CSE 4/521 Introduction to Operating Systems

Lecture 26 – Virtual Machines

(Overview, History, Benefits and Features, Building Blocks, Types of Virtual Machines and Their Implementations)

Summer 2018

Overview

Objective:

- To explore the history and benefits of virtual machines
- To discuss the various virtual machine technologies
- To describe the methods used to implement virtualization
- To show the most common hardware features that support virtualization

- Overview
- History
- Benefits and Features
- Building Blocks
- Types of Virtual Machines and Their Implementations

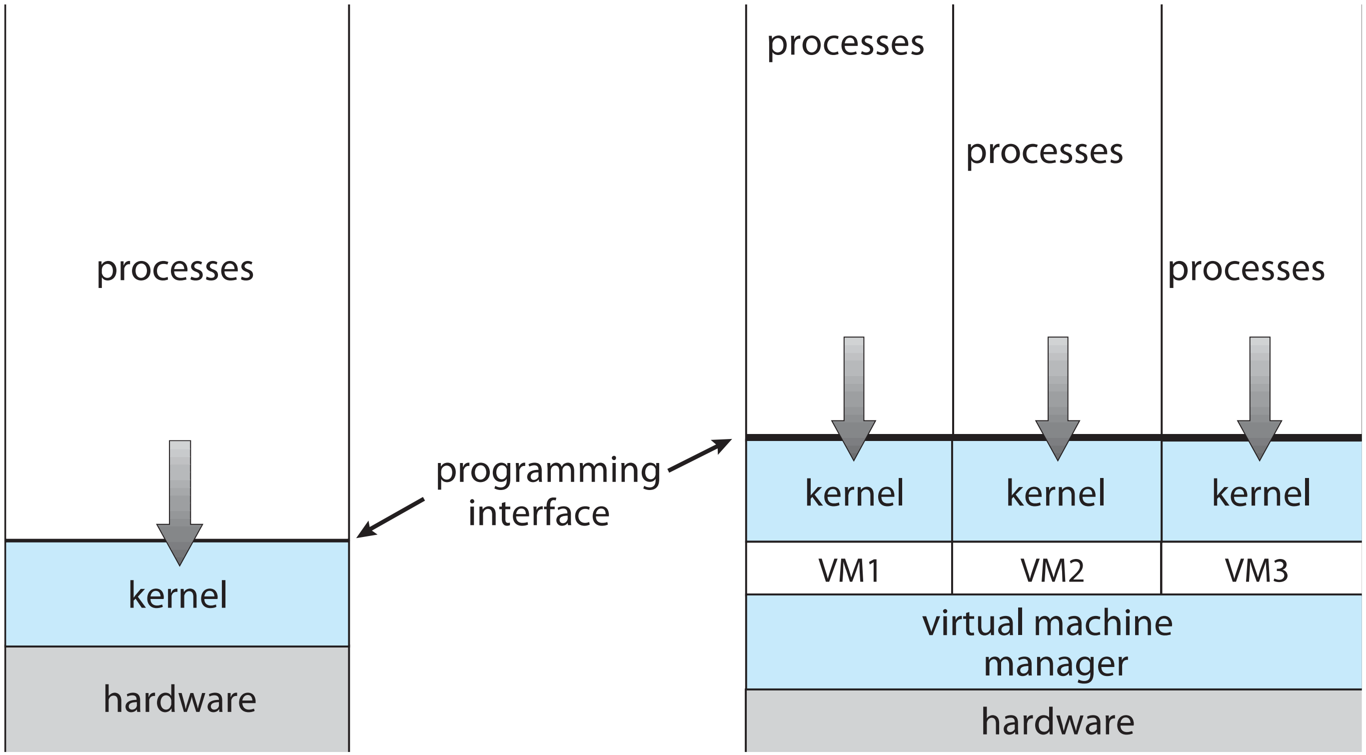
Overview

- Overview
- History
- Benefits and Features
- Building Blocks
- Types of Virtual Machines and Their Implementations

Overview

- Fundamental idea – abstract hardware of a single computer into several different execution environments
  - Similar to layered approach
  - But layer creates virtual system (virtual machine, or VM) on which operation systems or applications can run
- Several components
  - Host – underlying hardware system
  - Virtual machine manager (VMM) or hypervisor – creates and runs virtual machines by providing interface that is ***identical*** to the host
  - Guest – process provided with virtual copy of the host
    - Usually an operating system

Overview : System Models



Non-virtual machine

Virtual machine

Overview: Implementation of VMMs

- Type 0 hypervisors - Hardware-based solutions that provide support for virtual machine creation and management via firmware
  - IBM LPARs and Oracle LDOMs are examples
- Type 1 hypervisors - Operating-system-like software built to provide virtualization
  - Including VMware ESX, Joyent SmartOS, and Citrix XenServer
- Type 2 hypervisors - Applications that run on standard operating systems but provide VMM features to guest operating systems
  - Including VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox

Overview: Implementation of VMMs

- Other variations include:
  - Paravirtualization - Guest operating system is modified to work in cooperation with the VMM to optimize performance
  - Programming-environment virtualization - VMMs do not virtualize real hardware but instead create an optimized virtual system
    - Used by Oracle Java and Microsoft.Net
  - Emulators – Allow applications written for one hardware environment to run on a very different hardware environment, such as a different type of CPU
  - Application containment - Not virtualization at all but rather provides virtualization-like features by segregating applications from the operating system, making them more secure, manageable
    - Including Oracle Solaris Zones, BSD Jails, and IBM AIX WPARs

Overview

- Overview
- History
- Benefits and Features
- Building Blocks
- Types of Virtual Machines and Their Implementations

History

- First appeared in IBM mainframes in 1972
- Allowed multiple users to share a batch-oriented system
- Formal definition of virtualization helped move it beyond IBM
    1. A VMM provides an environment for programs that is essentially identical to the original machine
    2. Programs running within that environment show only minor performance decreases
    3. The VMM is in complete control of system resources

Overview

- Overview
- History
- Benefits and Features
- Building Blocks
- Types of Virtual Machines and Their Implementations

Benefits and Features

- Host system protected from VMs, VMs protected from each other
- Freeze, suspend, running VM
  - Then can move or copy somewhere else and resume
  - Snapshot of a given state, able to restore back to that state
  - Clone by creating copy and running both original and copy
- Run multiple, different OSes on a single machine
  - Consolidation, app dev, …

Benefits and Features

- Templating – create an OS + application VM, provide it to customers, use it to create multiple instances of that combination
- Live migration – move a running VM from one host to another!
  - No interruption of user access
- All those features taken together -> cloud computing
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops

Overview

- Overview
- History
- Benefits and Features
- Building Blocks
- Types of Virtual Machines and Their Implementations

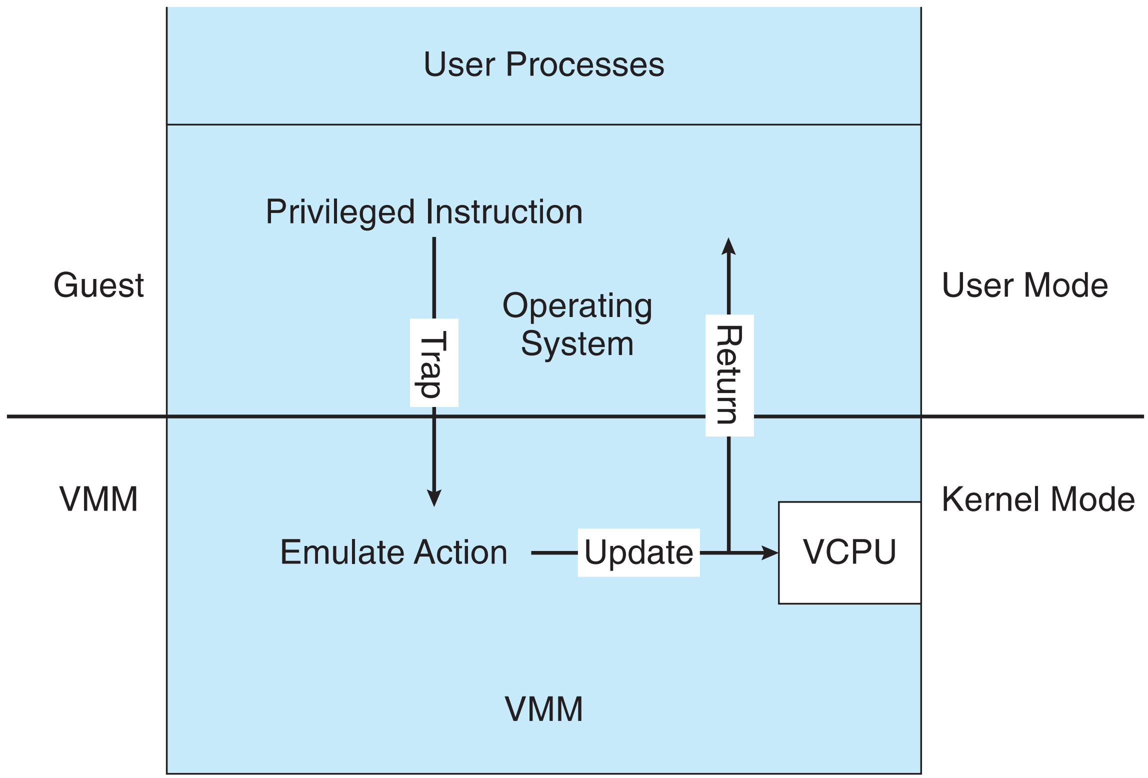
Building Blocks

- Generally difficult to provide an exact duplicate of underlying machine
  - Especially if only dual-mode operation available on CPU
  - Most VMMs implement virtual CPU (VCPU) to represent state of CPU per guest as guest believes it to be
    - When guest context switched onto CPU by VMM, information from VCPU loaded and stored

Building Blocks: Trap and Emulate

- Dual mode CPU means guest executes in user mode
  - Kernel runs in kernel mode
  - Not safe to let guest kernel run in kernel mode too
  - So VM needs two modes – virtual user mode and virtual kernel mode
    - Both of which run in real user mode
  - Actions in guest that usually cause switch to kernel mode must cause switch to virtual kernel mode

Building Blocks: Trap-and-Emulate Implementation



Building Blocks: Binary Translation

- Some CPUs don’t have clean separation between privileged and nonprivileged instructions
  - Earlier Intel x86 CPUs are among them
    - Earliest Intel CPU designed for a calculator
  - Backward compatibility means difficult to improve
  - Consider Intel x86 **`popf`** instruction
    - Loads CPU flags register from contents of the stack
    - If CPU in privileged mode -> all flags replaced
    - If CPU in user mode -> on some flags replaced
      - No trap is generated

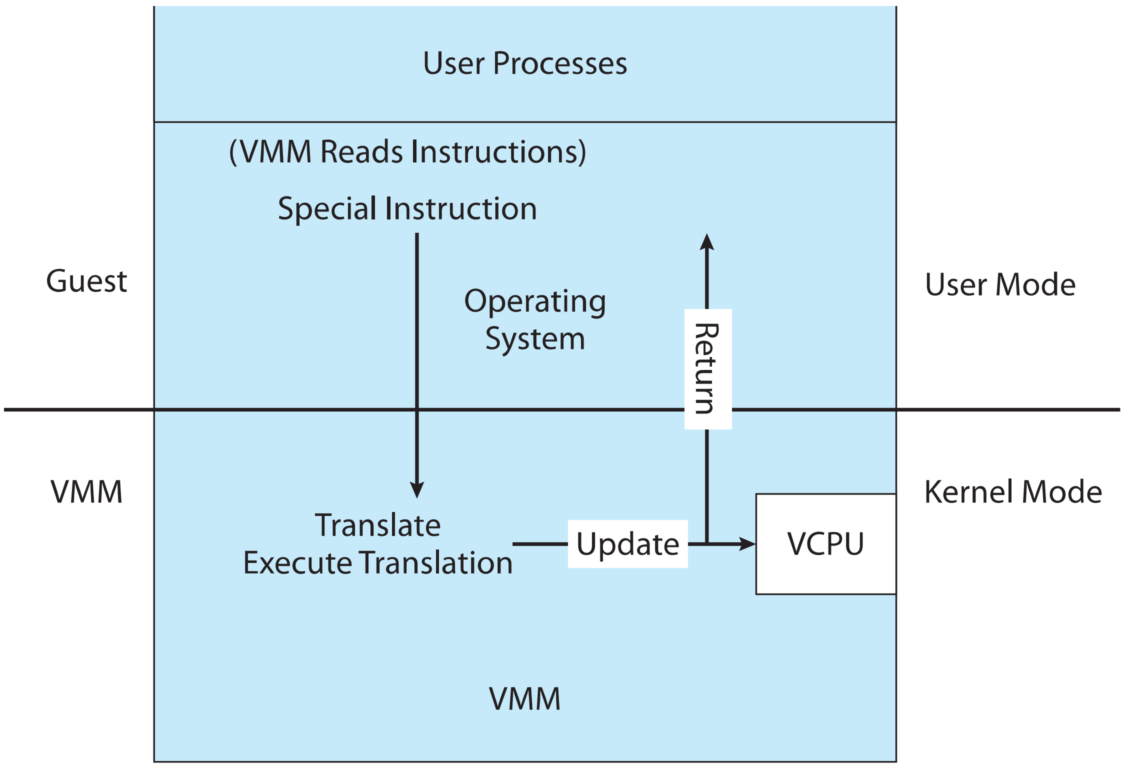
Building Blocks: Binary Translation

- Basics are simple, but implementation very complex
- If guest VCPU is in user mode, guest can run instructions natively
- If guest VCPU in kernel mode (guest believes it is in kernel mode)
  1. VMM examines every instruction guest is about to execute by reading a few instructions ahead of program counter
  2. Non-special-instructions run natively
  3. Special instructions translated into new set of instructions that perform equivalent task (for example changing the flags in the VCPU)

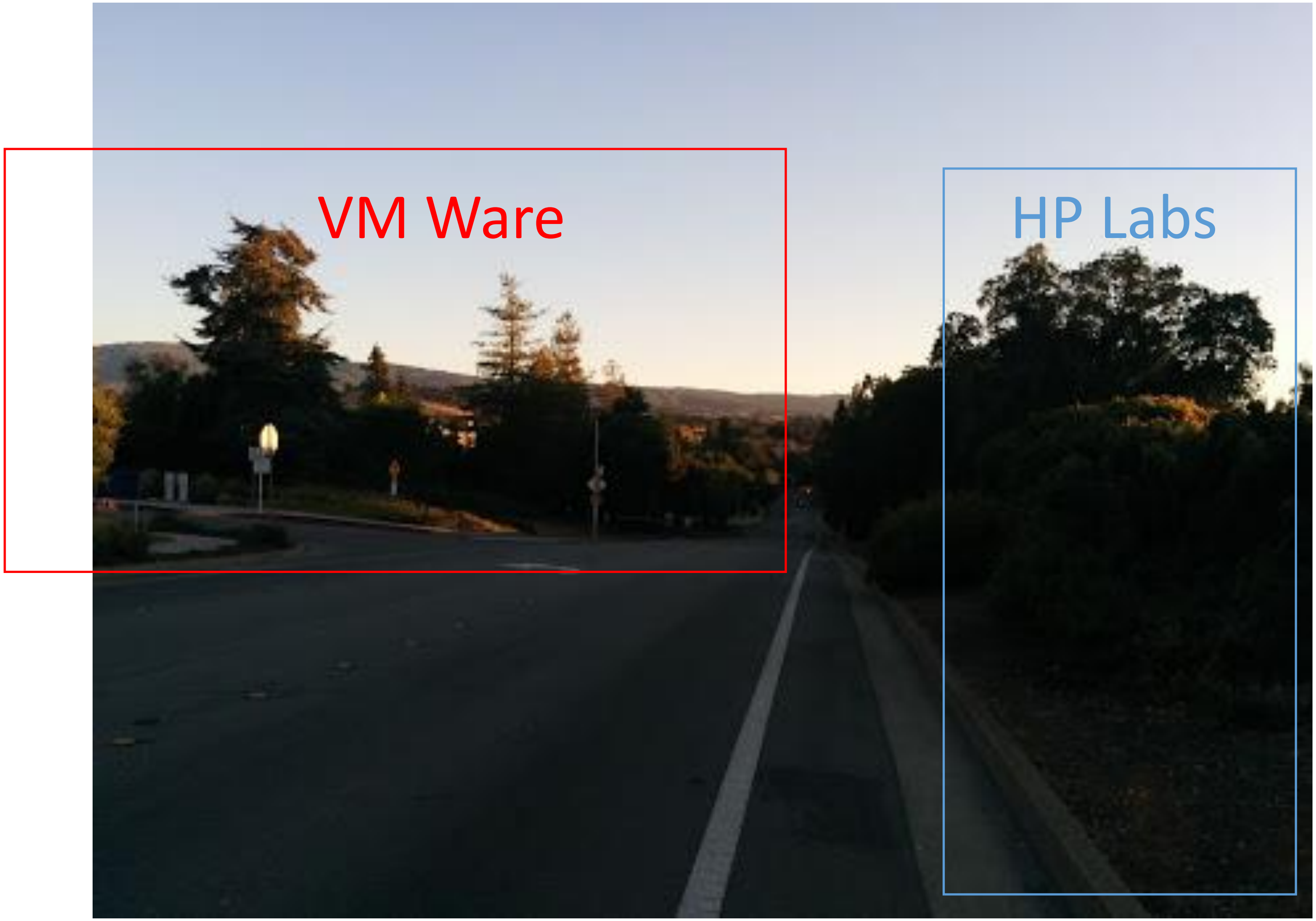
Building Blocks: Binary Translation

- Performance of this method would be poor without optimizations
  - Products like VMware use caching
    - Translate once, and when guest executes code containing special instruction cached translation used instead of translating again
    - Testing showed booting Windows XP as guest caused 950,000 translations, at 3 microseconds each, or 3 second (5 %) slowdown over native

Building Blocks: Binary Translation Implementation



VMWare Campus



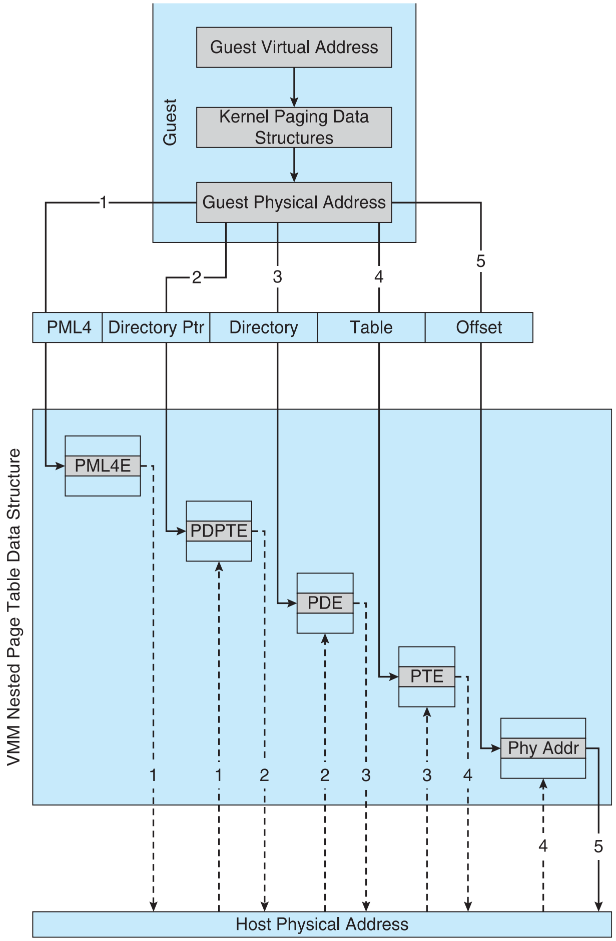
Building Blocks: Hardware Assitance

- All virtualization needs some HW support
- More support -> more feature rich, stable, better performance of guests
- Intel added new VT-x instructions in 2005 and AMD the AMD-V instructions in 2006
  - CPUs with these instructions remove need for binary translation
  - VMM can enable host mode, define characteristics of each guest VM, switch to guest mode and guest(s) on CPU(s)
- HW support for Nested Page Tables, DMA.

Building Blocks: Nested Page Tables

- Common method (for trap-and-emulate and binary translation) is **nested page tables (NPTs)**
  - Each guest maintains page tables to translate virtual to physical addresses
  - VMM maintains per guest NPTs to represent guest's page-table state
  - When guest on CPU -> VMM makes that guest's NPTs the active system page tables
  - Guest tries to change page table -> VMM makes equivalent change to NPTs and its own page tables

Building Blocks: Nested Page Tables



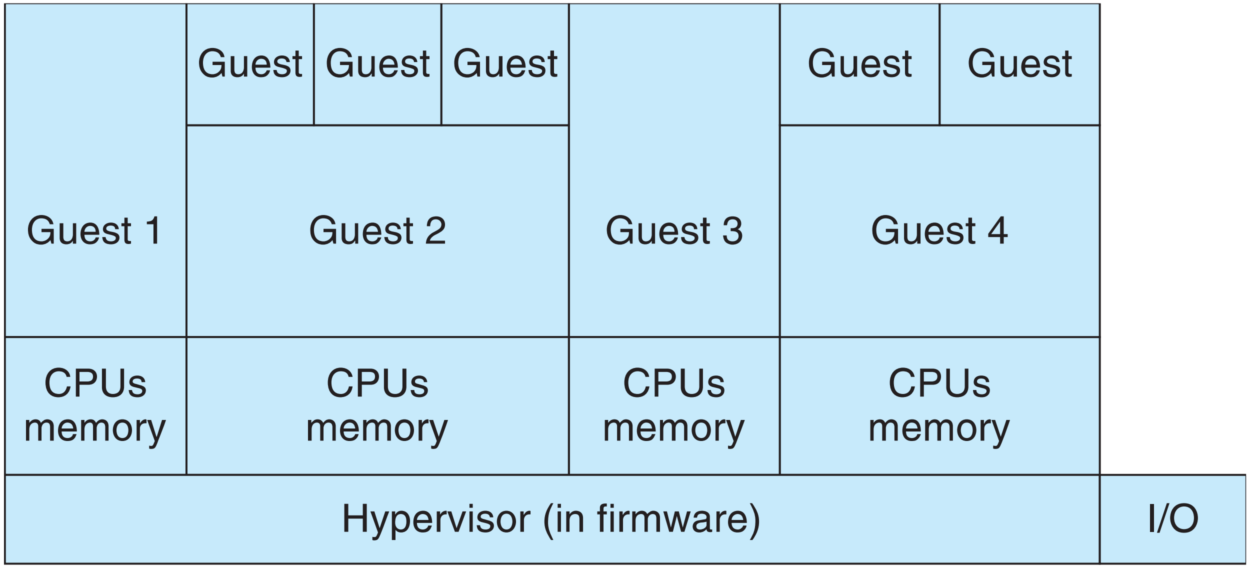
Overview

- Overview
- History
- Benefits and Features
- Building Blocks
- Types of Virtual Machines and Their Implementations

Types of VM: Type 0 Hypervisor

- Old idea, under many names by HW manufacturers
  - Smaller feature set than other types
  - Each guest has dedicated HW
- I/O a challenge as difficult to have enough devices, controllers to dedicate to each guest
- Sometimes VMM implements a control partition running daemons that other guests communicate with for shared I/O
- Can provide virtualization-within-virtualization (guest itself can be a VMM with guests
  - Other types have difficulty doing this

Types of VM: Type 0 Hypervisor



Types of VM: Type 1 Hypervisor

- Commonly found in company datacenters
  - Consolidation of multiple OSes and apps onto less HW
  - Move guests between systems to balance performance
  - Snapshots and cloning
- Special purpose operating systems that run natively on HW
  - Rather than providing system call interface, create run and manage guest OSes
  - Guests generally don’t know they are running in a VM
  - Implement device drivers for host HW because no other component can

Types of VM: Type 2 Hypervisor

- Less interesting from an OS perspective
  - Very little OS involvement in virtualization
  - VMM is simply another process, run and managed by host
    - Even the host doesn't know they are a VMM running guests
  - Tend to have poorer overall performance because can't take advantage of some HW features
  - But also a benefit because require no changes to host OS
    - Run multiple guests, all on standard host OS such as Windows, Linux, MacOS

Types of VM: Paravirtualization

- Does not fit the definition of virtualization – VMM not presenting an exact duplication of underlying hardware
  - But still useful!
  - VMM provides services that guest must be modified to use
  - Leads to increased performance
  - Less needed as hardware support for VMs grows
- Xen, leader in paravirtualized space, adds several techniques
  - For example, clean and simple device abstractions
    - Efficient I/O
    - Good communication between guest and VMM about device I/O
    - Each device has circular buffer shared by guest and VMM via shared memory

Types of VM: Paravirtualization

- Guest uses hypercall (call to hypervisor) when page-table changes needed
- Paravirtualization allowed virtualization of older x86 CPUs (and others) without binary translation
- Guest had to be modified to use run on paravirtualized VMM
- But on modern CPUs Xen no longer requires guest modification -> no longer paravirtualization

Credits for slides

Silberschatz, Galvin and Gagne