CS370 Operating Systems

Colorado State University
Yashwant K Malaiya
Fall 2016 Lecture 41

Virtualization

Slides based on
- Various sources

FAQ

- Does deleting a hard link also delete a file?
  - Hard link: same inode
  - An Inode has a counter that counts the number of links
  - If there are two links, you delete one, file still present with its inode with just one link
  - Will be deleted if the count is 0.
- In a soft link, what is filename3 pointing to

  ln -s /dirA/filenname1 /dirB/filename3
  - Filename3 points to filenname1
- Virtualization: If two guest OS run concurrently, then are there context switches within the hypervisor as well as the individual OSs?
  - Implementation may be implementation dependent. The guest OSs may do their own scheduling, but hypervisor manages partitioning the resources among the OSs.

Notes

- Poster Session Dec 9 10-11:30
  - See Assignments page, Canvas
  - You need to put it up and take it down later
  - At least one person should be available to provide an explanation
  - Prepare a 1 minute presentation
- Look at other posters. Need to review 3 posters, including one specified.
- Submit poster file: Dec 9, 5 PM, Reviews and Final paper Dec 13.

Terminology

Virtualization

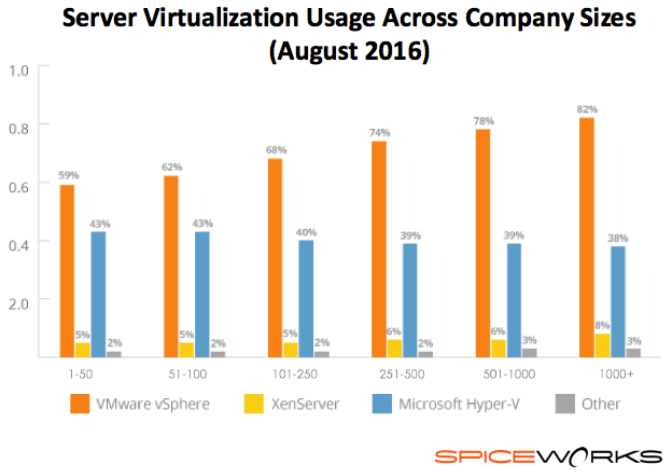
- Hypervisor based
  - Full virtualization
  - Para virtualization
  - Host OS virtualization
- Container system
- Environment virtualization
  - Java virtual machine, Dalvic virtual machine
- Software simulation of hardware/ISA
  - Android JDK
  - SoftPC etc.
- Emulation using microcode

User mode and Kernel (supervisor) mode

- Special instructions:
- Depending on whether it is executed in kernel/user mode
  - "Sensitive instructions"
- Some instructions cause a trap when executed in user-mode
  - "Privileged instructions"
- A machine is virtualizable only if sensitive instructions are a subset of privileged instructions
  - Intel's 386 did not always do that. Several sensitive 386 instructions were ignored if executed in user mode.
- Fixed in 2005
  - Intel CPUs: VT (Virtualization Technology)
  - AMD CPUs: SVM (Secure Virtual Machine)

Implementation of VMMs

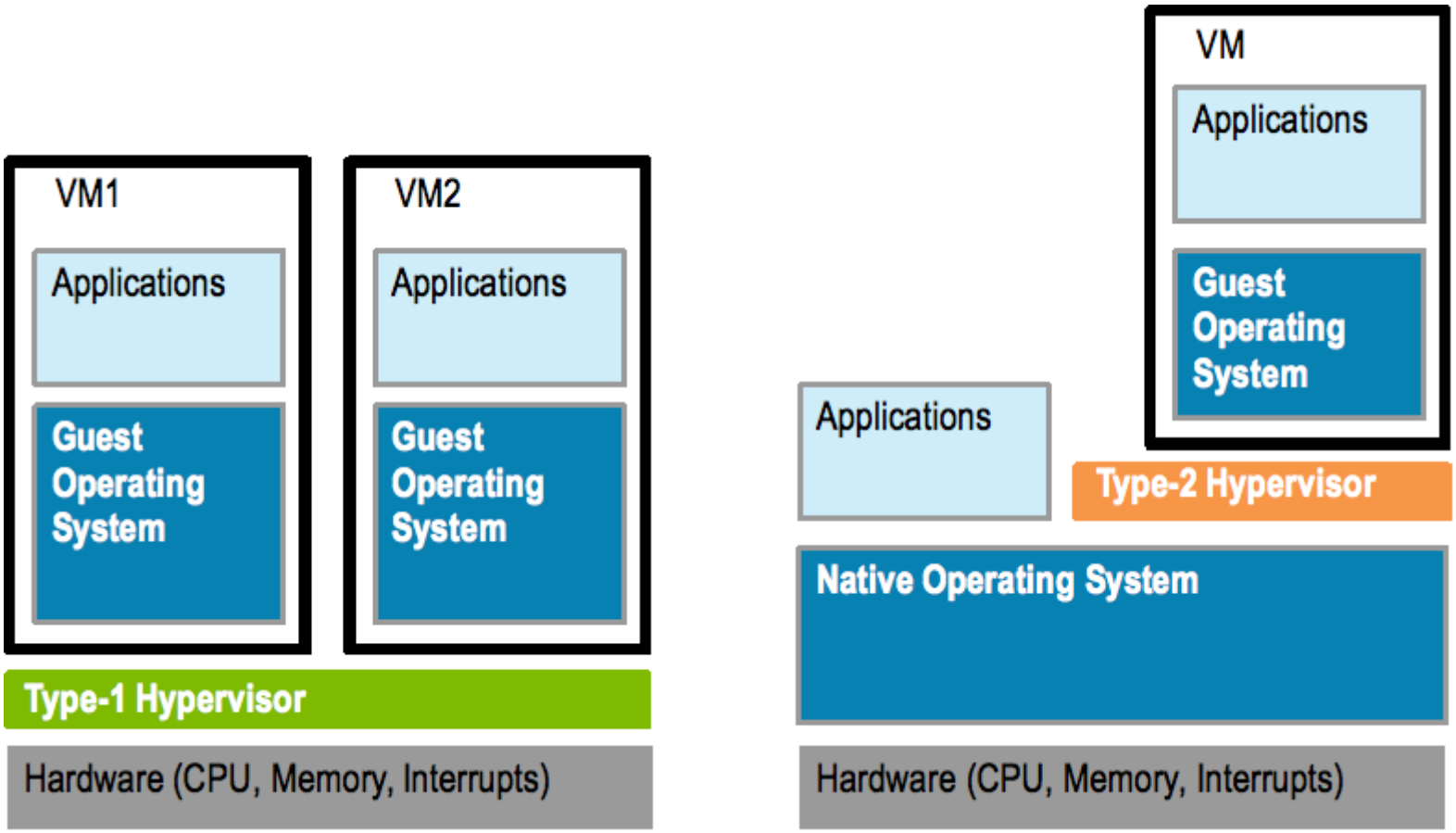
- **Type 1 hypervisors** - Operating-system-like software built to provide virtualization. Runs on 'bare metal".
  - Including VMware ESX, Joyent SmartOS, and Citrix XenServer
- Also includes general-purpose operating systems that provide standard functions as well as VMM functions
  - Including Microsoft Windows Server with HyperV and RedHat Linux with KVM
- **Type 2 hypervisors** - Applications that run on standard operating systems but provide VMM features to guest operating systems
  - Includeing VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox



All 3 are Type 1

http://www.virtualizationsoftware.com/top-5-enterprise-type-1-hypervisors/

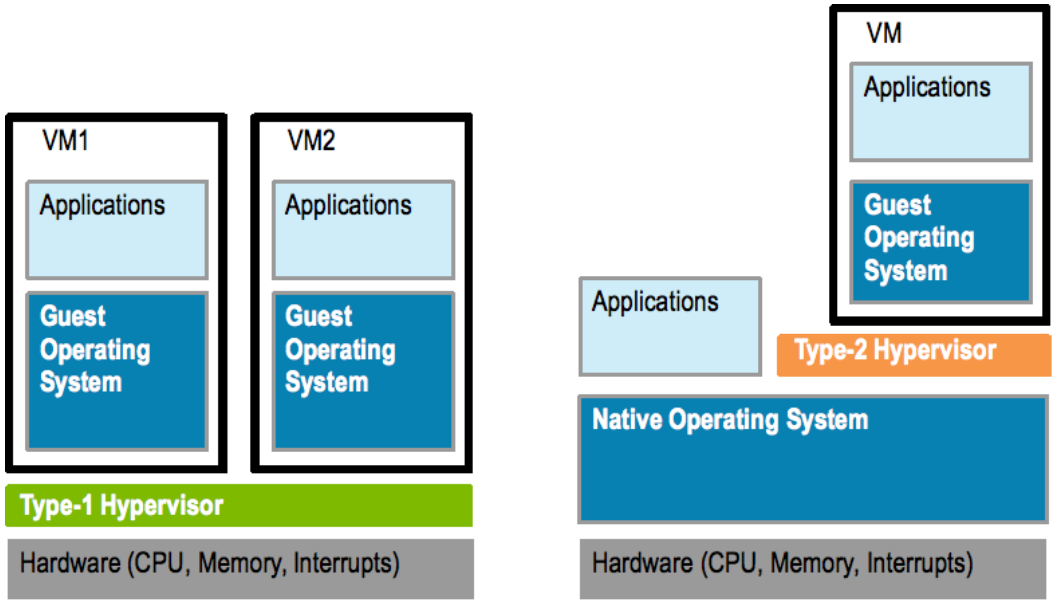
Implementation of VMMs



Type 1: Bare metal Type 2: Hosted

https://microkerneldude.files.wordpress.com/2012/01/type1-vs-2.png

Implementation of VMMs



What problems do you see?

- What mode does hypervisor run in? Guest OSs?
- Are Guest OSs aware of hypervisor?
- How is memory managed?
- How do we know what is the best choice?

Terms

- Guest Operating System
  - The OS running on top of the hypervisor
- Host Operating System
  - For a type 2 hypervisor: the OS that runs on the hardware ¨executions

Virtual Machine (VM) as a software construct

- Each VM is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports
- Once the VM is created it can be activated on like a physical server, loaded with an operating system and software solutions, and used just like a physical server
- But unlike a physical server, VM only sees the resources it has been configured with, not all of the resources of the physical host itself
- The hypervisor facilitates the translation and I/O between the virtual machine and the physical server.

Virtual Machine (VM) as a set of files

- Configuration file describes the attributes of the virtual machine containing
  - server definition,
  - how many virtual processors (vCPUs)
  - how much RAM is allocated,
  - which I/O devices the VM has access to,
  - how many network interface cards (NICs) are in the virtual server
  - the storage that the VM can access
- When a virtual machine is instantiated, additional files are created for logging, for memory paging etc.
- Copying a VM produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself

Virtualization benefits

- Run multiple, OSes on a single machine
  - **Consolidation**, app dev, …
- Security: Host system protected from VMs, VMs protected from each other
  - Sharing though shared file system volume, network communication
- Freeze, **suspend**, running VM
  - Then can move or copy somewhere else and **resume**
    - **Live migration**
  - Snapshot of a given state, able to restore back to that state
  - **Clone** by creating copy and running both original and copy
- Hence – cloud computing

Building Block – Trap and Emulate

- VM needs two modes: both in real user mode
  - virtual user mode and virtual kernel mode
- When Guest OS attempts to execute a privileged instruction, what happens?
  - Causes a trap
  - VMM gains control, analyzes error, executes operation as attempted by guest
  - Returns control to guest in user mode
  - Known as **trap-and-emulate**
- This was the technique used for implementing floating point instructions in CPUs without floating point coprocessor

Sensitive instructions

- Some CPUs didn't have clean separation between privileged and non-privileged instructions
  - Sensitive instructions
    - Consider Intel x86 `popf` instruction
    - If CPU in privileged mode -> all flags replaced
    - If CPU in user mode -> on some flags replaced
      - No trap is generated
- Binary translation (complex) solves the problem
  1. If guest VCPU is in user mode, guest can run instructions natively
  2. 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. Special instructions translated into new set of instructions that perform equivalent task (for example changing the flags in the VCPU)
  3. Cached translations can reduce overhead

Type 1 Hypervisors

- Guest OSs believe they are running on bare metal, are unaware of hypervisor
  - are not modified
  - Better performance
- Choice for data centers
    - Consolidation of multiple OSes and apps onto less HW
    - Move guests between systems to balance performance
    - Snapshots and cloning
- Create run and manage guest OSes
  - Run in kernel mode
  - Implement device drivers
  - Also provide other traditional OS services like CPU and memory management
- Examples: VMWare esx (dedicated) , Windows with Hyper-V (includes OS)

Type 2 Hypervisors

- Run on top of host OS
- VMM is simply a process, managed by host OS
  - host doesn't know they are a VMM running guests
- poorer overall performance because can't take advantage of some HW features
- Host OS is just a regular one
  - Individuals could have Type 2 hypervisor on native host (perhaps windows), run one or more guests (perhaps Linux, MacOS)

Full vs Para-virtualization

- Full virtualization: Guest OS is unaware of the hypervisor. It thinks it is running on bare metal.
- Para-virtualization: Guest OS is modified and optimized. It sees underlying hypervisor.
  - Introduced and developed by Xen
    - Modifications needed: Linux 1.36%, XP: 0.04% of code base
  - Does not need as much hardware support
  - allowed virtualization of older x86 CPUs without binary translation
  - Not used by Xen on newer processors

CPU Scheduling

- One or more virtual CPUs (vCPUs) per guest
  - Can be adjusted throughout life of VM
- When enough CPUs for all guests
  - VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
- Usually not enough CPUs (CPU overcommitment)
  - VMM can use scheduling algorithms to allocate vCPUs
  - Some add fairness aspect

CPU Scheduling (cont)

- Oversubscription of CPUs means guests may get CPU cycles they expect
  - Time-of-day clocks may be incorrect
  - Some VMMs provide application to run in each guest to fix time-of-day

Evaluation form

Memory Management

Memory mapping:

- On a bare metal machine: Each process has its own virtual address space. OS uses page table/TLB to map Virtual page number (VPN) to Physical page number (PPN) (physical memory is shared). Each process has its own page table/TLB.

  VPN -> PPN

Memory Management

Memory mapping:

- On a bare metal machine:
  - VPN -> PPN
- VMM: Real physical memory (*machine memory*) is shared by the OSs. Need to map PPN of each VM to MPN (Shadow page table)

  PPN ->MPN
- Where is this done?
  - In Full virtualization?
  - In Para virtualization?

Memory Management

- VMM: Real physical memory (*machine memory*) is shared by the OSs. Need to map PPN of each VM to MPN (Shadow page table)

  PPN ->MPN

- Where is this done?
  - In Full virtualization? Has to be done by hypervisor. Guest OS knows nothing about MPN.
  - In Para virtualization? May be done by guest OS. It knows about hardware. Commands to VMM are "hypercalls"
- Full virtualization: PT/TLB updates are trapped to VMM. It needs to do VPN->PPN ->MPN. It can do VPN->MPN directly (VMware ESX)

Handling memory oversubscription

Oversubscription solutions:

- Deduplication by VMM determining if same page loaded more than once, memory mapping the same page into multiple guests
- Double-paging, the guest page table indicates a page is in a physical frame but the VMM moves some of those to disk.
- Install a pseudo-device driver in each guest (it looks like a device driver to the guest kernel but really just adds kernel-mode code to the guest)
  - Balloon memory manager communicates with VMM and is told to allocate or deallocate memory to decrease or increase physical memory use of guest, causing guest OS to free or have more memory available.

Live Migration

Running guest can be moved between systems, without interrupting user access to the guest or its apps

- for resource management,
- maintenance downtime windows, etc

- Migration from source VMM to target VMM
  - Needs to migrate all pages gradually, without interrupting execution (details in next slide)
  - Eventually source VMM freezes guest, sends vCPU's final state, sends other state details, and tells target to start running the guest
  - Once target acknowledges that guest running, source terminates guest

Live Migration

- Migration from source VMM to target VMM
  - Source VMM establishes a connection with the target VMM
  - Target creates a new guest by creating a new VCPU, etc
  - Source sends all read-only memory pages to target
  - Source starts sending all read-write pages to the target, marking them as clean
    - repeats, as during that step some pages were modified by the guest and are now dirty.
  - Source VMM freezes guest, sends VCPU's final state, other state details, final dirty pages, and tells target to start running the guest
    - Once target acknowledges that guest running, source terminates guest

VIRTUAL APPLIANCES: “shrink-wrapped” virtual machines

- Developer can construct a virtual machine with
  - required OS, compiler, libraries, and application code
  - Freeze them as a unit … ready to run
- Customers get a complete working package
- Virtual appliances: “shrink-wrapped” virtual machines
- Amazon’s EC2 cloud offers many pre-packaged virtual appliances examples of *Software as a service*