

# Virtualization

CS 5309 – Advanced Operating Systems

Christian McArthur

# Types of Virtualization

- System
- Storage (SAN)
- Networks (VPN, VLAN, Hypersockets)
- Application (cloud computing)
- Computation (grid computing)

# System Virtualization

- Allow multiple virtual systems to run on a single physical system.
- Introduced by IBM in 1960s for partitioning mainframe environments.

# Business Benefits

- Growth in number of servers requires increases in:
  - Physical space.
  - Power needs & costs.
  - Cooling needs & costs.
- Virtualization reduces the number of servers, therefore the costs of running a server room/farm.

# Business Benefits

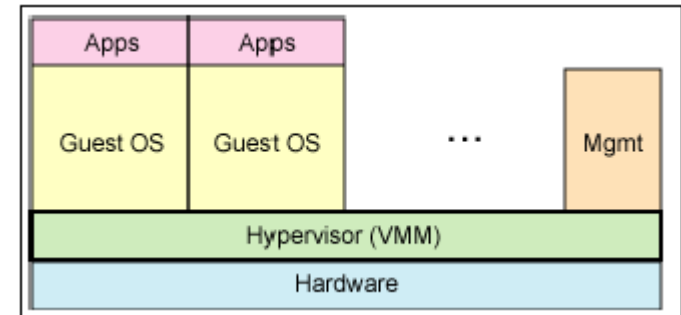
- Infrastructure simplification & improved manageability.
  - All virtual OSs can run on the same “hardware”.
  - Standard OS configuration, drivers, etc.

# Business Benefits

- Server infrastructure is the largest & fastest growing component of IT spending.
- Reducing costs of running servers & improving manageability of servers allows for a reducing in total cost of ownership.

# Full Virtualization

- VMM sits between the guest OS and the HW.



- Allows the OS to run unmodified.
- Sensitive instructions performed by OS need to be handled by the VMM.
- x86 Architecture could not support full virtualization until recently.

# Formal Requirements for VMM

- Privileged Instructions:
  - Processor traps when in user mode.
  - Will not trap when in system mode.
- Sensitive Instructions:
  - Changes or depends on resource configuration.
- Create a VMM if sensitive instructions are a subset of privileged instructions

Popek and Goldberg Virtualization Requirements; “Formal requirements for virtualizable third generation architectures”, Communications of the ACM, July 1974

# x86 Issues with Full Virtualization

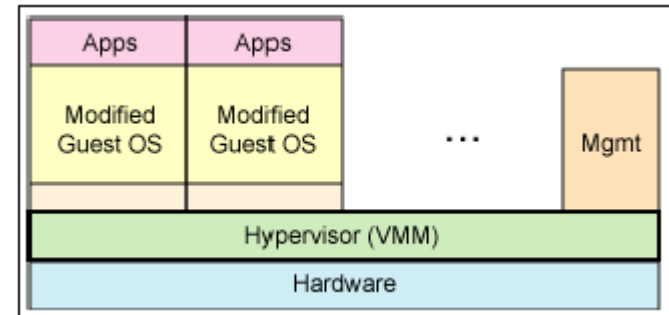
- When sensitive instruction executed in user mode, x86 would ignore it (silent fail).
- New versions of x86 architecture from Intel and AMD now support trapping sensitive instructions.

# Virtualization on old x86 arch.

- VMM would perform binary translation.
- Analyzes basic blocks of instructions.
- Detects if sensitive instruction is being called.
- VMM catches these instructions & performs trap to system mode.

# Paravirtualization

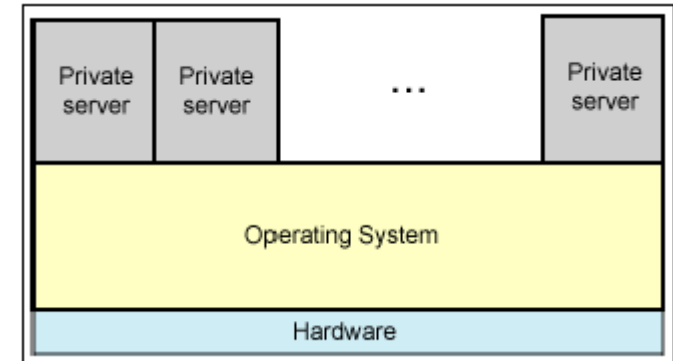
- Guest OS is modified to run on top of the VMM.



- Guest OS will make calls to the VMM to execute sensitive instructions.
- VMM doesn't need to do extra work to catch sensitive instructions.

# OS Level Virtualization

- A single OS runs on top of the hardware.



- Multiple virtual servers run instances of the same OS code.

# Virtualization & Mainframes

- PR/SM is full virtualization manager.
- Runs in firmware on all modern mainframes.
- Supports up to 60 logical partitions, each with its own OS.
- Separate address spaces & storage protection keys prevent one partition from modifying another's memory.

# Virtualization and Mainframes

- z/VM is a full virtualization operating system.
- Possible to run “hundreds to thousands of Linux servers” with one z/VM host.
- Each OS has its own virtual memory and up to 64 virtual CPUs.
- z/VM performs translation from virtual to actual memory & CPUs.

# VMWare

- Founded in 1998 by students from Berkeley and Stanford.
- First group to create virtual machine managers for x86 systems.
- VMWare workstation depends on Host OS.
- ESX Server used built-in “VMkernel”.

# VMWare Binary Translation

- “Just-in-time binary translation”
- Each block of code run by the guest OS is analyzed.
- Translates from x86 instruction set to “safe to execute instructions”
- Replace privileged instructions with VM specific operations
- Code blocks placed into translation cache

# VMWare: 32 & 64 bit issues

- VMkernel with ESX Server provides separation of kernel & VM.
  - Able to run 64bit VMs on a 32bit server.
  - Each VM is run on a separate VMM.
- Current VMWare software makes only little use of new x86 features designed for virtualization.

# VMWare: Memory Management

- x86 MMU aids operating systems with mapping physical memory to virtual memory.
- VMM provides an additional layer to map virtual physical memory to actual physical memory. (Shadow Page Tables)

# Shadow Page Tables

- SPT placed on top of the guest OS' top-level page table.
- SPT marked as read-only.
- When guest attempts to write, page fault occurs.
- VMWare gets control from the page fault trap & makes proper memory adjustments.

# Newer x86 MMU & Virtualization

- Shadow Page Tables incur overhead.
- AMD & Intel provide MMU virtualization aids in newer architectures.
- Physical MMU can point to two sets of page tables (VM table & guest table)
- Can result in higher overhead for some misses due to walking two trees.

# Xen

- Open source virtualization software
- Developed at Cambridge starting in 2003
- Specifically for x86 architectures
- Designed to support up to 100 VMs running Windows XP and Linux
- Uses paravirtualization; the OS must be modified to run with Xen.

# Xen: CPU issues

- OS runs at same privilege level as regular applications.
- Hypervisor provides virtual privilege levels
- OS & applications run in different address spaces.
- System calls and sensitive instructions are handled through Xen.

# Xen: Memory Access

- Guest OS has read access to HW page tables.
- OS allocates & initializes page tables directly with HW, then registers it with VMM.
- Writes & updates to page table then performed by VMM.

# Xen: I/O

- Full Virtualisation provides emulated I/O interfaces for Guest OS.
- Xen provides abstractions of the I/O.
- A 'standard' OS with needed drivers exist.
- Guest I/O operations reflected to this OS.
- Data transferred using shared memory

# Porting OS to Xen

- OS must be modified to run on Xen.

OS subsection	# lines	
	Linux	XP
Architecture-independent	78	1299
Virtual network driver	484	–
Virtual block-device driver	1070	–
Xen-specific (non-driver)	1363	3321
<b>Total</b>	<b>2995</b>	<b>4620</b>
(Portion of total x86 code base	<b>1.36%</b>	<b>0.04%</b> )

**Table 2: The simplicity of porting commodity OSes to Xen. The cost metric is the number of lines of reasonably commented and formatted code which are modified or added compared with the original x86 code base (excluding device drivers).**

# Scheduling CPU Resources

- Finite physical CPU resources; potentially infinite virtual CPU needs.
- Concerns over how to prevent a VM with high CPU needs from taking too many CPU cycles from less demanding VMs.
- Research in scheduling algorithms to be used by the VMM.

# Credit Scheduling

- Default scheduler with Xen.
- Every OS (including Host) assigned a weight and cap.
- Weight: Priority for CPU time, default 256.
- Cap: Percentage representing total CPU time it can consume. 100 = 1 CPU, 50 =  $\frac{1}{2}$  CPU.

# Credit Scheduling Algorithm

- Each CPU has local queue of runnable VCPU sorted by priority.
- As VCPU run, it consumes its credits.
- When a VCPU finishes its work, the CPU grabs the next VCPU off the local queue.
- If local queue is empty, it looks to other CPU queues for work.

# Hybrid Scheduling Framework

- Divides VMs into two types.
- High-throughput is default, normal, type.
  - Performance may deteriorate when running parallel or multi-threaded programs.
- Concurrent VMs set to reduce cost of synchronization

# Borrowed-Virtual Time Scheduling

- Designed for time sensitive thread sched.
- Thread with earliest effective virtual time is dispatched.
- Latency sensitive threads allowed to 'warp' back in time.
- Borrows virtual time from its future CPU allocation.

# Research Uses

- Lots of use of VMs in security research:
  - Virus collection & analysis
  - Malware & intrusion detection
  - Honey pots

# Temporal Search to Detect Timebomb Malware

- Some malware wait until a certain time before executing their payload or to receive additional instructions from C&C.
- Behavior based on absolute (clock) time and relative time
- Automated approach to detecting timers in use by malware. Speed up all aspects of 'time'.

# Out-of-the-box malware detection

- Malware detection software usually installed on the system its monitoring.
- Makes it vulnerable to attack and evasion by the malware.
- Inside view of the VM include processes, files, and kernel modules.
- Outside view of the VM are memory pages, registers and disk blocks.

# VMwatcher

- VMwatcher monitors a system from outside the VM.
- Nonintrusive introspection analyzes VM system state & registers.
- “Guest view casting” reconstructs the internal view of the VM from the outside
- Supports both full and paravirtualization.

# IntroVirt

- Uses VM replays to help detect vulnerability intrusions.
- Predicates, or program abstractions, are used to examine state of execution.
- VM introspection, check pointing, and time outs used to protect a system.
- Also used to determine when a system may have been infected.

# Virus Collection Repositories

- Anti-virus developers and security researchers collect and analyze user submitted virus samples.
- Samples are run on VMs
- Allows for easy reset of VM after testing.

# Attacking Virtual Machines

- Some malicious code attempts to detect VMs...
  - Can change behavior, ie not act like a virus
  - Perform malicious activities against VM, cause the VM to exit, DOS
  - Attack the VMM & host system itself.

# Detecting VM from Guest OS

- Check contents of certain HW tables
  - Local Descriptor Table used by VMMs but not Windows
  - Look for HW devices with constant names or NICs with specific MAC addresses.
  - Writing to certain tables cause exception unless run under VMWare

# Getting outside a VM

- Dec 2005, discovery of unchecked copy operation in VMnat for VMWare allowing an attacker to leave the Guest OS.
- Parallels could be crashed issuing a SIDT instruction (Store Interrupt Descriptor Table).

# Summary

- Virtualization can provide great benefits; i.e. lower costs to businesses & aiding in security research.
- Just like all things computers, they have drawbacks and vulnerabilities.