workloads
- batch / computational
- bursty + io

> support both

in these real workloads: do we really know time (#4?)
do we really know what kind of job?
(maybe we do! but unix didn't)

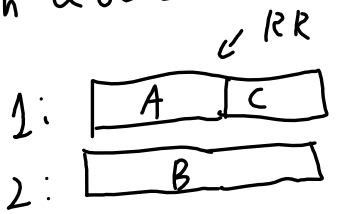
algorithm: MLFQ

Multi-level-feedback-queueing

idea: learn type of job over time

Queues with priority
(DMV, doctor office, etc)
MVC

RR within Queue



metric: turnaround time
response time
time-to-completion
to IO, etc.
not just time-to-first run

Sidebar / review: RR

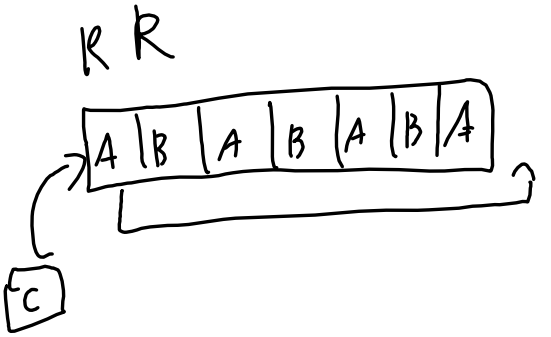
- slice jobs into individual quanta
- (how to set Quanta? magic number!
  100 us = 0.1 ms on courselab (minimum))
  L sidebar: /proc/sys/kernel, check it out
- balance responsiveness vs pre-emption

RR: live scheduling?

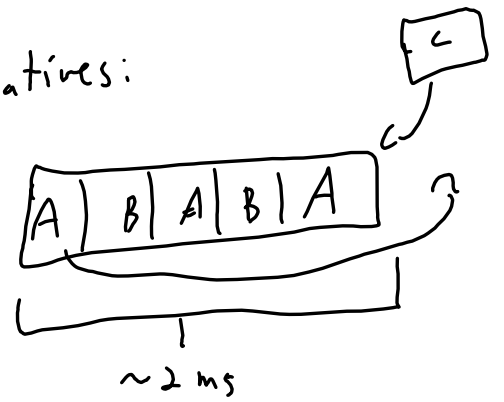
- book assumes fixed schedule

challenge: where to put new jobs?

last time:

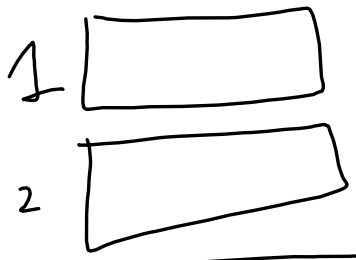


alternatives:



sidebar: flexible quantum?

back to MLFQ



Rules:

#1: run high-priority first

#2: RR within Queue

remember: we have to implement
what DS? what are we tracking?

Challenge: where to put jobs? How to set priority?

More rules: #3: enter at top

#4a: demote on timer

#4b: keep on yield

(no new info to track!)

problems? (starvation) (game the sched)

benefits? (responsiveness)

---

try again: add: rule #5: merge into top every N us.

what does this solve?

Try again: what's better? what's wrong?
(gaming)

fix gaming: track more state.
replace rule #4a, b!

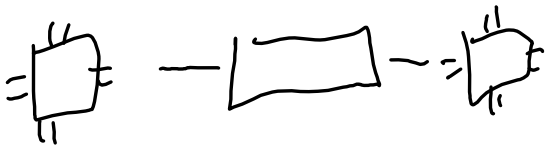
Full rules:
#1: run high-priority first
#2: RR within Queue
#3: enter at top
#4: demote after allotment used
#5: merge into top every N us.

- discuss: parameterization!

Nice: user-level control
- how long is a Q? can we enforce this?
- what if we know interactivity?

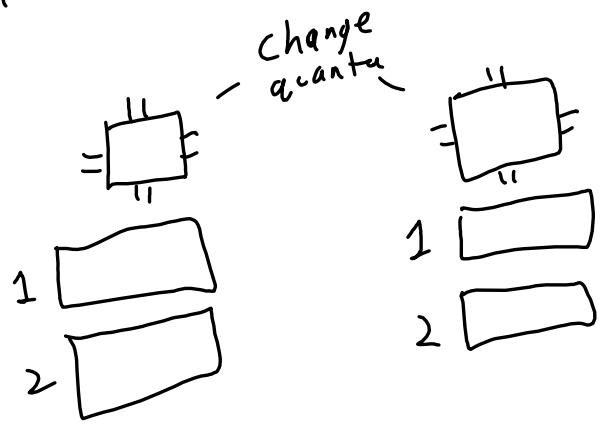
Modern-day: multicore

(NOT on the exam)



- sched per core?
- cores with specific purpose?

multi-core MLFQ



"specialize" processors

example: compute clusters

timer: core migration (configurable!)
look at /proc/sys/kernel

A fun alternative: lottery

- use randomness!

- process launch: gain N tickets (random)
- sched: select K random tickets
  (with replacement)

Benefits:
- ticket transfer
- low state tracking
- subdivide tickets
  - give tickets to users, e.g.

limitations:
- unpredictable
- source of randomness

CFS! (probably won't get here)

- use a min-heap