CS 181AI
Lecture 18

Serving ML Models: Scheduling

Arthi Padmanabhan

Mar 27 2023

Logistics

- Project proposals due today 10pm
- Wednesday – we'll start working on a problem in class in groups. Your assignment 5 is to complete the problem
- Next Monday (4/3): working session (instead of 4/12)

Today

- Scheduling principles and algorithms

Preemptible vs. Non-preemptible

- A preemptible resource can be taken away and used for something else
  - E.g., a GPU
- A preemptible resource is shared through explicit scheduling
- A non-preemptible resource cannot be taken away without acknowledgement
  - E.g., GPU memory
- A non-preemptible resource is shared through allocations and deallocations

Scheduler

- A scheduler is a high-level policy to decide which jobs to run when
- It is not responsible for the details of context-switching

When Does a Scheduler Run?

- A scheduler runs when a job changes state
- Let’s first consider a job that cannot be preempted – once it starts, it runs until completion
- In this case, the scheduler will only make a decision once the job finishes

Jobs

- For now, we will think of ML jobs at discrete, e.g., run ResNet50 on these 1000 images with a batch size of 16
  - Reality: if processing video, you might also have to think about how to make discrete jobs when frames arrive continually
- There are often several (tens of) models trying to run jobs on a single GPU

Metrics

- What makes a scheduling policy “good”?

Metrics

- Minimize waiting time and response time
  - Don’t have jobs waiting too long to start
- Maximize GPU utilization
  - Don’t have idle GPU
- Maximize throughput
  - Complete as many jobs as possible
- Fairness
  - Try to give each process a similar percentage of the GPU

First Come First Serve (FCFS)

- The simplest form of scheduling
- GPU runs ML jobs in the order they arrived

A Gantt Chart Illustrates the Schedule

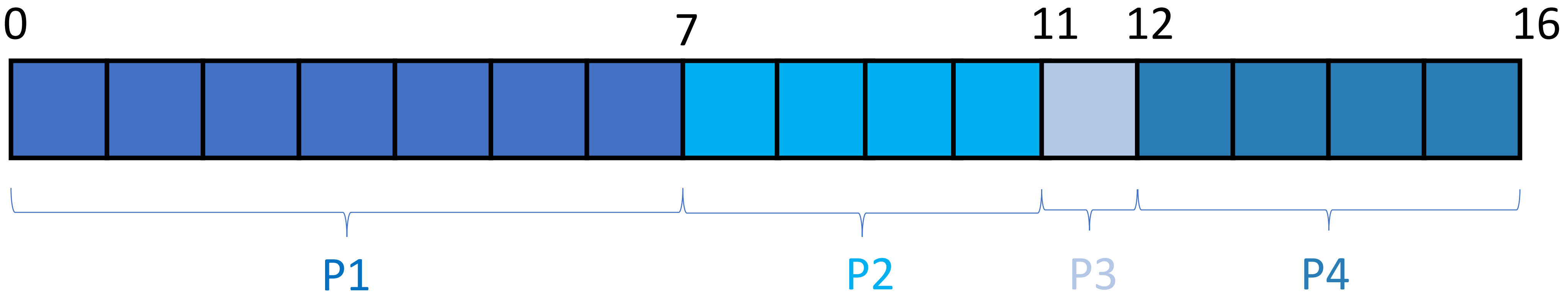
Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	1
P4	0	4

Assume they arrive in the order P1 -> P2 -> P3 -> P4. What is the average waiting time?

A Gantt Chart Illustrates the Schedule

Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	1
P4	0	4

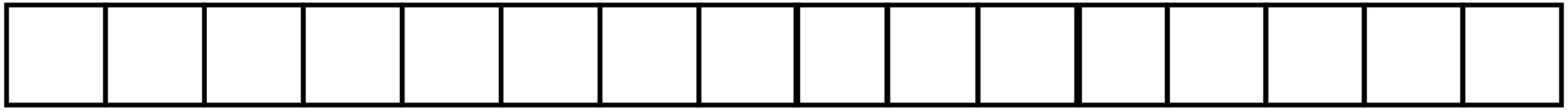
Assume they arrive in the order P1 -> P2 -> P3 -> P4. What is the average waiting time?



Different Arrival Order?

Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	1
P4	0	4

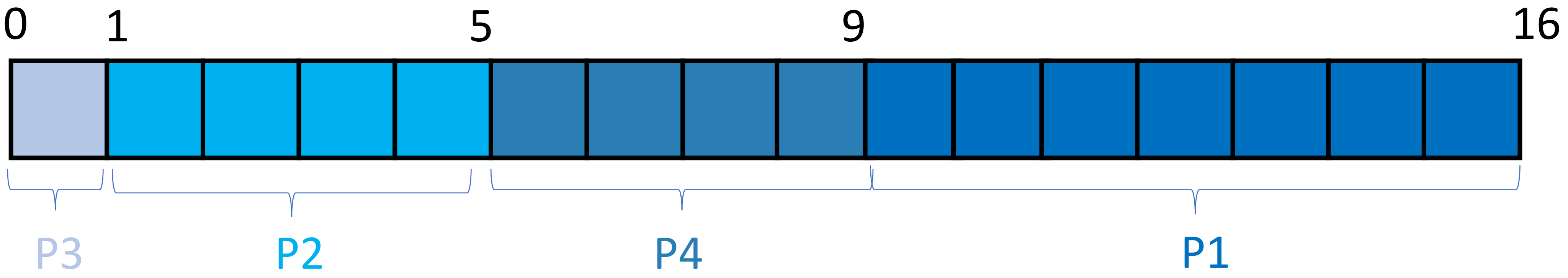
Assume they arrive in the order P3 -> P2 -> P4 -> P1. What is the average waiting time?



Different Arrival Order?

Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	1
P4	0	4

Assume they arrive in the order P3 -> P2 -> P4 -> P1. What is the average waiting time?

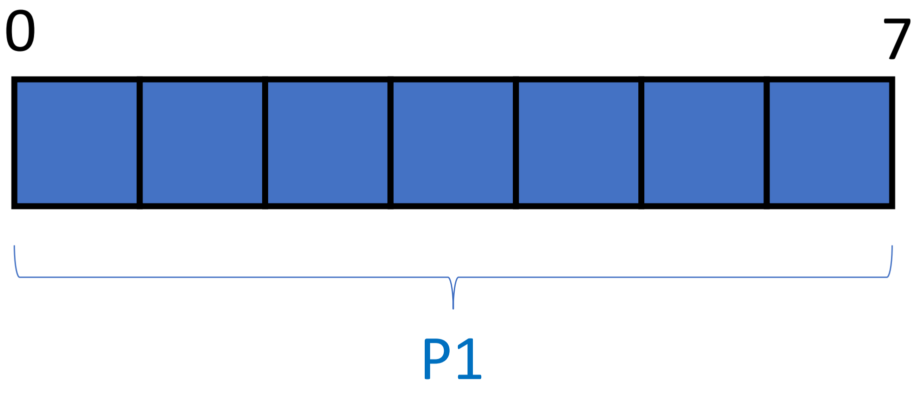


Shortest Job First

- Slight tweak to FCFS
- Always schedule the job with the shortest burst time first

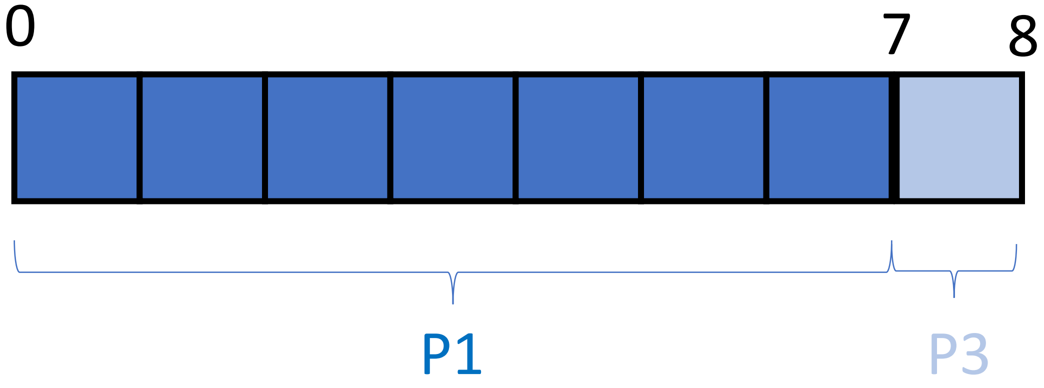
Shortest Job First

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



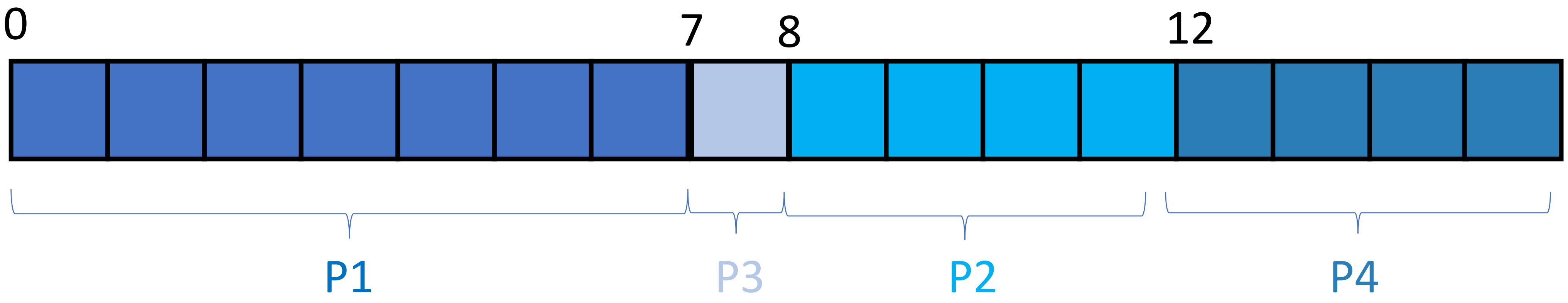
Shortest Job First

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



Shortest Job First (SJF)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



Is this always a good idea?

Is this always a good idea?

- You sometimes won’t know exactly how long a process takes
  - If you’ve run the model before, you’ll probably have some idea
- You might starve longer jobs (they may never execute) -> not good for fairness

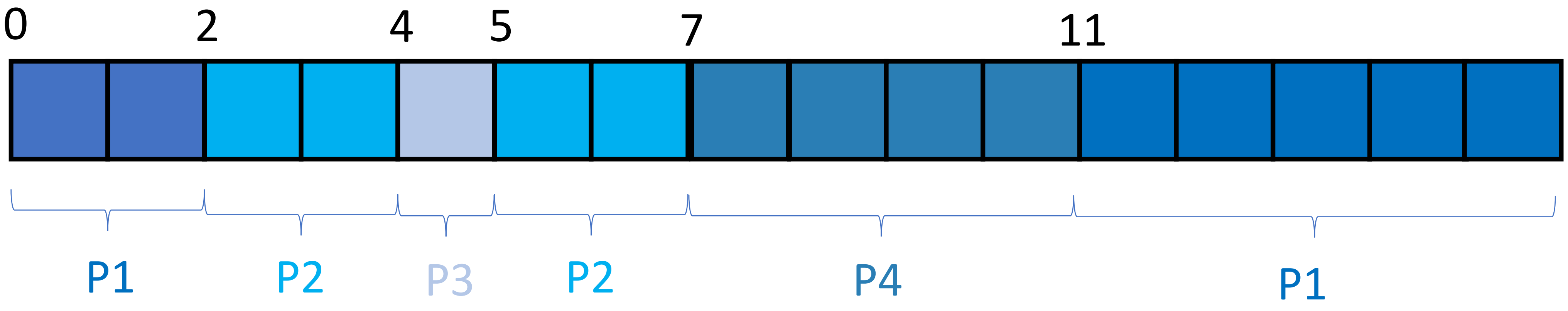
Adding preemptions

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Shortest Remaining Time First (SRTF)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Further reduces average waiting time

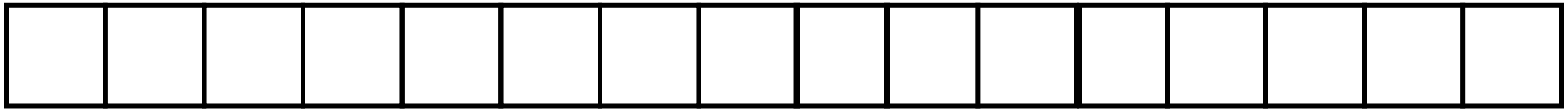


Round-Robin

- So far we haven’t handled fairness (it’s a tradeoff with the others)
- Scheduler divides time into slots (also called quanta, individual: quantum)
- Maintain a FCFS queue
    - Preempt if still running and re-add to queue

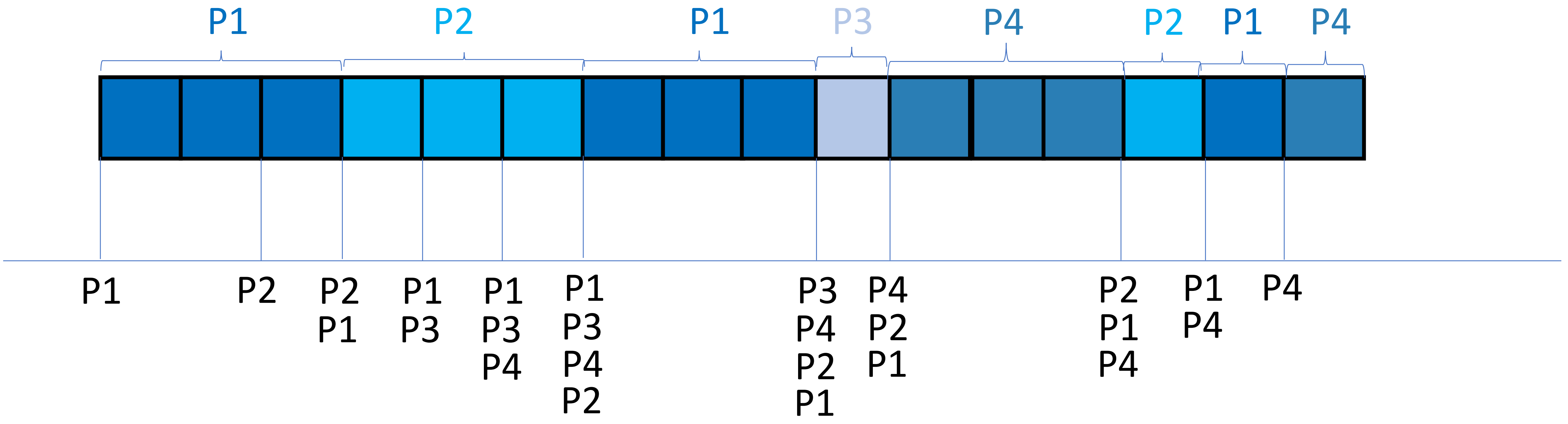
Round-Robin: Quantum = 3

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



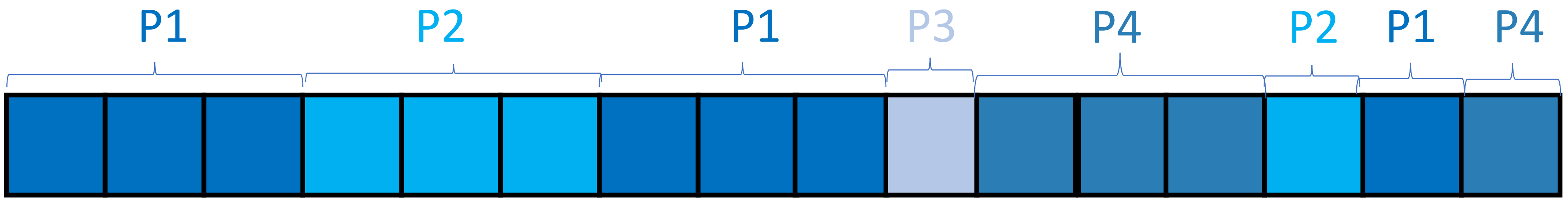
Round-Robin: Quantum = 3

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



Round-Robin: Quantum = 3

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

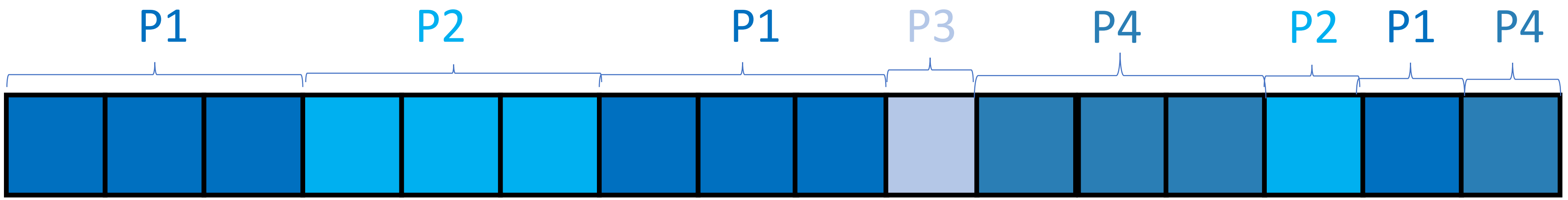


Number of context switches?
Average wait time?
Average response time?

Note on ties: if a new job is added exactly when one is preempted, favor the new one

Round-Robin: Quantum = 3

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



Number of context switches: 7
Average wait time: 7
Average response time: 2.75

Round-Robin: Quantum = 1

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Number of context switches?

Average wait time?

Average response time?

Round-Robin: Quantum = 10

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Number of context switches?
Average wait time?
Average response time?

Round-Robin Performance

- Depends on job length and quantum length
- Quantum length too low -> too many context switches
- Quantum length too high -> FCFS (high response time)
- Poor average waiting time when jobs are similar in length
- It is fair!

Next Time

- We'll start writing code to implement these scheduling policies, which will be part of your assignment

Acknowledgements

- Jon Eyolfson: UCLA CS 111