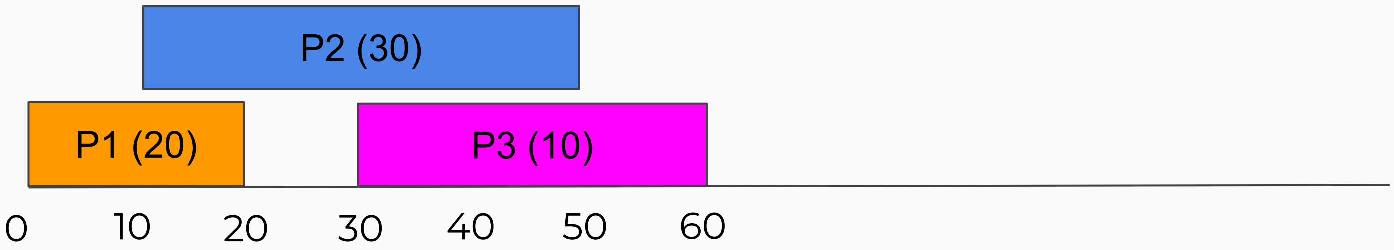
Turnaround-time(P1) = t2 - t1

Turnaround-time(P2) = t4 - t3

Scheduling Disciplines - FIFO

Scheduling Disciplines - FIFO

First-in First-out or FCFS



Arrival-time(P1) = 0

Arrival-time(P2) = 10

Arrival-time(P3) = 30

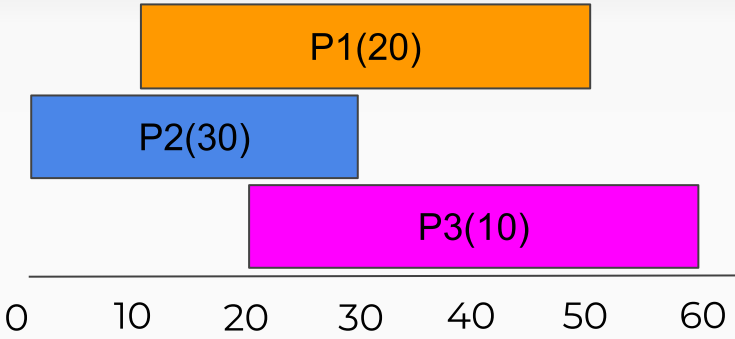
Completion-time(P1) = 20

Completion-time(P2) = 50

Completion-time(P3) = 60

Scheduling Disciplines - FIFO

First-in First-out or FCFS



Arrival-time(P1) = 10

Arrival-time(P2) = 0

Arrival-time(P3) = 20

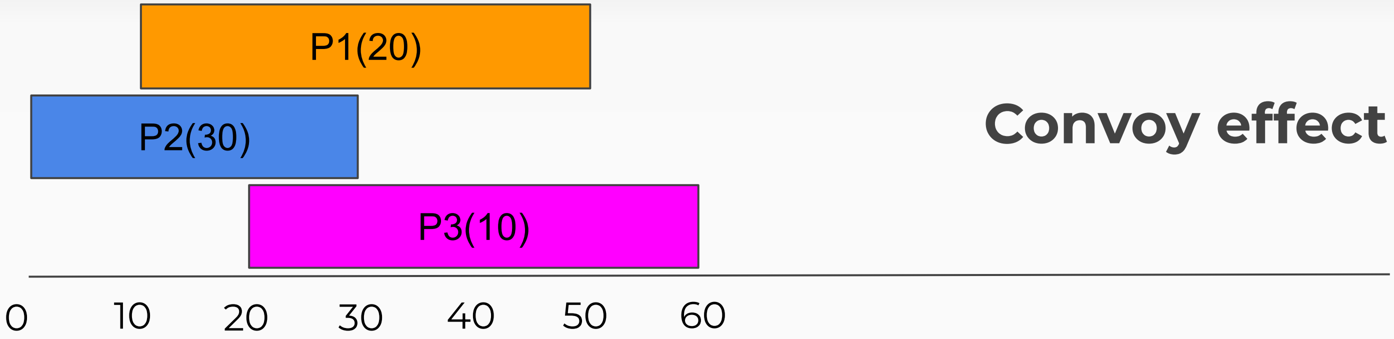
Completion-time(P1) = 50

Completion-time(P2) = 30

Completion-time(P3) = 60

Scheduling Disciplines - FIFO

First-in First-out or FCFS



Convoy effect

Arrival-time(P1) = 10

Arrival-time(P2) = 0

Arrival-time(P3) = 20

Completion-time(P1) = 50

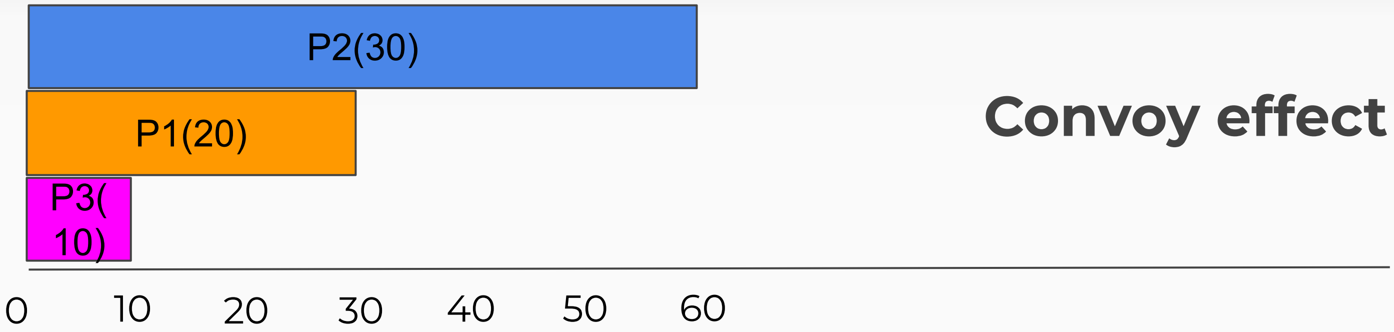
Completion-time(P2) = 30

Completion-time(P3) = 60

Scheduling Disciplines - SJF

Scheduling Disciplines - SJF

Shortest Job First



Convoy effect

Arrival-time(P1) = 0

Arrival-time(P2) = 0

Arrival-time(P3) = 0

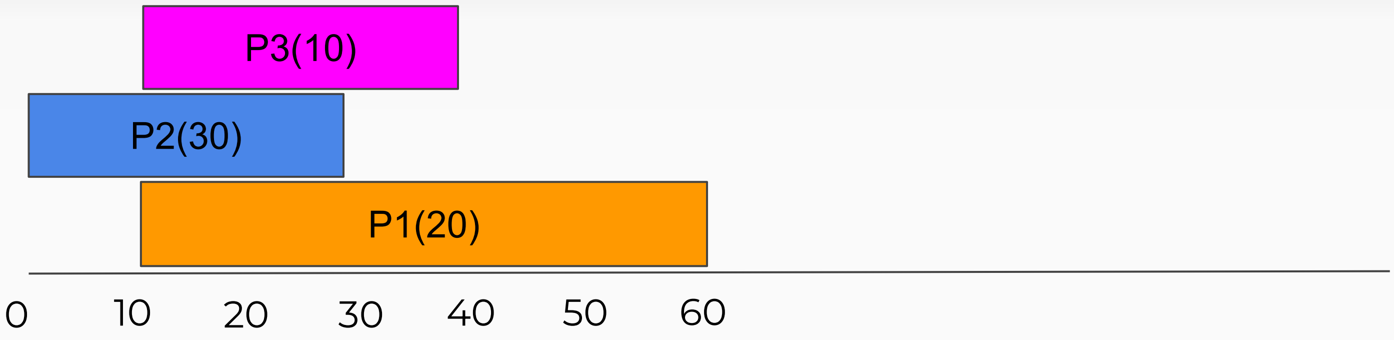
Completion-time(P1) = 30

Completion-time(P2) = 60

Completion-time(P3) = 10

Scheduling Disciplines - SJF

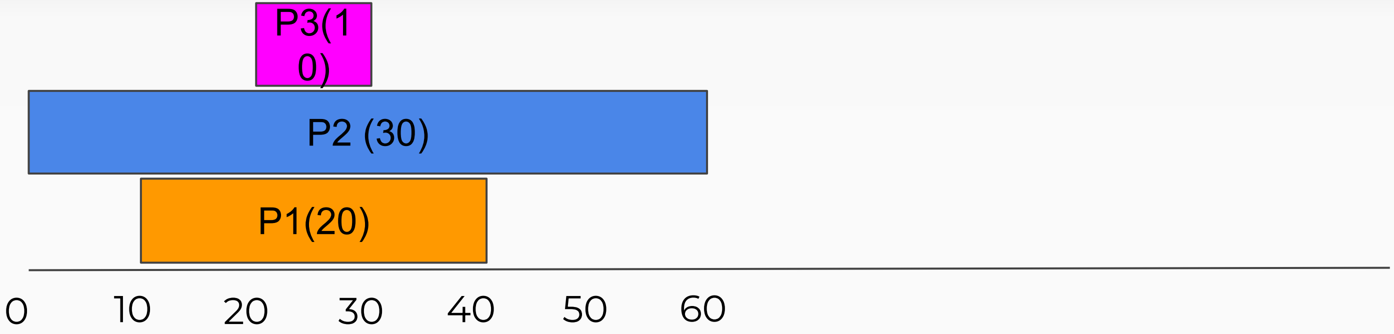
Shortest Job First



Arrival-time(P1) = 10	Completion-time(P1) = 60
Arrival-time(P2) = 0	Completion-time(P2) = 30
Arrival-time(P3) = 10	Completion-time(P3) = 40

Scheduling Disciplines - STCF / SRTF

Shortest Time-to-Completion (Remaining Time) First



Arrival-time(P1) = 10

Arrival-time(P2) = 0

Arrival-time(P3) = 20

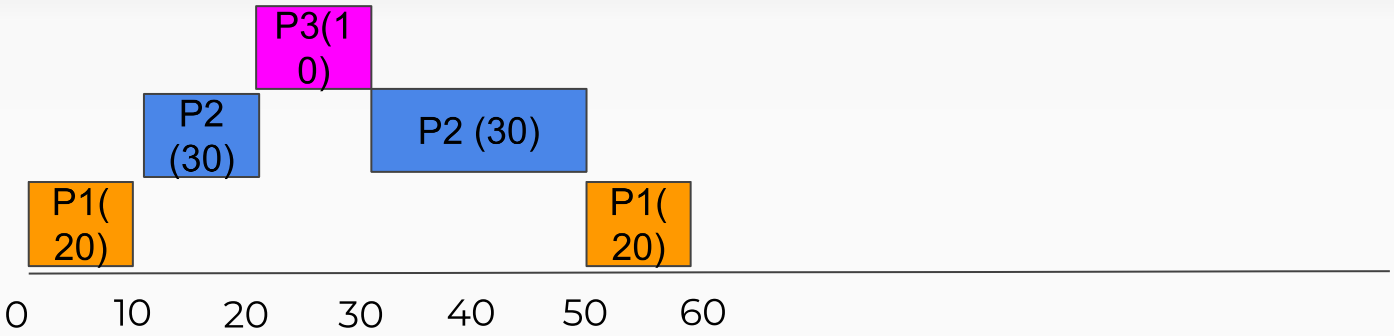
Completion-time(P1) = 40

Completion-time(P2) = 60

Completion-time(P3) = 30

Scheduling Disciplines - Priority

Priority



Arrival-time(P1) = 0

Arrival-time(P2) = 10

Arrival-time(P3) = 20

Completion-time(P1) = 60

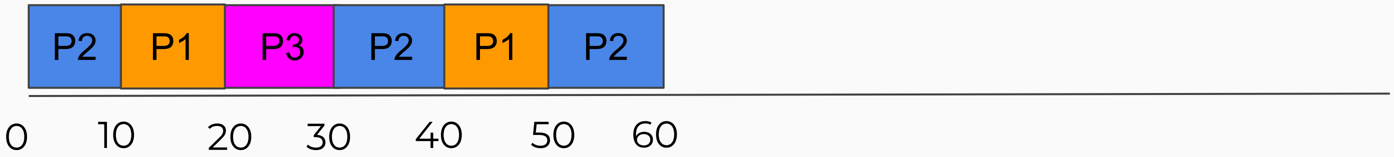
Completion-time(P2) = 50

Completion-time(P3) = 30

Scheduling Discipline - Round robin

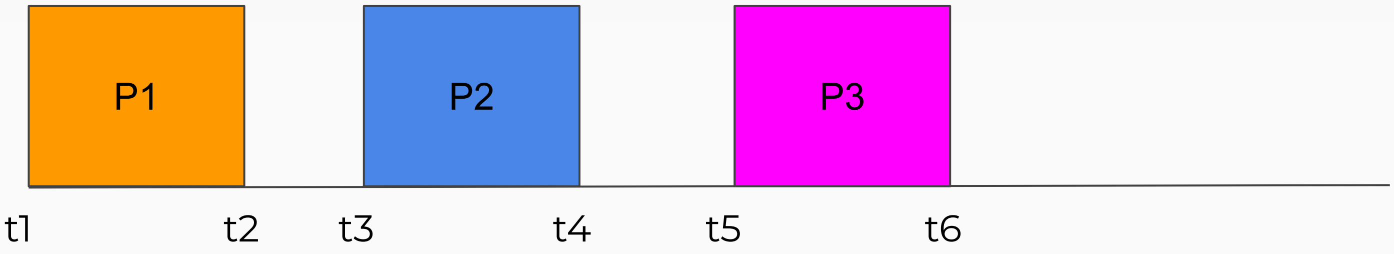
Round-robin Scheduling

- Run job for a **time slice** and then switch
  - Scheduling quantum
- Repeats until the jobs complete
- Length of time slice is multiple of timer interrupt
- Pessimal with turnaround time



Other Scheduling Metrics

Response time



Response-time(P) = First-run-time(P) - Arrival-time(P)

Response-time(P1) = t1 - t1

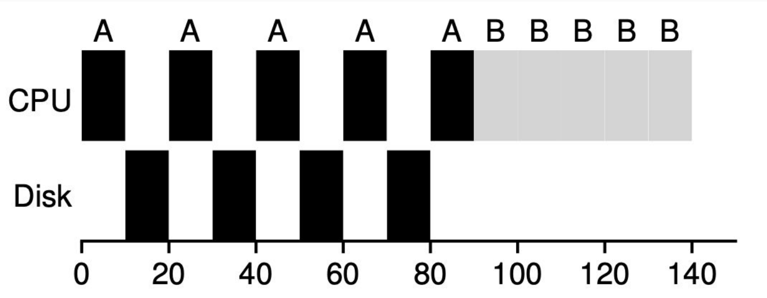
Response-time(P2) = t3 - t3

RR scheduling - Amortization

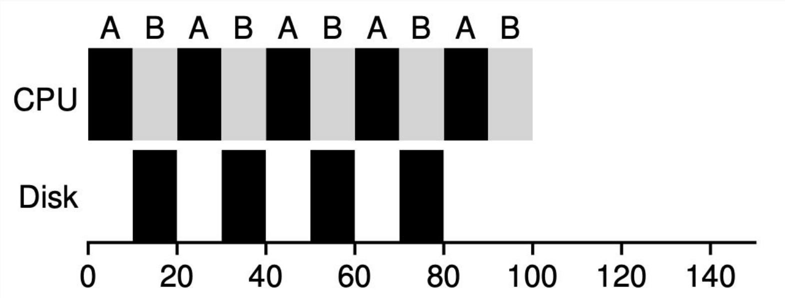
- Length of time slice critical
  - Shorter : better performance under response time
  - Too small time slice : Context-switching costs
- Amortization is used when fixed cost of operations
  - Incur that cost less often
  - Time slice at 10 sec and context switch 1 sec (10% overhead)
  - Time slice at 100 sec and context switch 1 sec (1% overhead)

Scheduling with I/O

- Programs perform I/O
- Processes block when performing I/O



Bad usage



Good usage

Job Lengths

- How does the OS know each job's running time in SJF/STCF?
  - Normally, it doesn't

Multi-Level Feedback Queue (MLFQ)

- Described by Corbato et al. (1962)
  - Compatible Time-Sharing System (CTSS)
  - Turing Award

Multi-Level Feedback Queue (MLFQ)

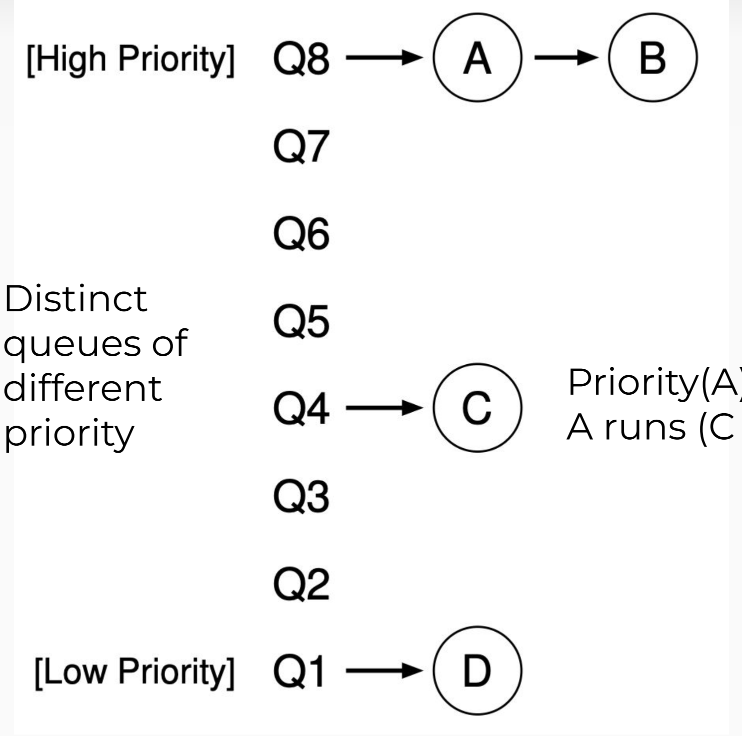
- Described by Corbato et al. (1962)
  - Compatible Time-Sharing System (CTSS)
  - Turing Award

- Optimize turnaround time
  - Do not know running time
- Optimize response time
  - Round-robin scheduling (but with better turnaround time?)

MLFQ

- Number of distinct queues
  - each assigned a different priority level (decreasing order)
- A job that is ready to run is on a single queue
  - priorities to decide which job should run at a given time
- More than one job may be on a given queue
  - have the same priority
  - round-robin scheduling among those jobs.

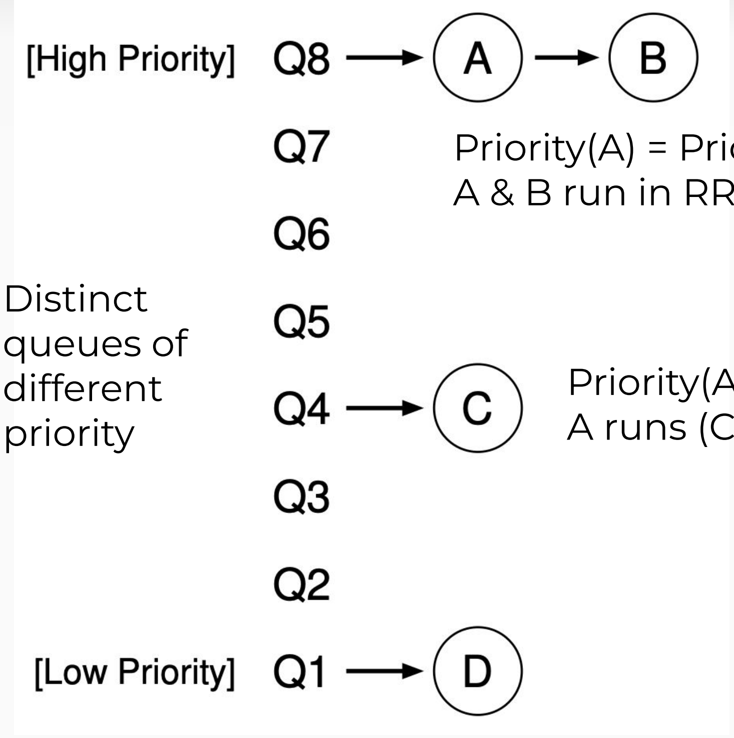
MLFQ : Rules



Distinct queues of different priority

Priority(A) > Priority(C):
A runs (C doesn't)

MLFQ : Rules



Priority(A) = Priority(B):
A & B run in RR

Distinct queues of different priority

Priority(A) > Priority(C):
A runs (C doesn't)

MLFQ: Priorities

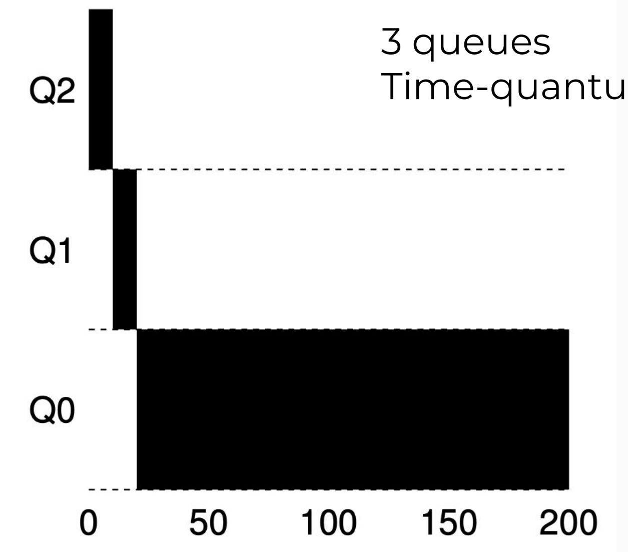
- How to set the priorities
  - Vary priority based on behavior
  - Interactive job (repeatedly switches b/w CPU and I/O)
    - High priority
  - CPU Intensive
    - Reduce priority

MLFQ: How to change the priorities

- Mix of interactive and CPU-intensive jobs
- Algorithm

MLFQ: How to change the priorities

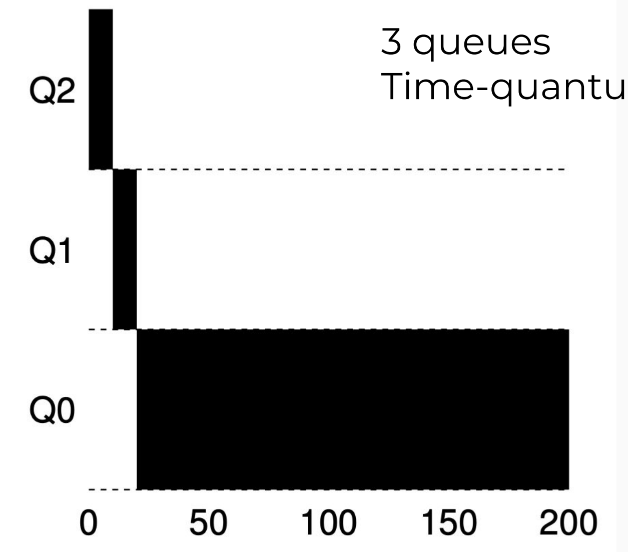
- Mix of interactive and CPU-intensive jobs
- Algorithm



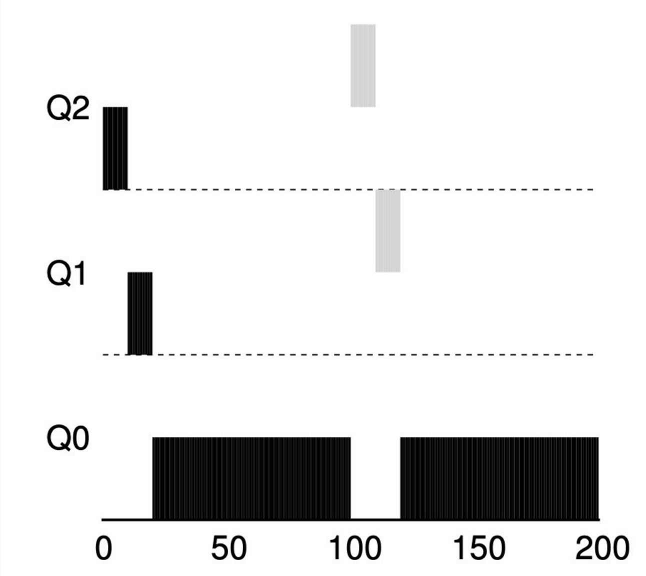
3 queues
Time-quantum = 10ms

MLFQ: How to change the priorities

- Mix of interactive and CPU-intensive jobs
- Algorithm

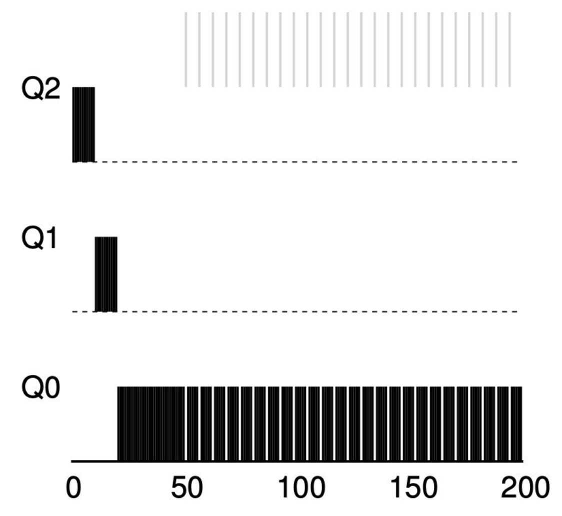


3 queues
Time-quantum = 10ms



MLFQ: How to change the priorities

- Mix of interactive and CPU-intensive jobs
- Algorithm



3 queues
Time-quantum = 10ms

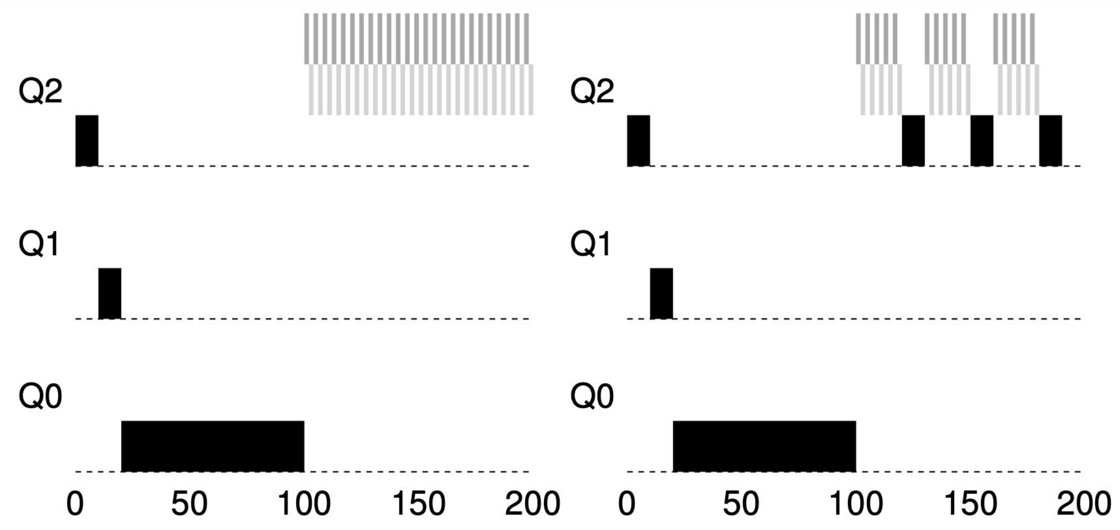
MLFQ: How to change the priorities

Problems with current algorithm:

- Starvation
- Trick the scheduler
- Benign program’s behavior change

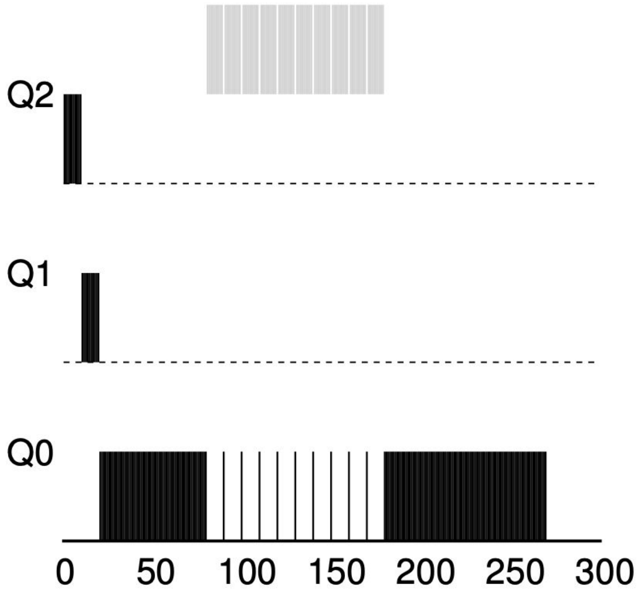
MLFQ: Priority Boost

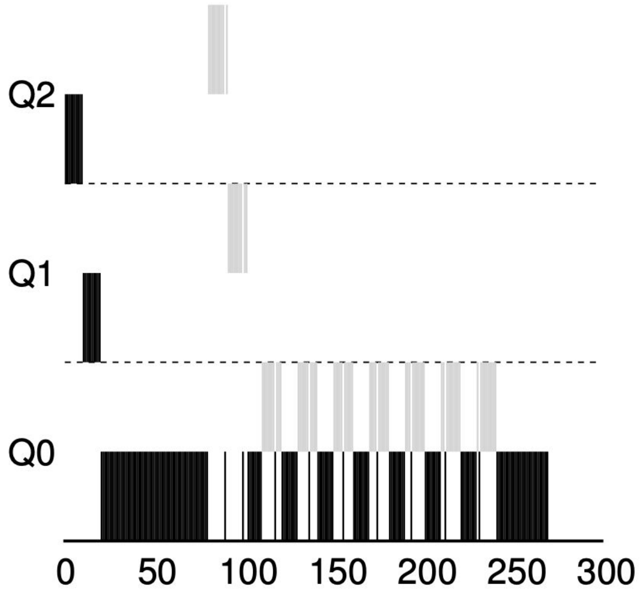
- After some time period S, move all the jobs in the system to the topmost queue



MLFQ: Accounting of CPU time

- Once a process has used its allotment, it is demoted to the next priority queue





MLFQ

- If Priority(A) > Priority(B), A runs (B doesn’t)
- If Priority(A) = Priority(B), A & B run in RR
- When a job enters the system, it is placed at the highest priority queue
- Once a job uses up its time allotment at a given level its priority is reduced (i.e., it moves down one queue)
- After some time period S, move all the jobs in the system to the topmost queue

Uniprocessor Scheduling

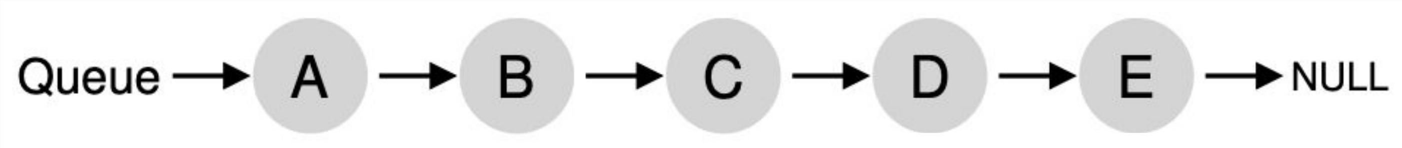
- FIFO is simple and minimizes overhead.
  - If tasks are variable in size, then FIFO can have very poor average turnaround time.
  - If tasks are equal in size, FIFO is optimal in terms of average turnaround time.
- SJF is optimal in terms of avg turnaround time.
  - SJF is pessimal in terms of variance in turnaround time.
  - If tasks are variable in size, Round Robin approximates SJF.
- For equal tasks, RR will have poor avg turnaround time.
  - Tasks that intermix processor and I/O benefit from SJF and can do poorly under Round Robin.
  - RR avoids starvation

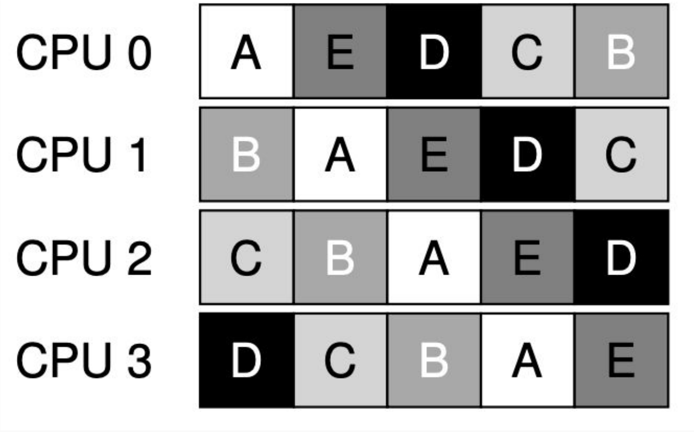
Multiprocessor Scheduling

- Different from single-processor
  - Contains multiple CPUs
  - Multiple caches
  - Data sharing across multiple processors
- Issues due to caches
  - Data cached in CPU 1 may be required in CPU 2 due to scheduling pattern
    - Cache coherence
    - Locality - temporal and spatial - affected
  - Cache affinity

Multiprocessor Scheduling

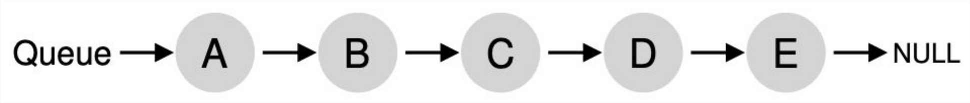
- Single queue
    - Simple

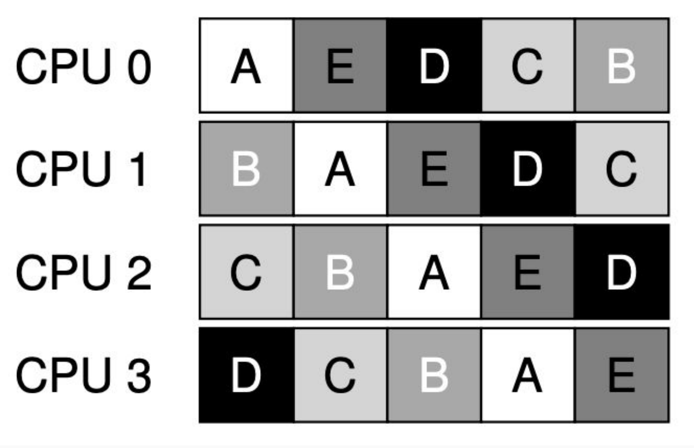




Multiprocessor Scheduling

- Single queue
  - Simple
  - Shortcomings?
    - Scalability - multiple processors cannot access the queue at the same time
    - Cache affinity may be affected



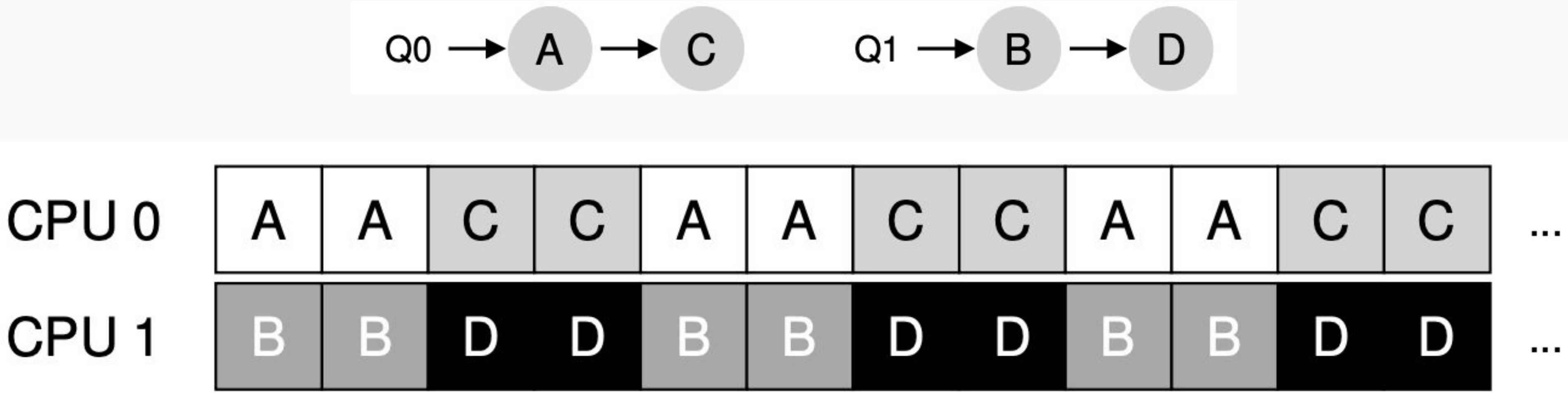


Multiprocessor Scheduling

- Single queue
  - Simple
  - Shortcomings
    - Scalability
    - Cache affinity
- Multiple queues
  - Multiple scheduling queues follow their own scheduling algo.
  - OS decides which CPU to schedule on
  - Scalable with cache affinity

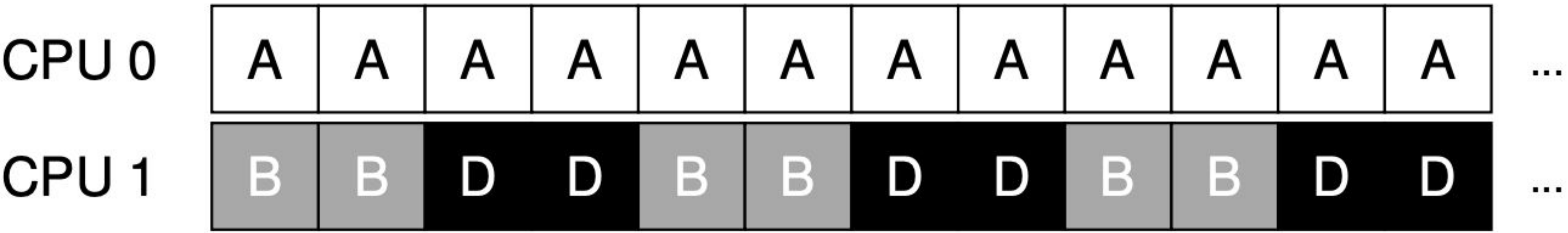
Multiprocessor Scheduling

- Multiple queues
  - Multiple scheduling queues follow their own scheduling algo.
  - OS decides which CPU to schedule on
  - Scalable with cache affinity



Load Imbalance and Migration

- Suppose job C finishes

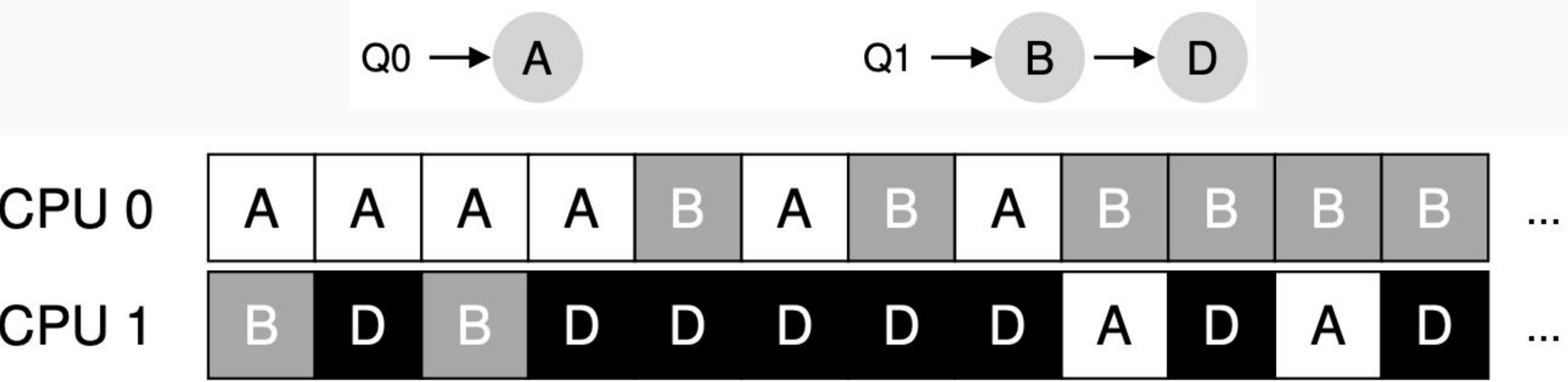


Load Imbalance and Migration

- Or both job A & C finish

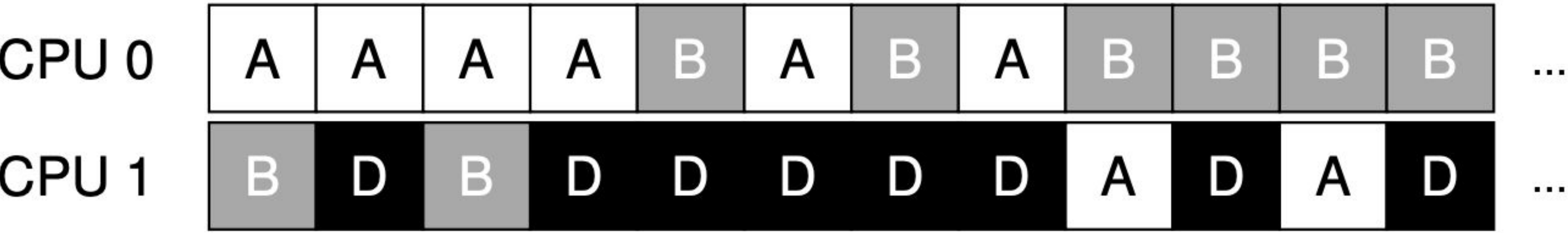
Load Imbalance and Migration

- Switch jobs occasionally



Load Imbalance and Migration

- Switch jobs occasionally



- How to migrate?
  - Work stealing