Basics of Operating Systems

March 4, 2001

What is an Operating System?

- Many definitions - better to describe what OS's do
- Provide an abstraction over hardware
  - Standard interfaces to a variety of hardware
- Provide a user interface (not required, but most do)
- Provide an Application Programming Interface
- Manage resource sharing
  - CPU, disk drives, network, sound, video, keyboard, mouse, ...
- Key objectives depend on OS (e.g., Linux vs. Windows vs. PalmOS)

Communicating virtual machines

- When hardware was expensive, objective was to keep CPU fully utilized (batch processing)
- As hardware became less expensive, focus shifted towards human usability
  - E.g., share large servers among many simultaneous users
- Today: At least 1 physical CPU per user
  - Run many user programs on a CPU
- Complexity managed by creating an abstraction: communicating virtual machines

Communicating virtual machines, cont.

- Simple Model: Write programs as if it was the only one running on the machine
- Virtual machine implemented over a set of interfaces that abstract a computer's hardware
- Many virtual machines run concurrently on the same physical machine
  - Virtual machines share all the resources of the physical machine
- Operating system "Kernel" runs and manages the virtual machines

Concurrency: CPU

- Process (a virtual machine): a set of instructions + a state
  - The virtual machine executes each instruction against its state
  - A state includes all the data used by the execution stream
  - States implemented by a variety of hardware - registers, stack, RAM, hard drive
- Objective: Fair sharing, protection - processes cannot directly change each others' states (enforced by OS)
- Communication: done through shared memory or other resources

5

Processes

- Context switching
  - OS runs one process, then another, then another
  - In between, the OS saves the state of the current process, then loads the state of the next process
- When do you switch contexts?
  - Typically, OS designer decides policy: time-slicing, pre-emptive, priority-based, ...
  - Objectives: response time, efficient resource utilization

6

Processes, cont.

- Context switching is typically expensive!
  - Save process state (register, I/O state, stack, memory)
  - Flush cache if necessary (ouch)
  - Load new process state
  - Run new process
- How can we mitigate the expense of context switching?
  - Make tradeoffs - threads

7

Threads

- Like a process, except that some state is shared
  - Threads have their own registers and stack frames
  - Threads share memory
- Tradeoffs: Ease of programming + better performance vs. programming complexity + less protection
- Switching to another thread: save registers, find the right stack frame, load registers, run thread
- Typically used to service asynchronous events

8

Virtual Memory, Paging

- Each virtual machine should have its own independent memory space (protection)
- One solution: give each virtual machine ~(1/N) of the physical machine's memory
- We can do much better - give each virtual machine the ability to address a real memory space (e.g., 32 bits, 64 bits)
- VM, Paging - OS manages the way each virtual machine accesses main memory
  - Result - each virtual machine thinks it has its own, protected memory space

Virtual Memory, Paging, cont.

- Each virtual memory address consists of two pieces: a page number, and an offset into that page
- OS/Hardware uses a page table to translate a virtual memory address into a physical memory address
  - Virtual memory address = page number + offset
  - OS maintains a page table that maps virtual page numbers to physical page addresses
  - Translation starts by mapping the page number to a physical page address (i.e., page frame)
  - Then, add the offset to the physical page address to create a physical memory address

Virtual Memory, Paging, cont.

- Benefits from paging
  - Protection - a process's page numbers correspond only to physical memory addressable by the process
  - Allocation - physical pages allocated/deallocated as needed by processes (memory on demand)
- Virtual memory
  - When OS runs out of physical memory to allocate, use the disk to "swap out" the pages
  - Tradeoffs - more virtual memory vs. performance hit

Virtual Memory, Paging, cont.

- VM Optimizations:
  - swap out groups of pages at a time
  - don't let the number of free pages go below a threshold
  - set the page size equal to the disk block size
  - Pre-allocate the virtual memory file
  - Eliminate double-caching w/file system
  - … lots and lots at different levels (e.g., reduce working set by re-writing your code)

Memory Subsystem Performance Issues

- Problem: Thrashing - the OS spends most of its time swapping, and little time running processes
  - Potential Solutions - Install more physical memory, rewrite your code, find memory leaks, increase amount of time a process runs, ...
- Problem: Each virtual memory reference produces two physical memory reads
  - Popular Solution: Translation Lookaside Buffer (TLB) with some hardware support
  - Need a good design for your page tables

Page table design

- Linear page table: all entries in physical memory
  - Problem - doesn't scale well to 64 bit address space
- Two-level page table: Outer page entries point to inner pages that contain the mapping
  - e.g., 4 kbyte page, 32 bit entries: outer page stores 2^{10} entries. 32 bit virtual addresses now have 3 pieces: 10 bit outer, 10 bit inner, 12 bit offset
  - Supports sparse address spaces
  - TLB miss more expensive

Page table design, cont.

- Three-level page table - like two-level scheme
  - Better for 64-bit address spaces
- Inverted page table - one entry for each physical page that maps to a virtual page number
  - TLB miss - search inverted page table for virtual page number
  - Optimizations?

Page faults - example

- Page faults are expensive - how do we reduce the number of page faults?
- Example: Page string 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- What happens under a FIFO page replacement algorithm with enough memory for 3 physical page frames?
- What happens if we increase memory to 4 physical page frames? (Belady's anomaly)
- What about LRU?