CLOUD COMPUTING APPLICATIONS

Virtualization: Background

Prof. Reza Farivar

Brief History Lesson

- Single program computers
  - VERY early mainframes (1950s)
  - MS-DOS
    - Single user program gets access to everything the hardware has
    - The OS is really a thin wrapper around BIOS
    - No real notion of process
- Multi-user / Multi-tasking
  - Need to isolate programs
  - Need to isolate users
  - Notion of Process
    - "executing program and its context"

World-view from a process

- An image of the program's executable machine code
- Memory
  - virtual address space → paging → a VM page is brought into memory when the process attempts to use it → managed by the OS
  - Process-specific data (input and output)
  - Stack: temp data, e.g. function parameters, local variables, return addresses, function call stack, and saved variables
  - Heap to hold intermediate data during run time
- OS resource descriptors: e.g. file descriptors, data sources and sinks
- Security attributes: e.g. process owner and the process' set of permissions (allowable operations)
- Processor state (context)
  - Program Counter
  - Content of registers and physical memory addressing

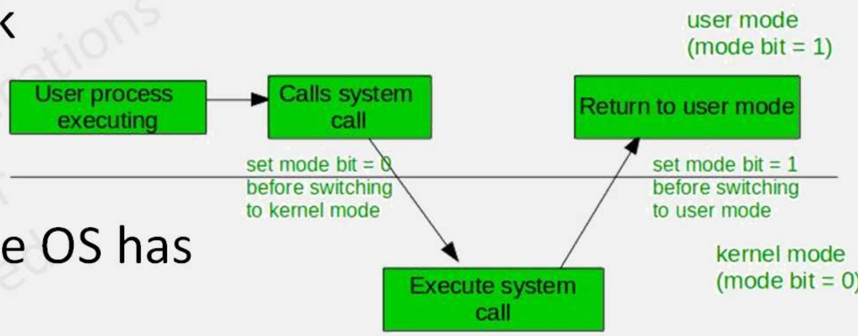
Process Isolation

- Need to isolate processes from each other
  - Virtualized, idealized, machine
  - A process is not capable of interacting with another process except through secure, kernel managed mechanisms
- User Processes should not be allowed to issue sensitive instructions
  - Things like loading memory mapping tables and accessing I/O devices.
- Normal applications better not use any of these instructions
- Imagine what would happen if a normal application like a word processor would suddenly be able to write to arbitrary memory locations, or get raw access to your hard drive.

Dual Mode Operations in OS

- The CPU and the Operating System work together to ensure process isolation

- To isolate processes from each other, the OS has two modes
  - User Mode
  - Kernel Mode

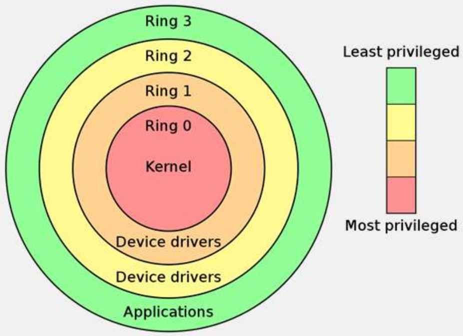


User and Kernel Modes

- User Mode
  - User processed operate in user mode
  - When the user application requests a service from the operating system, or an interrupt occurs, or a system call is made, there will be a transition from user to kernel mode to fulfill the requests
- Kernel Mode
  - When the system boots, hardware starts in kernel mode
  - privileged instructions which execute only in kernel mode
    - If user attempt to run privileged instruction in user mode then it will treat instruction as illegal and traps to OS
  - Example Privileged instruction: Input/Output management
  - Interrupt handling

CPU privilege protection

- When a privileged instruction is executed (or a safe instruction accesses a privileged resource), the CPU checks whether the process is allowed or not
  - Different mechanisms
  - x86 Example: Ring levels
  - Kernel mode code (OS, Device drivers, …) run in ring 0
  - User processes run in ring 3
- The CPU issues General Protection Fault (GPF) if a privileged instruction is executed in the wrong ring level



CPU + OS

- Certain operations are not allowed in user mode code
  - Read and write from a hardware device
  - Enabling/Disabling system interrupts
- Such operations only allowed in Kernel mode
- The task of enforcing this requirement is performed by the CPU
- Examples of privileged operations
  - HLT: Halt CPU till next interrupt
  - INVLPG: Invalidate a page entry in the translation look-aside buffer (TLB)
  - LIDT: Load Interrupt Descriptor Table
  - MOV CR registers: load or store control registers
    - In this case the MOV instruction (a non-privileged instruction on its own) is accessing a privileged register