UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Introduction to Operating Systems

Andrea C. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau

CPU Scheduling

Questions answered in this lecture:

- What is scheduling vs. allocation?
- What is preemptive vs. non-preemptive scheduling?
- What are FCFS, SJF, STCF, RR and priority-based scheduling policies?
- What are their advantages and disadvantages?

Types of Resources

Resources can be classified into one of two groups

Type of resource determines how the OS manages it

1) Non-preemptible resources
   - Once given resource, cannot be reused until voluntarily relinquished
   - Resource has complex or costly state associated with it
   - Need many instances of this resource
   - Example: Blocks on disk
   - OS management: allocation
     - Decide which process gets which resource
2) Preemptible resources
   - Can take resource away, give it back later
   - Resource has little state associated with it
   - May only have one of this resource
   - Example: CPU
   - OS management: scheduling
     - Decide order in which requests are serviced
     - Decide how long process keeps resource

Levels of CPU Management

Dispatcher
- Low-level mechanism
- Performs context-switch
  - Save execution state of old process in PCB
  - Add PCB to appropriate queue (ready or blocked)
  - Load state of next process from PCB to registers
  - Switch from kernel to user mode
  - Jump to instruction in user process

Scheduler
- Policy to determine which process gets CPU when

Allocator
- Policy to determine which processes compete for which CPU
- Needed for multiprocessor, parallel, and distributed systems

CPU Workload Model

Workload contains collection of jobs (processes)

Job model
- Job alternates between CPU and I/O bursts (i.e., moves between ready and blocked queues)
- CPU-bound job: Long CPU bursts
- I/O-bound job: Short CPU bursts

Do not know type of job before it executes
- Do not know duration of CPU or I/O burst

Need job scheduling for each ready job
- Schedule each CPU burst

Scheduling Performance Metrics

Minimize waiting time
- Do not want to spend much time in Ready queue

Minimize turnaround time
- Do not want to wait long for job to complete

Maximize throughput
- Want many jobs to complete per unit of time

Minimize response time
- Schedule interactive jobs promptly so users see output quickly

Maximize resource utilization
- Keep expensive devices busy

Minimize overhead
- Reduce number of context switches

Maximize fairness
- All jobs get same amount of CPU over some time interval

When may Scheduler switch?

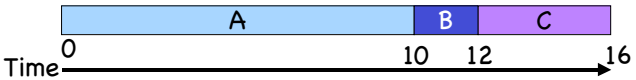
Non-preemptive scheduler
- Process remains scheduled until voluntarily relinquishes CPU
- Scheduler may switch in two cases:
  -
  -

Preemptive scheduler
- Process may be descheduled at any time
- Additional cases:
  -
  -
  -

Gantt Chart

Illustrates how jobs are scheduled over time on CPU

Example:

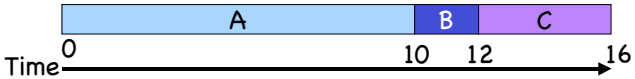


First-Come-First-Served (FCFS)

Idea: Maintain FIFO list of jobs as they arrive
- Non-preemptive policy
- Allocate CPU to job at head of list

Job	Arrival	CPU burst
A	0	10
B	1	2
C	2	4

Average wait time:
(0 + (10-1) + (12-2))/3=6.33
Average turnaround time:
(10 + (12-1) + (16-2))/3=11.67

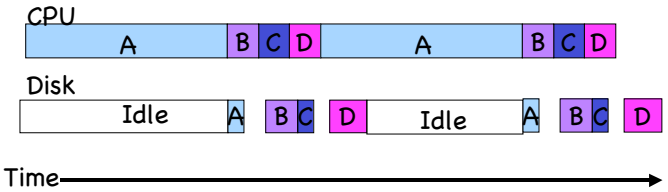


FCFS Discussion

Advantage: Very simple implementation

Disadvantage

- Waiting time depends on arrival order
- Potentially long wait for jobs that arrive later
- Convoy effect: Short jobs stuck waiting for long jobs
  - Hurts waiting time of short jobs
  - Reduces utilization of I/O devices
  - Example: 1 mostly CPU-bound job, 3 mostly I/O-bound jobs



Shortest-Job-First (SJF)

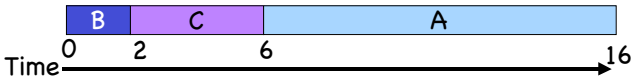
Idea: Minimize average wait time by running shortest CPU-burst next

- Non-preemptive
- Use FCFS if jobs are of same length

Job	Arrival	CPU burst
A	0	10
B	0	2
C	0	4

Average wait:

Average turnaround:



SJF Discussion

Advantages

- Provably optimal for minimizing average wait time (with no preemption)
  - Moving shorter job before longer job improves waiting time of short job more than it harms waiting time of long job
- Helps keep I/O devices busy

Disadvantages

- Not practical: Cannot predict future CPU burst time
  - OS solution: Use past behavior to predict future behavior
- Starvation: Long jobs may never be scheduled

Shortest-Time-to-Completion-First (STCF or SCTF)

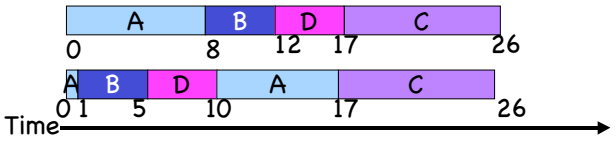
Idea: Add preemption to SJF

- Schedule newly ready job if shorter than remaining burst for running job

Job	Arrival	CPU burst
A	0	8
B	1	4
C	2	9
D	3	5

SJF Average wait:

STCF Average wait:



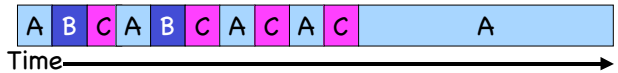
Round-Robin (RR)

Idea: Run each job for a time-slice and then move to back of FIFO queue

- Preempt job if still running at end of time-slice

Job	Arrival	CPU burst
A	0	10
B	1	2
C	2	4

Average wait:



RR Discussion

Advantages

- Jobs get fair share of CPU
- Shortest jobs finish relatively quickly

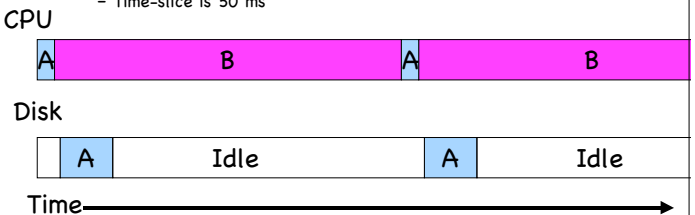
Disadvantages

- Poor average waiting time with similar job lengths
  - Example: 10 jobs each requiring 10 time slices
  - RR: All complete after about 100 time slices
  - FCFS performs better!
- Performance depends on length of time-slice
  - If time-slice too short, pay overhead of context switch
  - If time-slice too long, degenerate to FCFS

RR Time-Slice

IF time-slice too long, degenerate to FCFS

- Example:
  - Job A w/ 1 ms compute and 10ms I/O
  - Job B always computes
  - Time-slice is 50 ms



Goal: Adjust length of time-slice to match CPU burst

Priority-Based

Idea: Each job is assigned a priority

- Schedule highest priority ready job
- May be preemptive or non-preemptive
- Priority may be static or dynamic

Advantages

- Static priorities work well for real time systems
- Dynamic priorities work well for general workloads

Disadvantages

- Low priority jobs can starve
- How to choose priority of each job?

Goal: Adjust priority of job to match CPU burst

- Approximate SCTF by giving short jobs high priority