Monday

Monday, April 15, 2019 12:03 PM

Review SYCHRONIZATION CODE
Review BATCH COMPUTING

Bounded buffer

- Everything size n
- Producer
  - Down empty
  - Down mutex
  - Input item
  - Increase in by 1
  - Up mutex
  - Up full
- Consumer
  - Down full
  - Down mutex
  - Set item to out
  - Increase out
  - Up mutex
  - Up empty
- If, by any chance, empty and mutex lines switch
  - The code will block up
  - This might not always break

- Don’t use semaphores
- Use monitors instead

Dining Philosophers problem

- N philosophers around a table
  - All hungry and like to think
- N chopsticks available
  - 1 between each pair of philosophers
- Need 2 chopsticks to eat
- Alternate between eating and thinking
- Goal: coordinate use of chopsticks
- Example of Deadlock
  - Aborting a process by eliminating a "philosopher"
- Hold and wait
- NO preemption
- Circular wait
- Mutual exclusion

Scheduling

- What is Scheduling
  - Doing things in a certain order
  - Which process do I run next
  - Without loss of generality
  - Goals/Mechanism

Processes

- I/O bound vs CPU bound
- 1 second of cpu time is roughly 1 week of i/o bound
- Bursts of CPU usage alternate with periods of I/O wait
- Some processes are CPU-bound: they don't make many I/O requests
- Other processes are I/O-bound and make many kernel requests

When are processes scheduled

- At the time they enter the system
  - Common in batch systems
  - Two types of batch scheduling
    - Submission of a new job causes the scheduler to run
    - Scheduling only done when a job voluntarily gives up the CPU
- At relatively fixed intervals

    - Necessary for interactive systems
    - May also be used for batch systems
    - Scheduling algorithms at each interrupt, and picks the next process from the pool of "ready" processes

Scheduling goals

- All systems
    - Fairness
    - Enforcement
    - Balance
- Batch systems
    - Throughput
    - Turnaround time
    - CPU utilization
- Interactive systems
    - Response time
    - Proportionality
- Real-time systems
    - Meet deadlines
    - Predictability

Interactive vs Batch

- Batch
    - First Come First Served (FCFS)
    - Shortest Job First (SJF)
    - Shortest Remaining Time First (SRTF)
    - Priority (non-preemptive)
- Interactive
    - Round-Robin (RR)
    - Priority (preemptive)
    - Multi-level feedback queue
    - Lottery scheduling

Wednesday

Wednesday, April 17, 2019 11:50 AM

FCFS

- Do jobs in the order they arrive
- Very simple algorithm
- Problem
    - Long jobs delay every job after them
    - Many jobs may be delayed for a long time

SJF

- Do shortest jobs first
    - Short jobs first
    - Long jobs delay jobs after them
- Jobs sorted in increasing order of execution time
- SRTF: preemptive form of SJF
    - Re-evaluate when a new job is submitted
- Problem:
    - How does the scheduler know how long a job will take
    - Starvation
        - When shorter jobs are introduce, the longer jobs keep getting delayed

Three level scheduling

- Jobs held in input queue until moved into memory
    - Pick complementary jobs: small and large, CPU and I/O intensive
    - Jobs move into memory when admitted
- CPU scheduler pickes next job to run
- Memory scheduler picks some jobs from main memory and moves them to disk if insufficient memory space

Round Robin

- Scheduling interactive processes
    - Give each process a fixed time slot
        - Quantum
    - Rotate through ready processes
    - Each process makes some progress
- Whats a good quantum
    - Too short
        - Many process switches hurt efficiency
    - Too long
        - Poor response to interactive requests
    - Typical length = 10 to 100 ms
- Strict rotation

Priority Scheduling

- Assign a priority to each process
    - Ready process with the highest priority allowed to run
    - Running process may be interrupted after its quantum expires
- Priorities may be assigned dynamically

    - Reduced when a process uses CPU time
    - Increased when a process waits for IO
- Often processes grouped into multiple queues based on priority, and run round-robin per queue
- When quantum runs out, we punish the process like decreasing its priority
    - Policy = how we said things should happen
    - Mechanism - enforcement

Shortest process next
- Run the process that will finish the soonest
    - In interactive systems, job completion time is unknown
- Guess at completion time based on previous runs

Lottery Scheduling
- Give processes tickets for CPU time
    - More tickets = higher share of CPU
- Each quantum, pick a ticket at random
    - If there are n tickets, pick a number from 1 to n
        - Pseudo-random number is OK if it's a good RNG
    - Process holding the ticket gets to run for a quantum
    - This can be implemented efficiently without real tickets
        - Track range of tickets belonging to each process
- Over the long run, each process gets the CPU m/n of the time if the process has m of the n existing tickets
- Tickets can be transferred
    - Cooperating processes can exchange tickets
    - Clients can transfer tickets to a server so it can have a higher priority
    - Parent (shell) can transfer tickets to a child process

Scheduling in BSD4
- Quantum is 100 ms:
    - Longest that's OK for interactive scheduling
- Scheduler is based on multilevel feedback queues
    - Priorit is based on two things
        - Resource requirements

Calculate priority
- Thread priority is set by:
    - Pri = MIN + [estcpu/4] + 2 * nice
    - Values above MAX are set to MAX
    - MIN = 160 MAX =223
    - Nice is set by the user to manually lower thread priority
    - Estcpu is an estimate of the number of ready processes in the CPU when the calculation is made
        - Has a bit of memory so it doesn't change too quickly
        - Estcpu is updated each clock tick
    - Higher numbers indicate lower priority: threads with lowest priority values are scheduled first
- Thread priority is set every 40 ms
- Scheduling is more complex for multiprocesseors