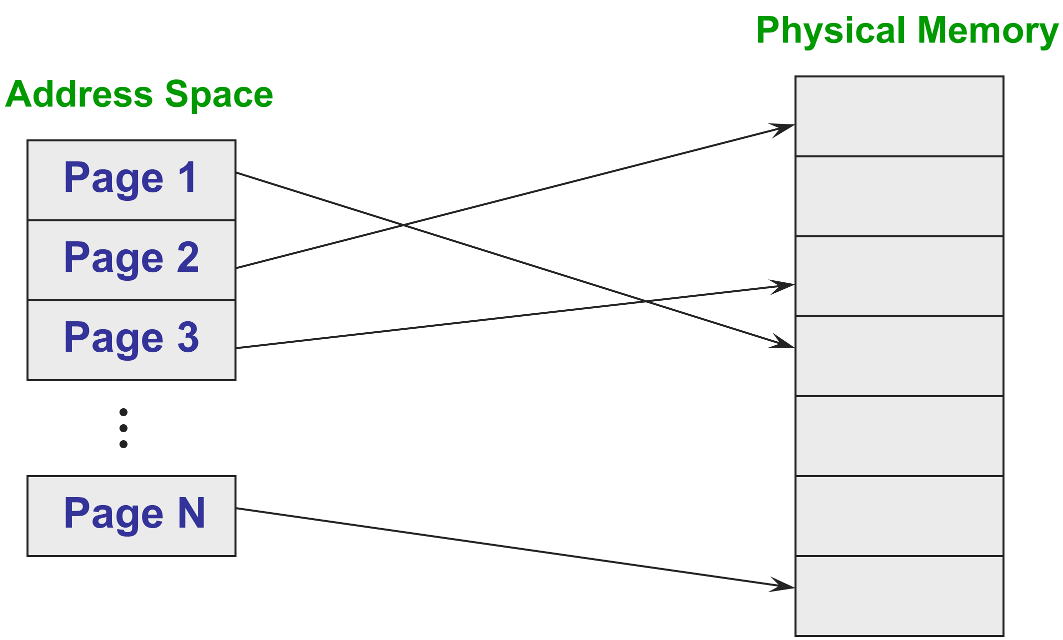
# EECS 482
# Introduction to Operating Systems

## Winter 2018

Harsha V. Madhyastha

# Recap: Paging

- Both address spaces and physical memory broken up into fixed size pages



# Recap: Paging

- Virtual address to physical address translation using page table

| Virtual page # | Physical page # | Protection |
|---|---|---|
| 0 | 105 | RX |
| 1 | 15 | R |
| 2 | 283 | RW |
| 3 | invalid | |
| ... | invalid | |
| 1048575 | invalid | |

- Can manipulate protection bits to maintain other bits (resident, referenced, dirty) in OS

# Recap: Page Replacement

- Not all virtual pages can be in physical mem.
- Steady state: Evict a page to make another page resident
  - Use reference bit to identify pages to evict
  - Use dirty bit to identify need for write-back

# Recap: Process creation

- System calls to start a process:
  1. Fork() creates a copy of current process
  2. Exec(program, args) replaces current address space with specified program

- How to optimize execution of fork?

# Processes sharing memory

- How to divide phys. memory among processes?
  - Goals: fairness versus efficiency
- Global replacement
  - Can evict pages from faulting process or any other
- Local replacement
  - Can evict pages only from faulting process
  - Must determine how many frames each process gets
- Pros and cons?

# Thrashing

- What happens if many large processes all actively use their entire address space?

- Performance degrades rapidly as miss rate goes up
  - Avg access time = hit rate * hit time + miss rate * miss time
  - E.g., hit time = .0001 ms; miss time = 10 ms
    - » Average access time (100% hit rate) = .0001 ms
    - » Average access time (1% miss rate) = .100099 ms
    - » Average access time (10% miss rate) = 1.00090 ms

# Solutions to Thrashing

- Buy more DRAM
  - Very common solution in cloud servers
  - Price per GB fallen by 4x since 2009
- Run fewer processes for longer time slices
  - Reduces page faults
  - But, poor interactivity due to long time slices

# Working set

- Thrashing depends on portion of address space actively used by each process
  - What do we mean by “actively using”?
- Working set = all pages used in last T seconds
  - Larger working set → need more memory
- Sum of all working sets should fit in memory
  - Only run subset of processes that fit in memory
- How to measure size of working set?

# Project 3

- Hope you have a state machine for swap-backed pages by now???
- Things to consider:
  - Transitions?
  - Properties that capture state of a page?
  - Protection bits?
- Don’t translate state machine into if-else cases!
- Think ahead in designing data structures

# Project 3: App vs. OS

- Protection
  - All pages can be read from and written to
  - Using R/W bits to track reference, dirty, etc.
- Sharing
  - File-backed pages
  - Copy-on-write

# CPU scheduling

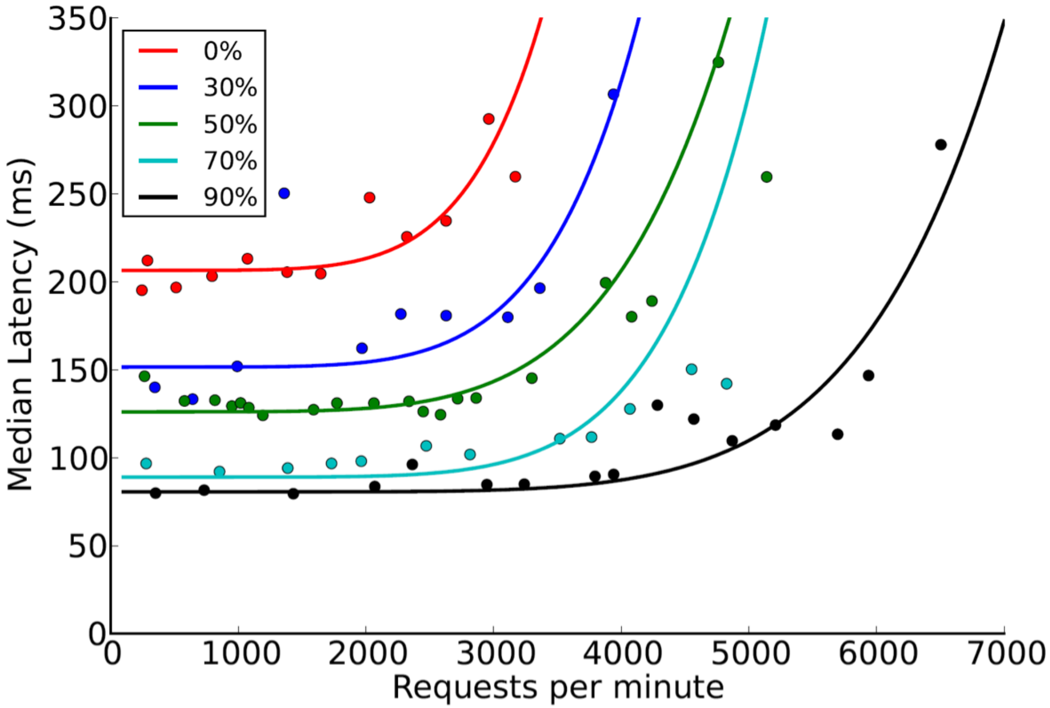
- If >1 thread is ready, choose which to run

- Many possible scheduling policies
  - Goal today is to explore fundamental ones
  - Real schedulers often a complex mix of policies

# Scheduling: Goals

- What are good goals for a CPU scheduler?
  - Minimize average response time
  - Maximize throughput
  - Fairness
- “Minimize latency” at odds with “maximize tput”

# Throughput-response curves




- Collected from Facebook production service [Chow '16]
  - Each colored line: throughput vs. latency at different quality
  - Left of graph – adding load → little effect on response time
  - Right of graph – adding load → exponential increase in latency

# Load testing

# Fairness

- Share CPU among threads in equitable manner
- How to share between 1 big and 1 small job?
  - Response time proportional to job size?
  - Or equal time for each job?
- Fairness often conflicts with response time

# Starvation = extremely unfair

- Starvation can be outcome of synchronization

- Starvation can also be outcome of scheduling

# First-come, first-served (FCFS)

- FIFO ordering among jobs
- No preemption (no timer interrupts)
  - Thread runs until it calls yield() or blocks

# FCFS Example

- Job A: Arrives at t=0, takes 100 seconds
- Job B: Arrives at t=0+, takes 1 second

# FCFS Summary

- Pros:
  - Simple to implement
- Cons:

# Round Robin

- Improve average response time for short jobs
- Add preemptions (via timer interrupts)
  - Fixed time slice (time quantum)
  - Preempt if still running when time slice is over

# Round Robin Example

- Job A: Arrives at t=0, takes 100 seconds
- Job B: Arrives at t=0+, takes 1 second

# Choosing a time slice

- What's the problem with a big time slice?
- What's the problem with a small time slice?
- OS typically compromises: e.g., 1ms or 10ms

# Round Robin Summary

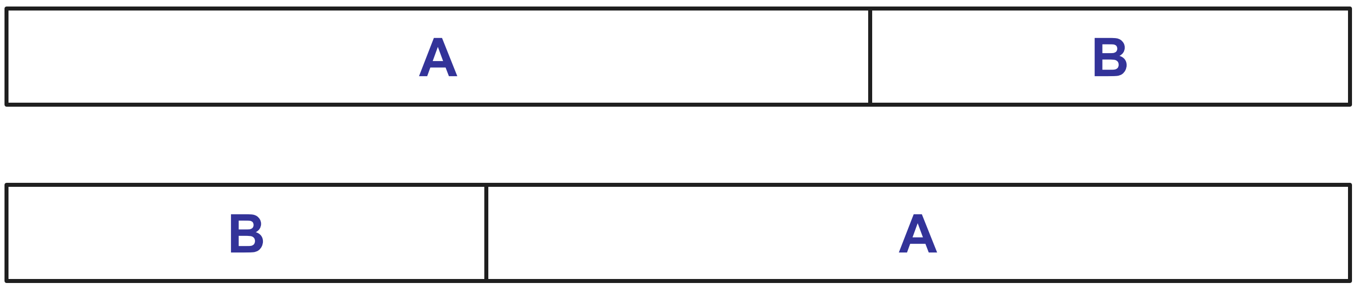
- Pros:
  - Still pretty simple
  - Good for interactive computing
- Cons?
- Comparison: Does RR always reduce average response time vs. FCFS?

# Round Robin vs. FCFS

# STCF

- Shortest time to completion first
- Run job with least work to do
  - Preempt current job if shorter job arrives
  - Job size is time to next blocking operation
- Finish short jobs first
  - Improves response time of short jobs (by a lot)
  - Hurts response time of long jobs (by a little)
- STCF gives optimal average response time

# Analysis of STCF




- Consider 2 jobs: A longer than B

# STCF

- Pro:
  - Optimal average response time
- Cons?

# Predicting job run times

- Ask the job or the user?

- OS schedulers often identify interactive apps and boost their priority

# Priority

- Priority
  - Assign external priority to each job
  - Run high-priority jobs before low-priority ones
  - Use, e.g., round-robin for jobs of equal priority
  - Prone to starvation

- Methods for preventing starvation?

# Multimedia: Soft real-time

- Examples:
  - Audio should not skip when compiling projects
  - Predictable: video player plays n frames per sec
- Can reserve a share of the CPU
  - X% of the CPU over some time interval
  - Unused CPU split among remaining jobs

# Hard real-time scheduling

- Jobs have to complete before deadline
  - Demand / deadline known in advance
  - Example: vehicle control, aviation, etc.

- Earliest-deadline first (EDF)
  - Always run jobs whose deadline is soonest
  - Preempt if newly arriving job has earlier deadline
  - Always succeeds if schedule is feasible
  - But, may be very poor if schedule is infeasible

# Scheduling: Summary

- Many different policies
  - FCFS
  - Round robin
  - STCF
  - Priority
  - Proportional share
  - EDF
- Scheduling strategy in grocery stores?
- OS schedulers mix all of these
  - Many heuristics and complex tuning

# Next time ...

- File systems