Operating Systems

24. Virtualization

Paul Krzyzanowski

Rutgers University

Spring 2015

© 2013-2015 Paul Krzyzanowski

Virtualization inside the OS

- Memory virtualization
  - Process feels like it has its own address space
  - Created by MMU, configured by OS
- Storage virtualization
  - Logical view of disks “connected” to a machine
  - External pool of storage
- CPU/Machine virtualization
  - Each process feels like it has its own CPU
  - Created by OS preemption and scheduler

© 2013-2015 Paul Krzyzanowski

Storage Virtualization

© 2013-2015 Paul Krzyzanowski

Logical Volume Management

- Physical disk
  - Divided into one or more Physical Volumes
- Logical partitions – *Volume Groups*
  - Created by combining Physical Volumes
    - May span multiple physical disks
  - Can be resized
  - Each can hold a file system

© 2013-2015 Paul Krzyzanowski

Mapping Logical to Physical data

- Storage on physical volumes is divided into clusters (misnamed *extents*): fixed-size chunks

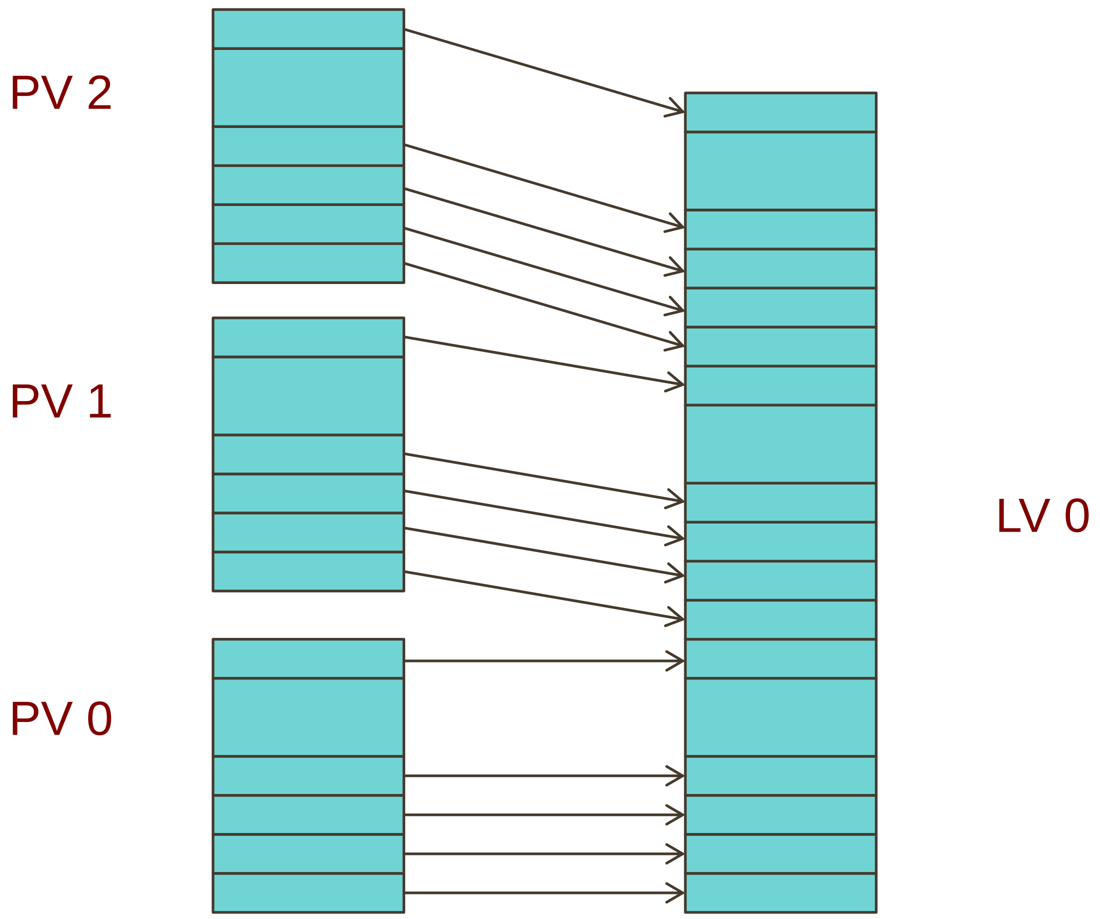
- Logical volume defined and managed by mapping of logical extents to physical extents

- Logical Volume Manager (LVM) takes care of this mapping

© 2013-2015 Paul Krzyzanowski

LVM Linear Mapping

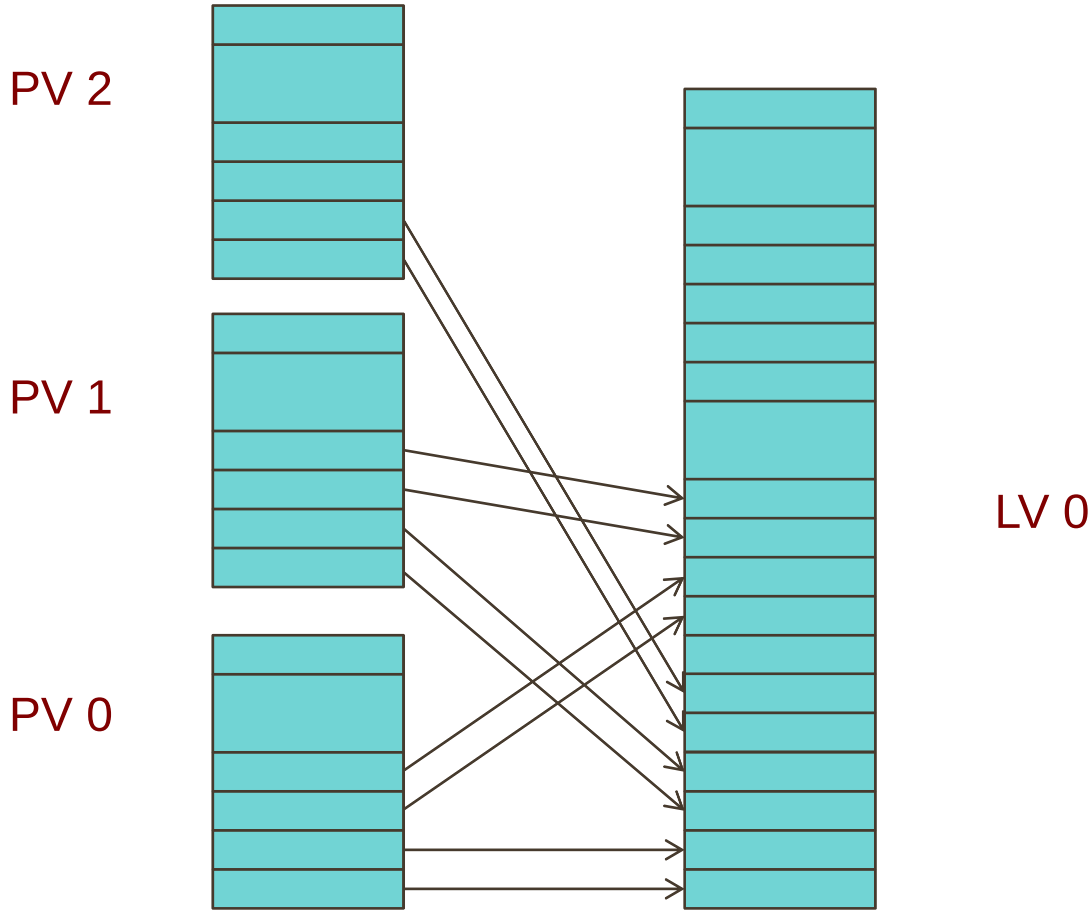
Concatenate multiple physical disks to create a larger disk



© 2013-2015 Paul Krzyzanowski

LVM Striped Mapping

Groups from alternate physical volumes mapped to a logical volume.
N physical extents per stripe. Improve bandwidth of file transfers



© 2013-2015 Paul Krzyzanowski

Advantages

- Logical disks can be resized while mounted
  - Some file systems (e.g., ext3 on Linux or NTFS) support dynamic resizing
- Data can be relocated from one disk to another
- Improved performance (through disk striping)
- Improved redundancy (disk mirroring)
- Snapshots
  - Save the state of the volume at some point in time.
  - Allow backups to proceed while the file system is being modified

© 2013-2015 Paul Krzyzanowski

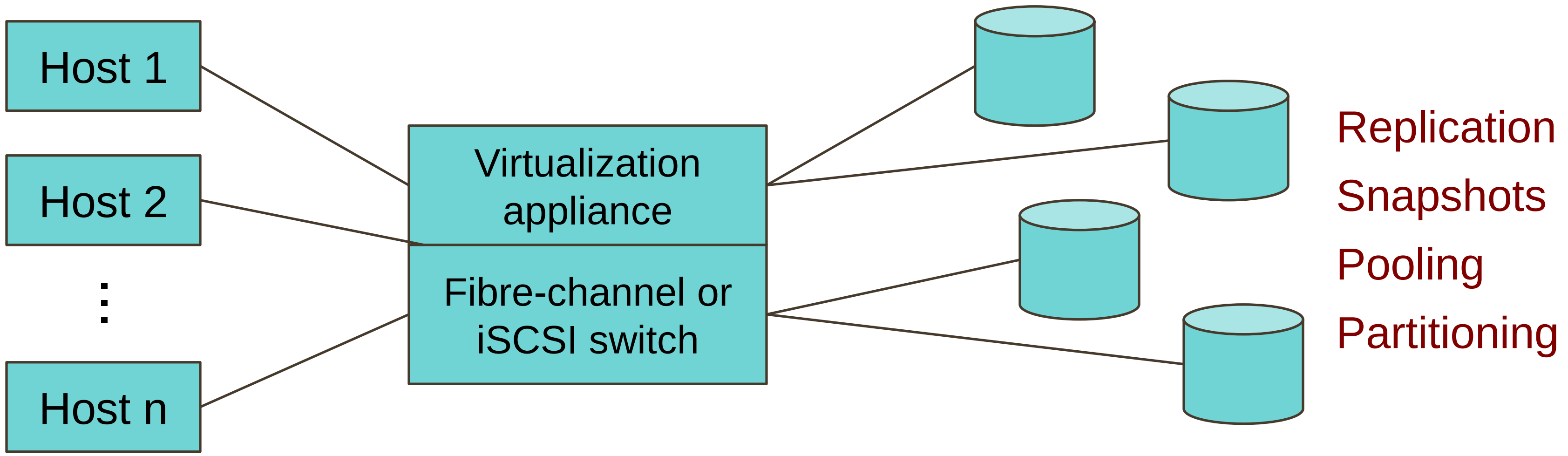
Storage Virtualization

- Dissociate knowledge of physical disks
  - The computer system does not manage physical disks
- Software between the computer and the disks manages the view of storage
- Virtualization software translates read-block / write-block requests for logical devices to read-block / write-block requests for physical devices

© 2013-2015 Paul Krzyzanowski

Storage Virtualization

- Logical view of disks "connected" to a machine
- Separate logical view from physical storage
- External pool of storage



© 2013-2015 Paul Krzyzanowski

Processor Virtualization

© 2013-2015 Paul Krzyzanowski

Virtual CPUs (sort of)

What time-sharing operating systems give us

- Each process feels like it has its own CPU & memory
  - But cannot execute privileged instructions (e.g., modify the MMU or the interval timer, halt the processor, access I/O)
- Illusion created by OS preemption, scheduler, and MMU
- User software has to "ask the OS" to do system-related functions.

© 2013-2015 Paul Krzyzanowski

Process Virtual Machines

- CPU interpreter running as a process
- Pseudo-machine with interpreted instructions
  - 1966: O-code for BCPL
  - 1973: P-code for Pascal
  - 1995: Java Virtual Machine (JIT compilation added)
  - 2002: Microsoft .NET CLR (pre-compilation)
  - 2003: QEMU (dynamic binary translation)
  - 2008: Dalvik VM for Android
  - 2014: Android Runtime (ART) – ahead of time compilation
- Advantage: run anywhere, sandboxing capability
- No ability to even pretend to access the system hardware
  - Just function calls to access system functions
  - Or “generic” hardware

© 2013-2015 Paul Krzyzanowski

Machine Virtualization

© 2013-2015 Paul Krzyzanowski

Machine Virtualization

Normally all hardware and I/O managed by one operating system

- Machine virtualization
  - Abstract (virtualize) control of hardware and I/O from the OS
  - Partition a physical computer to act like several real machines
    - Manipulate memory mappings
    - Set system timers
    - Access devices
  - Migrate an entire OS & its applications from one machine to another

- 1972: IBM System 370

© 2013-2015 Paul Krzyzanowski

Machine Virtualization

An OS is just a bunch of code!

- Privileged vs. unprivileged instructions
- Regular applications use unprivileged instructions
  - Easy to virtualize
- If regular applications execute privileged instructions, they trap
- VM catches the trap and emulates the instruction
  - Trap & Emulate

© 2013-2015 Paul Krzyzanowski

Hypervisor

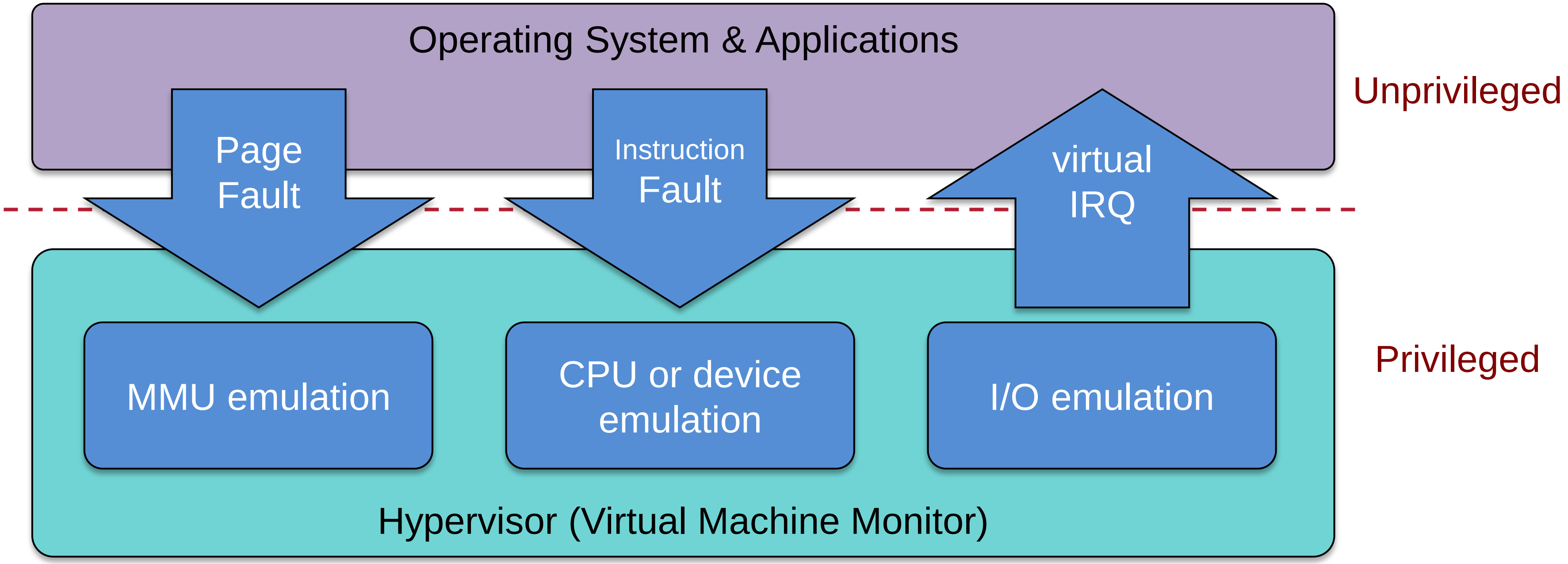
- Hypervisor: Program in charge of virtualization
  - Aka Virtual Machine Monitor
  - Provides the illusion that the OS has full access to the hardware
  - Arbitrates access to physical resources
  - Presents a set of virtual device interfaces to each host

© 2013-2015 Paul Krzyzanowski

Hypervisor

Application or Guest OS runs until:

- Privileged instruction traps
- System interrupts
- Exceptions (page faults)
- Explicit call: VMCALL (Intel) or VMMCALL (AMD)



© 2013-2015 Paul Krzyzanowski

Intel & ARM Didn't Make VM Easy

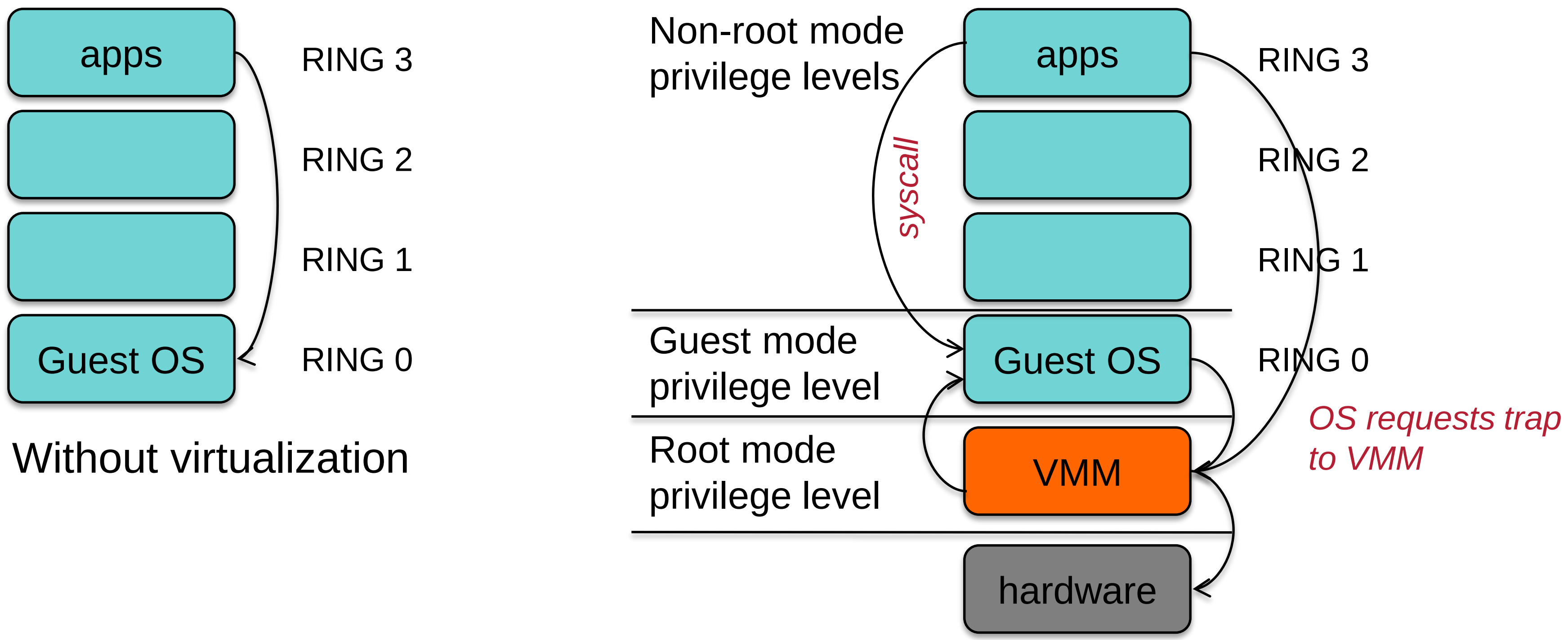
- Intel/AMD systems prior to Core 2 Duo (2006) did not support trapping privileged instructions
- Most ARM architectures also did not trap on certain privileged instructions
  - Hardware support added in Cortex-A15 (ARMv7 Virtualization Extension): 2011
- Two approaches
  - Binary translation (BT)
    - Scan instruction stream on the fly (when page is loaded) and replace privileged instructions with instructions that work with the virtual hardware (VMware approach)
  - Paravirtualization
    - Don't use non-virtualizable instructions (Xen approach)
    - Invoke hypervisor calls explicitly

© 2013-2015 Paul Krzyzanowski

Hardware support for virtualization

Root mode (Intel example)

– Layer of execution more privileged than the kernel



© 2013-2015 Paul Krzyzanowski

Architectural Support

- Intel Virtual Technology
- AMD Opteron

- Guest mode execution: can run privileged instructions directly
  - E.g., a system call does not need to go to the VM
  - Certain privileged instructions are intercepted as VM exits to the VMM
  - Exceptions, faults, and external interrupts are intercepted as VM exits
  - Virtualized exceptions/faults are injected as VM entries

© 2013-2015 Paul Krzyzanowski

CPU Architectural Support

- Setup
  - Turn VM support on/off
  - Configure what controls VM exits
  - Processor state
    - Saved & restored in guest & host areas
- VM Entry: go from hypervisor to VM
  - Load state from guest area
- VM Exit
  - VM-exit information contains cause of exit
  - Processor state saved in guest area
  - Processor state loaded from host area

© 2013-2015 Paul Krzyzanowski

Two Approaches to Running VMs

1. Native VM (hypervisor model)
2. Hosted VM

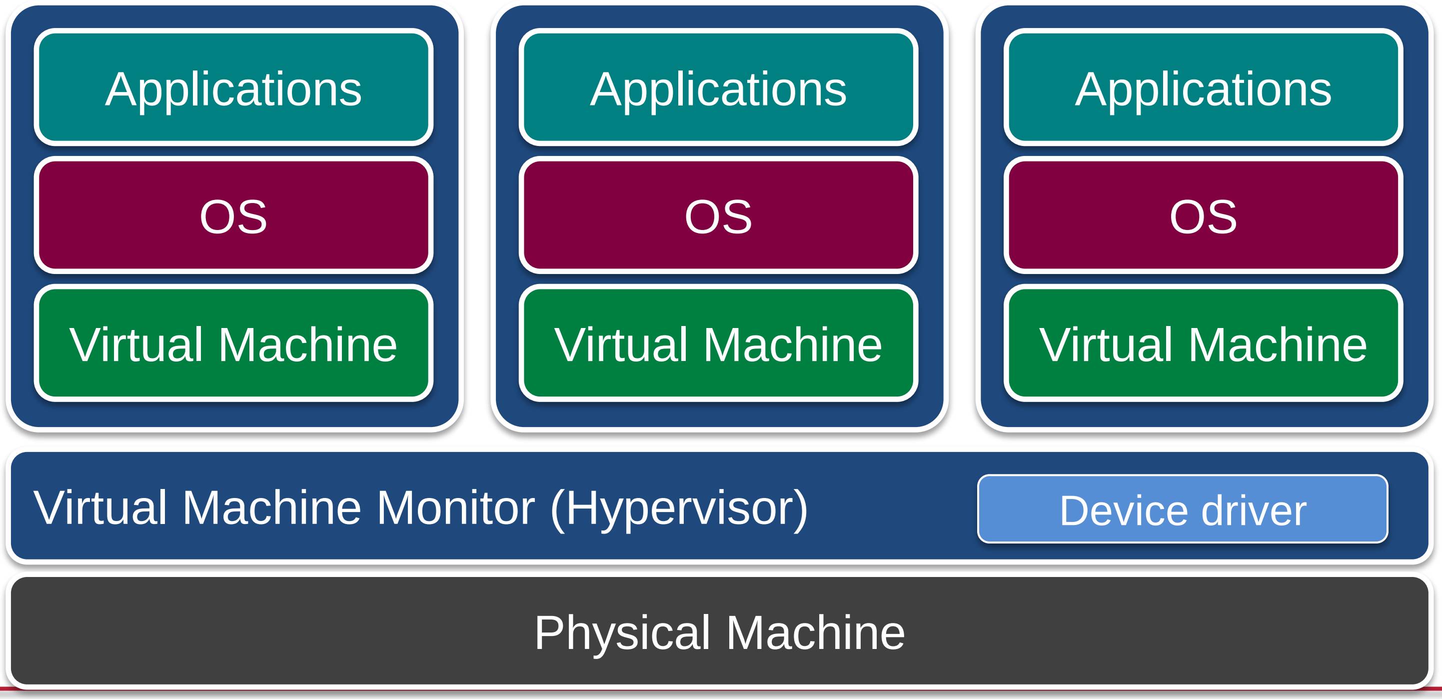
© 2013-2015 Paul Krzyzanowski

Native Virtual Machine

Example:
VMware ESX

Native VM (or *Type 1* or *Bare Metal*)

– No primary OS
– Hypervisor is in charge of access to the devices and scheduling
– OS runs in “kernel mode” but does not run with full privileges



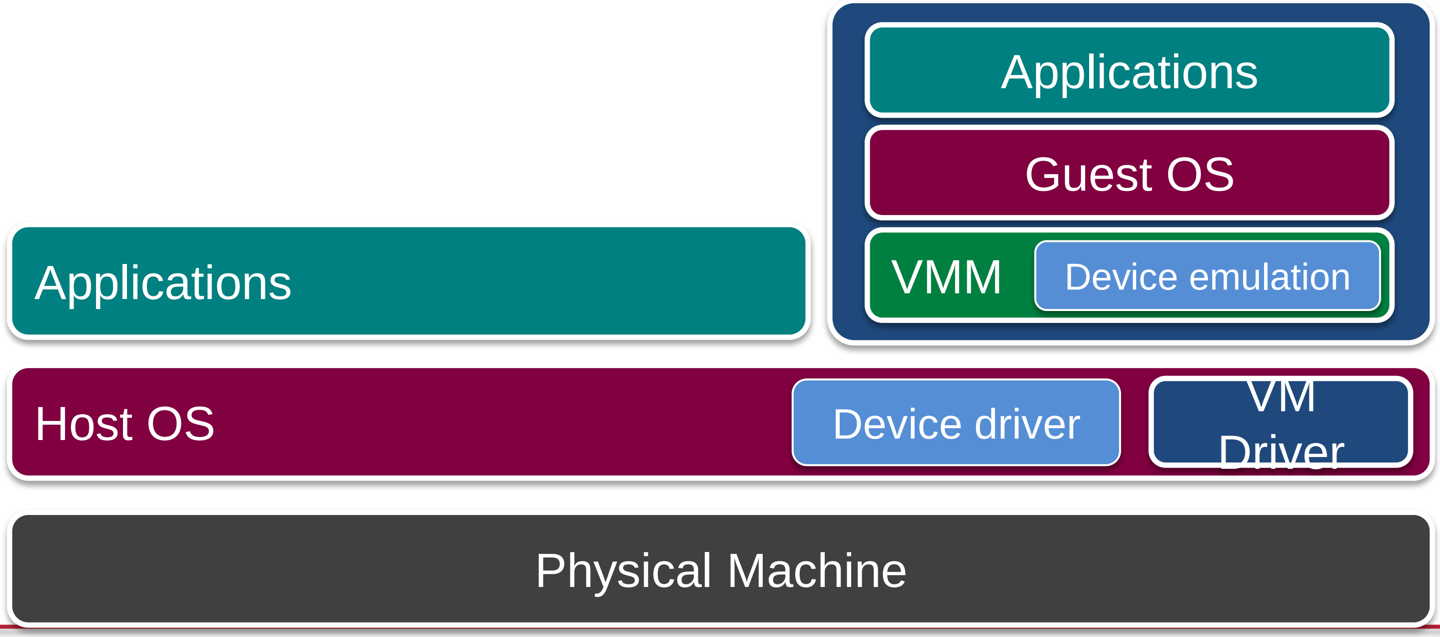
© 2013-2015 Paul Krzyzanowski

Hosted Virtual Machine

Example: VMware Workstation

Hosted VM

- VMM runs without special privileges
- Primary OS responsible for access to the raw machine
  - Lets you use all the drivers available for that primary OS
- Guest operating systems run under a VMM
- VMM invoked by host OS
  - Serves as a proxy to the host OS for access to devices



Virtualizing Memory

- Similar to OS-based virtual memory
  - An OS sees a contiguous address space
  - But it is not necessarily tied to physical memory
- Need to virtualize MMU
  - Two levels of translation: Shadow page tables
    - Host allocates virtual memory for guest
      - Guest treats that as physical memory
    - Guest OS cannot access real page tables
      - Access attempts are trapped and emulated
    - VMM maps guest "physical memory" settings to actual memory
  - Second-level address translation (SLAT) = Nested page tables
    - Hardware support in MMU – similar to multilevel page tables
      - Performance enhancement over shadow page tables
    - A guest's physical address is treated as a virtual address

© 2013-2015 Paul Krzyzanowski

Scheduling VMs

- Each VM competes for a physical CPU
  - Typically # VMs > # CPUs

- VMs need to get scheduled
  - Each VM gets a time slice
    - Often round robin scheduler – or minor variations
  - Allocate CPU to a single-CPU VM
  - Allocate multiple CPUs to multi-CPU VMs: co-scheduling
    - Strict co-scheduler: VM with two virtual CPUs gets two real CPUs
    - Relaxed co-scheduler: if two CPUs are not available, use one
  - CPU affinity: try to run the VM on the same CPU
- VM scheduler controls the level of multiprogramming of VMs

Virtualizing Drivers & Events

- Operating systems cannot interact directly with I/O devices
- Device drivers
  - VMM has to multiplex physical devices & create network bridges
  - Virtualize network interfaces (e.g., MAC addresses)
  - Guest OS gets device drivers that interface to an abstract device implementation provided by the VMM
- VMM gets all system interrupts and exceptions
  - Needs to figure out which OS gets a simulated interrupt
  - Simulate those events on the guest OS

© 2013-2015 Paul Krzyzanowski

Live Migration

- Select alternate host (B)
  - Mirror block devices (for file systems)
  - Initialize VM on B
- Initialize
  - Copy dirty pages to host B iteratively
- To migrate
  - Suspend VM on A
  - Send ARP message to redirect traffic to B
  - Synchronize remaining VM state to B
  - Release state on A

© 2013-2015 Paul Krzyzanowski

Some Popular VM Platforms

- Native VMs
  - Microsoft Hyper-V
  - VMWare ESX Server
  - IBM z/VM (mainframe)
  - XenServer
    - Ran under an OS and provides virtual containers for running other operating systems. Runs a subset of x86. Routes all hardware accesses to the host OS.
    - Non-modified OS support for processors that support x86 virtualization
  - Sun xVM Server

- Hosted VMs
  - VMWare Workstation
  - VirtualBox
  - Parallels

© 2013-2015 Paul Krzyzanowski

Security Threats

- Hypervisor-based rootkits
- A system with no virtualization software installed but with hardware-assisted virtualization can have a hypervisor-based rootkit installed.
- Rootkit runs at a higher privilege level than the OS.
  - It's possible to write it in a way that the kernel will have a limited ability to detect it.

© 2013-2015 Paul Krzyzanowski

OS-Level Virtualization

- Not full machine virtualization
- Multiple instances of the same operating system
  - Each has its own environment
    - Process list, mount table, file descriptors, virtual network interface
- Advantage: low overhead: no overhead to system calls
- Examples:
  - *Linux VServer, Solaris Containers, FreeBSD Jails*
  - *Symantec Software Virtualization Solution*
    (originally *Altris Software Virtualization Services*)
    - Windows registry & directory tweaking
    - Allows multiple instances of applications to be installed

© 2013-2015 Paul Krzyzanowski

BSD Jails

- Directory subtree
  Root of namespace. Process cannot escape from this subtree

- Hostname
  Hostname that will be used within the jail

- IP address
  IP address used for a process within the jail

- Command
  Command that will be run within the jail

© 2013-2015 Paul Krzyzanowski

The End

© 2013-2015 Paul Krzyzanowski