CS323 Operating Systems
CPU Scheduling II

Yuanyuan Zhou
Lecture 6
2/3/2003

Content of this lecture

- Administrative announcements
- Scheduling algorithms
  - Batch systems
    - Shortest job first
  - Interactive systems
    - Round-robin
    - Priority scheduling
  - Multi Queue & Multi-level Feedback
  - Shortest process time
  - Guaranteed Scheduling
  - Lottery Scheduling
  - Fair Sharing Scheduling
- Summary

Administrative

- Quiz1starts today, due Friday 5pm
- MP1(thread scheduling) starts

Review: Scheduling

- Deciding which process/thread should occupy the resource (CPU, disk, etc)
- Preemptive vs. non-preemptive
- Scheduling objectives
- Scheduling performance criteria
- First-come-first-serve (FCFS)
  - Process that requests the CPU FIRST is allocated the CPU FIRST.
  - Used in batch systems

Shortest Job First (SJF)

- Schedule the job with the shortest elapse time first
- Scheduling in Batch Systems
- Two types:
  - Non-preemptive
  - Preemptive
- Requirement: the elapse time needs to know in advance
- Optimal if all the jobs are available (provable) simultaneously
  - Gives the best possible AWT (average waiting time)

Non-preemptive SJF: Example

Process	Duration	Order	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0

P4 waiting time: 0
P1 waiting time: 3
P3 waiting time: 9
P2 waiting time: 16

The total time is: 24
The average waiting time (AWT):
(0+3+9+16)/4 = 7

Comparing to FCFS

Process	Duration	Order	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0

P1 waiting time: 0
P2 waiting time: 6
P3 waiting time: 14
P4 waiting time: 21

The total time is the same.
The average waiting time (AWT):
(0+6+14+21)/4 = 10.25
(comparing to 7)

SJF is not always optimal

- Is SJF optimal if all the jobs are not available simultaneously?

Process	Duration	Order	Arrival Time
P1	10	1	0
P2	2	2	2

P1 waiting time: 0
P2 waiting time: 8

The average waiting time (AWT):
(0+8)/2 = 4

What if the scheduler waits for 2 time units? (Do it yourself)

Process	Duration	Order	Arrival Time
P1	10	1	0
P2	2	2	2

P1 waiting time: 4
P2 waiting time: 0

The average waiting time (AWT):
(0+4)/2 = 2

However: waste 2 time units of CPU

Preemptive SJF

- Also called Shortest Remaining Time First
  - Schedule the job with the shortest remaining time required to complete
- Requirement: the elapse time needs to be known in advance

Preemptive SJF: Same Example

Process	Duration	Order	Arrival Time
P1	10	1	0
P2	2	2	2

P1 waiting time: 4-2 =2
P2 waiting time: 0

The average waiting time (AWT):
(0+2)/2 = 1

No CPU waste!!!

A Problem with SJF

- Starvation
  - In some condition, a job is waiting for ever
  - Example: SJF
    - Process A with elapse time of 1 hour arrives at time 0
    - But ever 1 minute, a short process with elapse time of 2 minutes arrive
    - Result of SJF: A never gets to run

Interactive Scheduling Algorithms

- Usually preemptive
  - Time is sliced into quantum (time intervals)
  - Scheduling decision is also made at the beginning of each quantum
- Performance Criteria
  - Min Response time
  - best proportionality
- Representative algorithms:
  - Priority-based
  - Round-robin
  - Multi Queue & Multi-level Feedback
  - Shortest process time
  - Guaranteed Scheduling
  - Lottery Scheduling
  - Fair Sharing Scheduling

Priority Scheduling

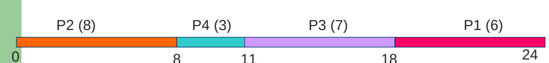
- Each job is assigned a priority.
- FCFS within each priority level.
- Select highest priority job over lower ones.
- Rational: higher priority jobs are more mission-critical
  - Example: DVD movie player vs. send email
- Problems:
  - May not give the best AWT
  - indefinite blocking or starvation a process

Set Priority

- Two approaches
  - Static (for system with well known and regular application behaviors)
  - Dynamic (otherwise)
- Priority may be based on:
  - Cost to user.
  - Importance of user.
  - Aging
  - Percentage of CPU time used in last X hours.

Priority Scheduling: Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0



P2 waiting time: 0
P4 waiting time: 8
P3 waiting time: 11
P1 waiting time: 18

The average waiting time (AWT): (0+8+11+18)/4 = 9.25 (worse than SJF)

Priority in Unix

```
yyzhou|csil-linux1|~|[73]% ps -l
  F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY          TIME CMD
100 S 14828  2047  2045  0  79   4    -   822 rt_sig pts/1    00:00:00 csh
000 R 14828 23001  2047  0  80   4    -   791 -      pts/1    00:00:00 ps
yyzhou|csil-linux1|~|[74]%
yyzhou|csil-linux1|~|[74]%
```

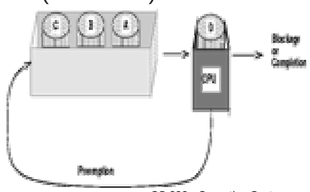

Nobody wants to Be "nice" in Unix

```
NICE(1)                         FSF                         NICE(1)

NAME
       nice - run a program with modified scheduling priority

SYNOPSIS
       nice [OPTION] [COMMAND [ARG]...]

DESCRIPTION
       Run COMMAND with an adjusted scheduling priority.  With no
       COMMAND, print the current scheduling priority.  ADJUST is
       10  by default.  Range goes from -20 (highest priority) to
       19 (lowest).

       -ADJUST
              increment priority by ADJUST first

       -n, --adjustment=ADJUST
              same as -ADJUST

       --help display this help and exit

       --version
lines 1-25
```

Round-robin

- One of the oldest, simplest, most commonly used scheduling algorithm
- Select process/thread from ready queue in a round-robin fashion (take turns)

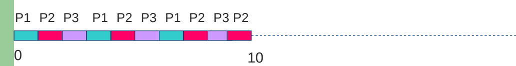
Problem:
- Do not consider priority
- Context switch overhead



Round-robin: Example

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is: 1 unit, P1, P2 & P3 never block



P1 waiting time: 4
P2 waiting time: 6
P3 waiting time: 6

The average waiting time (AWT): (4+6+6)/3 = 5.33

Time Quantum

- Time slice too large
  - FIFO behavior
  - Poor response time
- Time slice too small
  - Too many context switches (overheads)
  - Inefficient CPU utilization
- Heuristic: 70-80% of jobs block within time-slice
- Typical time-slice 10 to 100 ms
- Time spent in system depends on size of job.

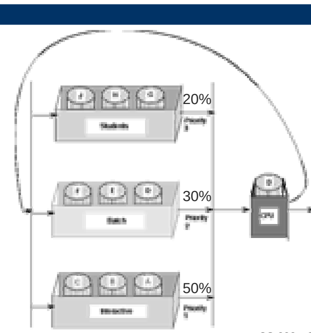
Bag-of-envelop Calculation (1 minute)

- Suppose a context switch takes 1 msec
- Suppose the scheduling quantum is 100 msec
- What is the percentage overhead of context switches in round-robin scheduling?
- What if the scheduling quantum is 10 msec?

Multi-Queue Scheduling

- Hybrid between priority and round-robin
- Processes assigned to one queue permanently
- Scheduling between queues
  - Fixed Priorities
  - % CPU spent on queue
- Example
  - System processes
  - Interactive programs
  - Background Processes
  - Student Processes
- Address the starvation and infinite blocking problems

Multi-Queue Scheduling: Example



Real Life Analogy

- Tasks (to-do list) for poor *Bob*
  - Class 1 priority (highest) : tasks given by his boss
    - Finish the project (50%)
  - Class 2 priority: tasks for his wife
    - Buy a valentine present (30%)
  - Class 3 priority (lowest): Bob's tasks
    - Watch TV (20%)

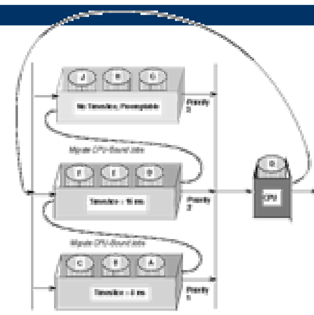
A Variation: Multi-level Feedback Algorithm

- Multi-Level Queue with priorities
- Processes **move** between queues
- Each queue represents jobs with similar CPU usage
- Jobs in a given queue are executed with a given time-slice
- Rational:
  - Once an I/O process completes an I/O request, it should have higher CPU priority.

Multi-level Feedback Algorithm (Details)

- Example: *Queue$_i$* has time-slice *t 2^i*
  - If a job in *Queue$_i$* doesn't block by end of time-slice, it is moved to *Queue$_{i}$* +1
- Lowest priority Queue is FIFO
- Problem:
  - Starvation: Aging may move Process to higher priority queue

Multi-level Feedback Algorithm: Example



Summary

- Shortest job first (SJF)
- Priority
- Round-robin
- Multi-Queue
- Multi-level Feedback

Reminder

- Next lecture: scheduling for
  - Real-time systems
  - Multiple processors
  - Threads
- Quiz1 due Friday 5pm
- MP1