Lecture 5: Scheduling (Feb 1, 2005) Yap

February 17, 2005

1 ADMIN

- Hw1 due today (but programming part has extension to Thursday Feb 3)
- Todays Reading: p.132-152
- Start Reading Chapter 3

2 Review

- Q: What is common to producers, consumers, barbers, philosophers, readers-writers?

  A: They are prototypes of the kinds of synchronization problems that must be solved in an OS.

- Q: Peterson's solution to mutual exclusion still has one potential defect. Explain.

  A: It does busy waiting. If processors have different priorities, we can still get a deadlock.

- Q: Tanenbaum explains that the Producer-Consumer Problem requires the solution of two kinds of IPC issues, which he calls "mutual exclusion" and "synchronization". Explain.

  A: Call these MUTEX and SYNCH problems.

  MUTEX: P and Q must not be in the critical section at the same time.

  SYNCH: P and Q must satisfy MUTEX for a critical section but in addition, they must visit them in a particular order. In our case, P must not over produce and Q must not over consume.

- Q: Name the 3 most important registers in a CPU?

  A: PC, PS, PSW Registers.

  REMARK: in a hardware interrupt, only these 3 registers are saved. The other registers need to be saved by this is dependent on the particular interrupt.

3 Scheduling – BACKGROUND

- Scheduling provides the "substrate" in which processes interact!

  This substrate is rather independent of how the processes interact (in IPC communication) or do not interact.

- In scheduling we need to distinguish between I/O bound and CPU-bound processes.

  As CPU's get faster, scheduling the former is getting more critical.

- There are 2 kinds of scheduling – preemptive and non-preemptive.

  Actually, preemption is usually relative to the system clock interrupts. At each clock interrupt, we must decide if we want to preempt.

- Three kinds of environments for scheduling:

  BATCH, INTERACTIVE, REALTIME.

  The mechanisms and goals needed are quite different.

- GOALS OF SCHEDULING:

  1. fairness (per process, per user, per thread)
  2. load balance (per computing unit)
  3. metric: throughput (maximize) – total CPU utilization, total # processes completed
  4. metric: turnaround (minimize)
  5. metric: responsiveness (interactive)
  6. metric: meeting deadlines (realtime)
  7. proportionality: subjective expectation that more difficult tasks should take more time.

4 Scheduling – BATCH Systems

- First come first serve
- Shortest jobs first
- Shortest remaining time first (PREEMPTIVE)
- 3 LEVEL SCHEDULING: 4 entities (input queue, RAM, CPU, Disk)

  1. Admission Scheduler: who goes from input queue to RAM
  2. CPU Scheduler: who goes from to RAM to CPU (and back)
  3. Memory Scheduler: who goes from to RAM to Disk (and back)

5 Scheduling – INTERACTIVE Systems

- Round Robin – each process has a quantum
  1. Advantage: no need to know the length of job. Disadvantage: process switching expensive
  2. TRADEOFF: context switching takes 1ms. Quantum is chosen to be 20-50 ms.
- Priority Scheduling – e.g., mail daemon has lower priority than video renderer.
  1. Priority classes
  2. Higher priority is scheduled first, and/or has more quantum.
  3. Combining Priority and Round Robin: round robin within priority classes, priority
- METHODS OF ASSIGNING priority:
  1. If a process uses fraction f of its quantum, its priority next time is $1/f$.
  2. QUANTUM QUEUES Q_i ($i = 0, 1, 2, \ldots$). Queue Q_i has 2^i quanta. Initially, all processes go to Q_0. When preempted from Q_i, goes into Q_{i+1}.
  3. AGING: How to estimate time to completion? If T is current estimate, and after the current run that takes time T', the next estimate is $(T + T')/2$.
  4. LOTTERY SCHEDULING:
     - Each process holds a number of lottery tix.
     - Scheduling is based on who owns the winning tix.
     - If you hold 20 of 100 outstanding tix, your chance is 1/20.
     - Cooperating processes can exchange tix (e.g., a client blocks and gives all his tix to server).

6 Scheduling – REALTIME Systems

- E.g., playing audio/video, monitoring physical processes in a hospital or nuclear plant, autopilot transportation.
- Difference from before: we now have **hard deadlines**. Note we can have "semi-hard" deadlines too.
- HOW TO ACHIEVE THIS?

  Divide program into small processes, each with predictable and known computing time.

- 2 kinds of events – **periodic** and **aperiodic** events
- Calculation of feasibility for periodic events: the ith event occurs every P_i seconds and requires C_i seconds of CPU time. Then feasible iff

$$\sum_i \frac{C_i}{P_i} \leq 1.$$

- Scheduler can be static or dynamic.