Virtualisation Technology

Sirak Kaewjamnong

Department of Computing

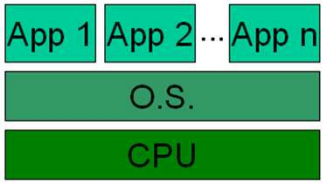
Silpakorn University

Introduction

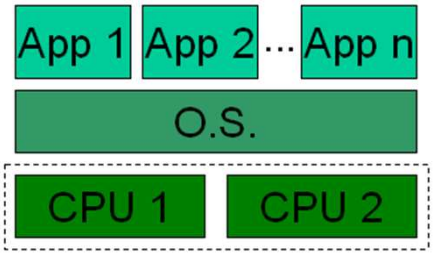
- The traditional solution for data centre is to install standard OS on the individual systems and rely on traditional techniques for resource sharing
- The alternative is resource virtualization
- Virtualisation is a basic tenet of cloud computing
- Resource sharing in a virtual machine environment requires both ample hardware support and architectural support for multilevel control

Terminology

- Multitasking: concurrent executing processes

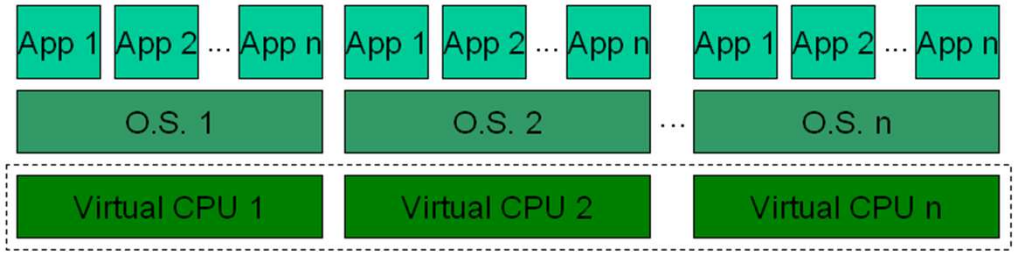


- Multi-core: technology that allows a single processor to have more than one physical processor inside
- Hyper-Threading: technology that simulates an additional processor per CPU core. For example, a dual-core CPU with Hyper-Threading is seen by the OS as if it were a quad-core CPU.



Virtualisation

- Virtualization is a proven software technology that makes it possible to run multiple OSs and applications on the same server at the same time



http://www.vmware.com/virtualization.html

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
- In late 1990s Intel CPUs fast enough for researchers to try virtualizing on general purpose PCs
  - **Xen** and **VMware** created technologies, still used today
  - Virtualization has expanded to many OSes, CPUs, VMMs

Virtualisation's means

- Virtualization simulates the interface to a physical object by any one of four means:
  - Multiplexing: create multiple virtual objects from one instance of a physical object, e.g. a processor is multiplexed among a number of processes
  - Aggregation: create one virtual object from multiple physical objects, e.g. a number of physical disks are aggregated into a RAID disk.
  - Emulation: construct a virtual object from a different type of a physical object, e.g. a physical disk emulates a Random Access Memory.
  - Multiplexing and emulation, e.g. virtual memory with paging multiplexes real memory and disk and a virtual address emulates a real address

Virtualisation in Cloud Computing

- Virtualisation plays an important role for:
  - System security, as it allows isolation of services running on the same hardware
  - Performance and reliability, as it allows applications to migrate from one platform to another
  - The development and management of services offered by a provider
  - Performance isolation

Benefits and Features

- Host system protected from VMs, VMs protected from each other
  - I.e. A virus less likely to spread
  - Sharing is provided though via shared file system volume, network communication
- 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
    - Some VMMs allow multiple snapshots per VM
  - **Clone** by creating copy and running both original and copy
- Great for OS research, better system development efficiency
- Run multiple, different OSes on a single machine
  - **Consolidation**, app dev, …

Benefits and Features (cont.)

- **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

Benefits of Virtualisation

- Virtualization can increase IT agility, flexibility, and scalability while creating significant cost savings
  - Reduce capital and operating costs.
  - Deliver high application availability.
  - Minimize or eliminate downtime.
  - Increase IT productivity, efficiency, agility and responsiveness.
  - Speed and simplify application and resource provisioning.
  - Support business continuity and disaster recovery.
  - Enable centralised management.
  - Build a true Software-Defined Data Center.

http://www.vmware.com/virtualization.html

Virtualisation supported by CPU

- Intel's virtualization technology is available in two versions:
  - VT-x, for x86 processors
  - VT-i, for Itanium (i.e., IA-64) processors
- AMD integrates virtualization technologies directly into all AMD Opteron processors
  - AMD SVM, basic virtualization for AMD CPUs

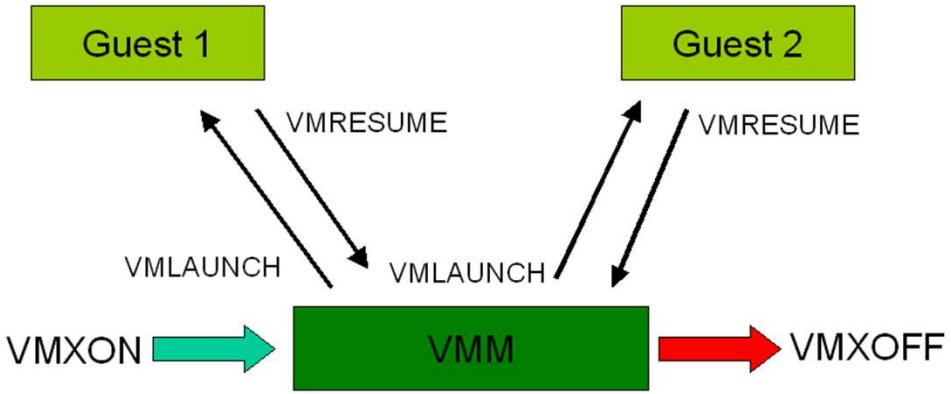
How Virtualisation Works

- Processors with Virtualisation Technology have an extra instruction set called Virtual Machine Extensions(VMX)
- VMX brings 10 new virtualisation-specific instructions to the CPU
- There are two modes to run under virtualisation:
  - VMX root operation: Virtual Machine Monitor (VMM), runs under root operation
  - VMX non-root operation: OSs running on top of the virtual machines run under non-root operation.
  - Software running on top of virtual machines is called “guest software”

How Virtualisation Works

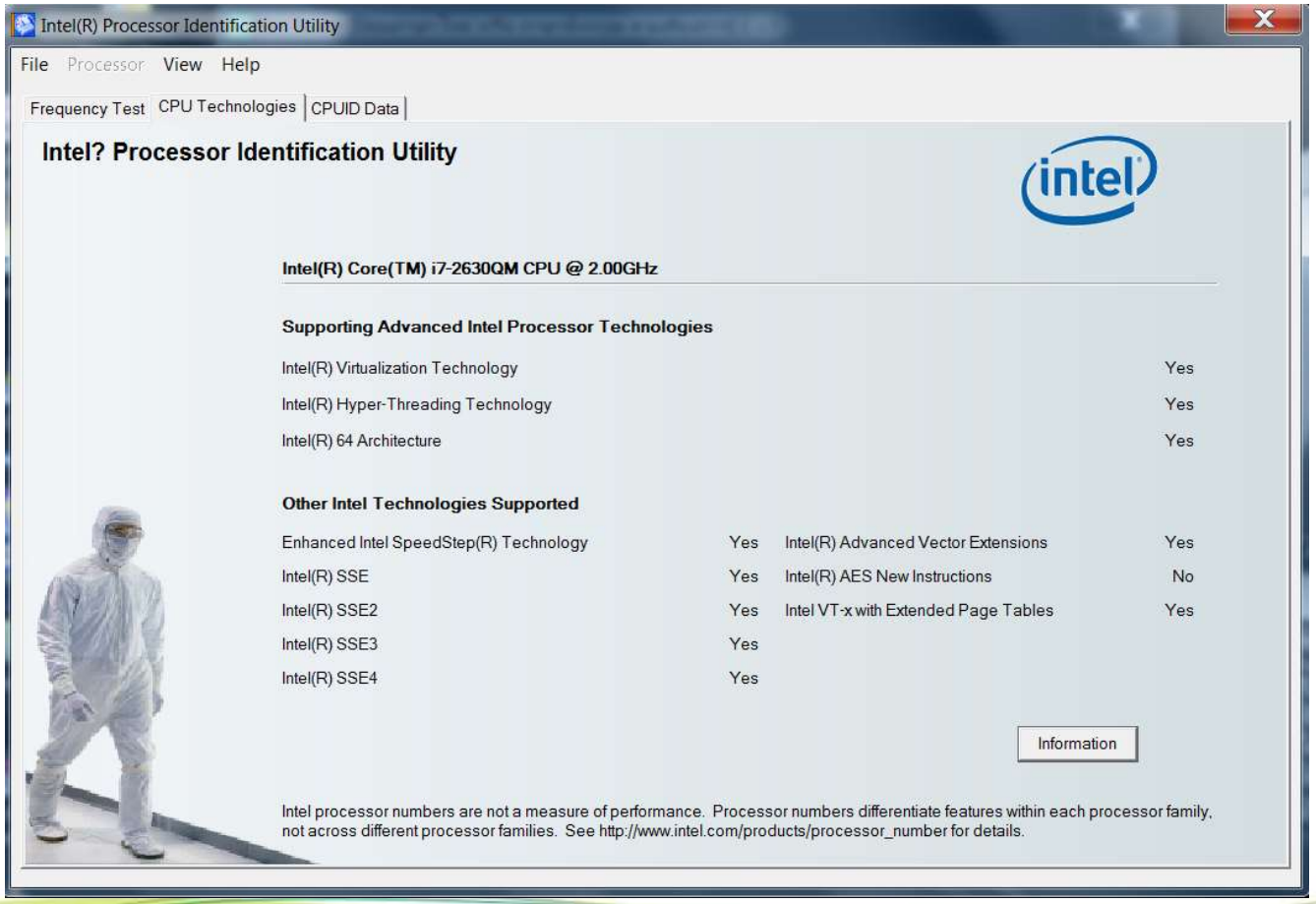
- To enter virtualisation mode, the software should execute the VMXON instruction and then call the VMM software
- The VMM software can enter each virtual machine using the VMLAUNCH instruction, and exit it by using the VMRESUME instruction
- If the VMM wants to shutdown and exit the virtualisation mode, it executes the VMXOFF instruction

How Virtualization Works



- Recent processors have an extension called EPT (Extended Page Tables), which allows each guest to have its own page table to keep track of memory addresses
- Without this extension, the VMM has to exit the virtual machine to perform address translations. This exiting-and-returning task reduces performance

Intel Processor Identification Utility



ISA (Instruction Set Architecture)

- The ISA defines the set of instructions of a processor
- The hardware supports two execution modes, a privileged(kernel mode) and a user mode
  - Privileged instructions can only be executed in kernel mode such as I/O requests
  - Non-privileged instructions can be executed in user mode

Virtual Machine Monitors(VMM)

- A Virtual Machine Monitor (VMM) also called hypervisor is the software that securely partitions the resources of computer system into one or more virtual machines
- A guest operating system is an operating system that runs under the control of a VMM rather than directly on the hardware
- The VMM runs in kernel mode while a guest OS runs in user mode

VMM

- The VMMs allow several operating systems to run concurrently on a single hardware platform at the same time
- VMMs enforce isolation among these systems
- The VMM enables:
  - Multiple services to share the same platform
  - The movement of a server from one platform to another, the so-called live migration
  - System modification while maintaining backward compatibility with the original system

VMM Roles

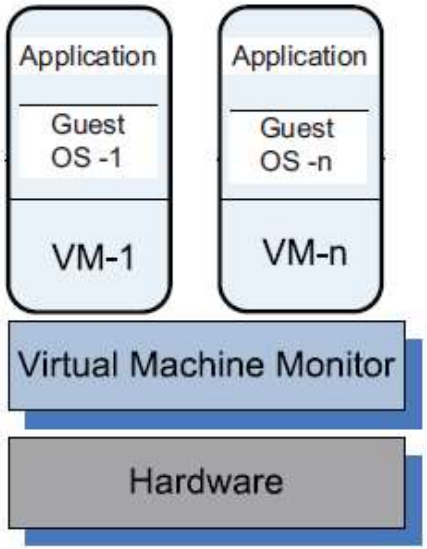
- When a guest OS attempts to execute a privileged instruction the VMM traps the operation and enforces the correctness and safety of the operation
- The VMM monitors the system performance and takes corrective actions to avoid performance degradation,
  - e.g. swap out a Virtual Machine (copies all pages of that VM from real memory to disk and makes the real memory frames available for paging by other VMs)
- The VMM traps interrupts and dispatches them to the individual guest operating systems

Virtual Machines (VM)

- A *Virtual Machine (VM)* is an isolated environment that appears to be a whole computer, but actually only has access to a portion of the computer resources
- Each virtual machine appears to be running on the bare hardware, giving the appearance of multiple instances of the same computer, though all are supported by a single physical system

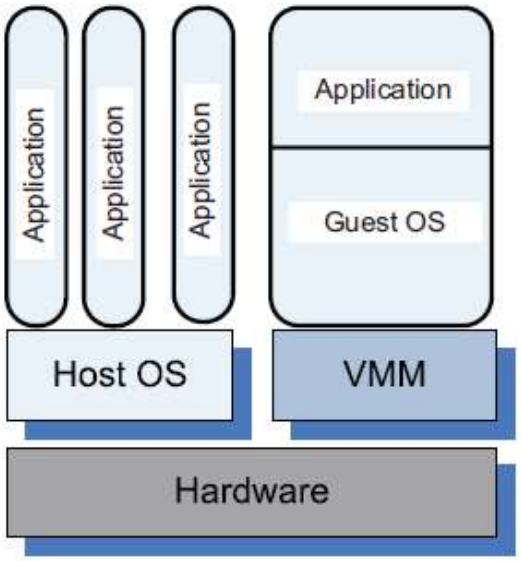
Traditional VMs

- The VMM supports multiple virtual machines and runs directly on the hardware, e.g. VMWare ESX, ESXi Servers, Xen, OS370, and Denali



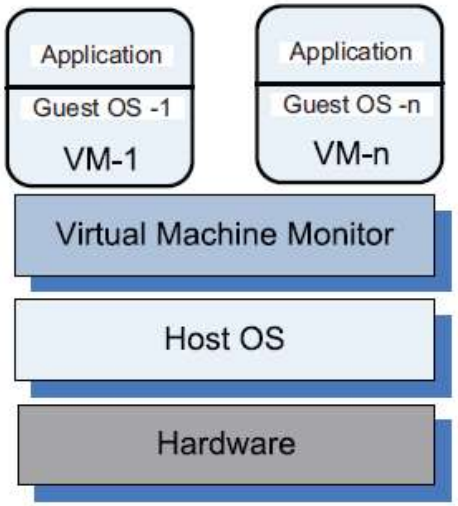
Hybrid VM

- The VMM shares the hardware with a host operating system and supports multiple virtual machines, e.g. VMWare Workstation



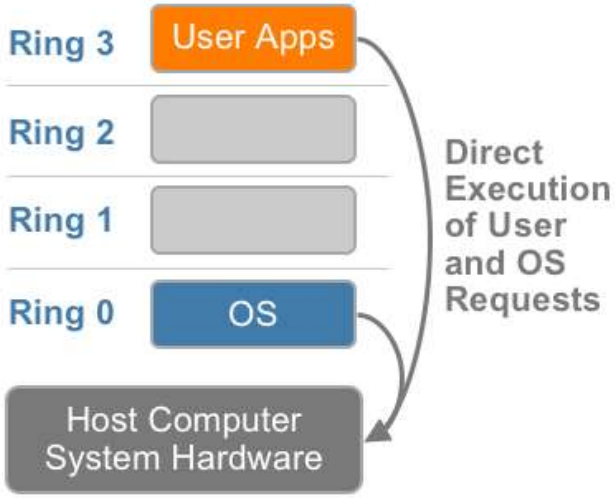
Hosted VM

- The VMM runs under a host operating system
  - Easy to build and install
  - VMM could use components of host OS. E.g. scheduler, the pager and the I/O drivers
  - Increased overhead and the associated performance penalty, e.g. I/O operations, page faults
- Less attractive for servers in a cloud computing environment



CPU Virtualisation

- The x86 architecture offers four levels of privilege known as Ring 0, 1, 2 and 3
- user level applications typically run in Ring 3
- the OS needs to have direct access to the memory and hardware and must execute its privileged instructions in Ring 0
- Virtualising the x86 architecture requires placing a virtualisation layer under the OS (which expects to be in the most privileged Ring 0)



Implementation of VMMs

- Vary greatly, with options including:
  - **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 1 hypervisors** – 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

Implementation of VMMs

- Other variations include:
  - **Paravirtualization** - Technique in which the 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
- Much variation due to breadth, depth and importance of virtualization in modern computing

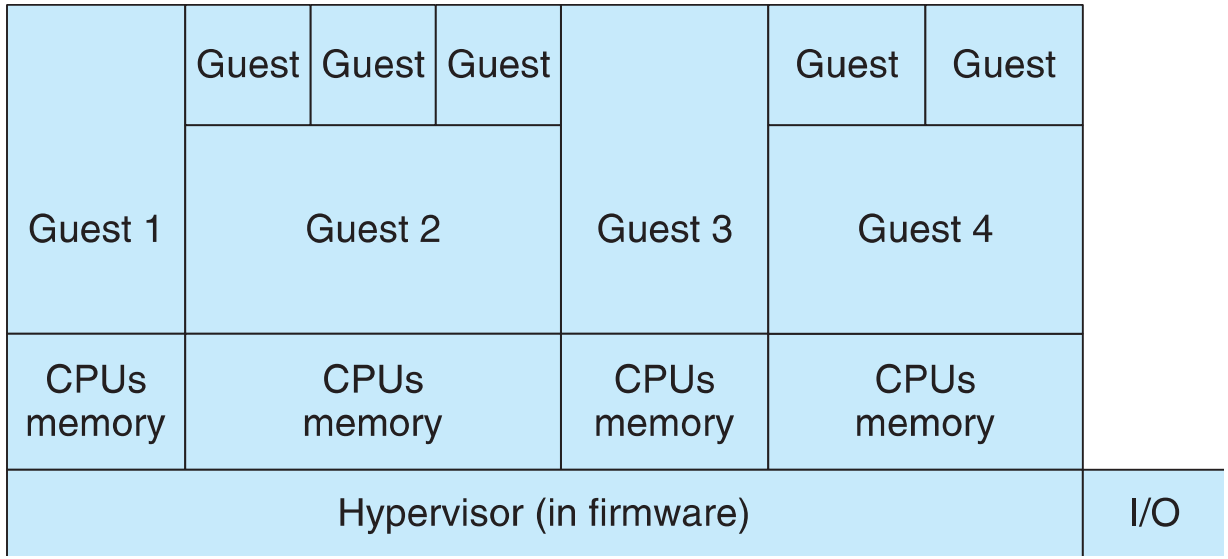
Types of Virtual Machines and Implementations

- Many variations as well as HW details
  - Assume VMMs take advantage of HW features
    - HW features can simplify implementation, improve performance
- Whatever the type, a VM has a lifecycle
  - Created by VMM
  - Resources assigned to it (number of cores, amount of memory, networking details, storage details)
  - In type 0 hypervisor, resources usually dedicated
  - Other types dedicate or share resources, or a mix
  - When no longer needed, VM can be deleted, freeing resouces
- Steps simpler, faster than with a physical machine install
  - Can lead to **virtual machine sprawl** with lots of VMs, history and state difficult to track

Types of VMs – Type 0 Hypervisor

- Old idea, under many names by HW manufacturers
  - “partitions”, “domains”
  - A HW feature implemented by firmware
  - OS need to nothing special, VMM is in firmware
  - 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

Type 0 Hypervisor



Types of VMs – Type 1 Hypervisor

- Commonly found in company data centers
  - In a sense becoming “datacenter operating systems”
    - Datacenter managers control and manage OSs in new, sophisticated ways by controlling the Type 1 hypervisor
    - Consolidation of multiple OSs 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 OSs
  - Can run on Type 0 hypervisors but not on other Type 1s
  - Run in kernel mode
  - Guests generally don’t know they are running in a VM
  - Implement device drivers for host HW because no other component can
  - Also provide other traditional OS services like CPU and memory management

Types of VMs – Type 1 Hypervisor (cont.)

- Another variation is a general purpose OS that also provides VMM functionality
  - RedHat Enterprise Linux with KVM, Windows with Hyper-V, Oracle Solaris
  - Perform normal duties as well as VMM duties
  - Typically less feature rich than dedicated Type 1 hypervisors
- In many ways, treat guests OSes as just another process
  - Albeit with special handling when guest tries to execute special instructions

Types of VMs – 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
    - Student could have Type 2 hypervisor on native host, run multiple guests, all on standard host OS such as Windows, Linux, MacOS

Types of VMs – 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 VMs – Paravirtualization (cont.)

- Xen, leader in paravirtualized space, adds several techniques (Cont.)
  - Memory management does not include nested page tables
    - Each guest has own read-only tables
    - 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

Types of VMs – Programming Environment Virtualization

- Also not-really-virtualization but using same techniques, providing similar features
- Programming language is designed to run within custom-built virtualized environment
  - For example Oracle Java has many features that depend on running in **Java Virtual Machine** (**JVM**)
- In this case virtualization is defined as providing APIs that define a set of features made available to a language and programs written in that language to provide an improved execution environment
- JVM compiled to run on many systems (including some smart phones even)
- Programs written in Java run in the JVM no matter the underlying system
- Similar to **interpreted languages**

Types of VMs – Emulation

- Another (older) way for running one operating system on a different operating system
  - Virtualization requires underlying CPU to be same as guest was compiled for
  - Emulation allows guest to run on different CPU
- Necessary to translate all guest instructions from guest CPU to native CPU
  - Emulation, not virtualization
- Useful when host system has one architecture, guest compiled for other architecture
  - Company replacing outdated servers with new servers containing different CPU architecture, but still want to run old applications
- Performance challenge – order of magnitude slower than native code
  - New machines faster than older machines so can reduce slowdown
- Very popular – especially in gaming where old consoles emulated on new

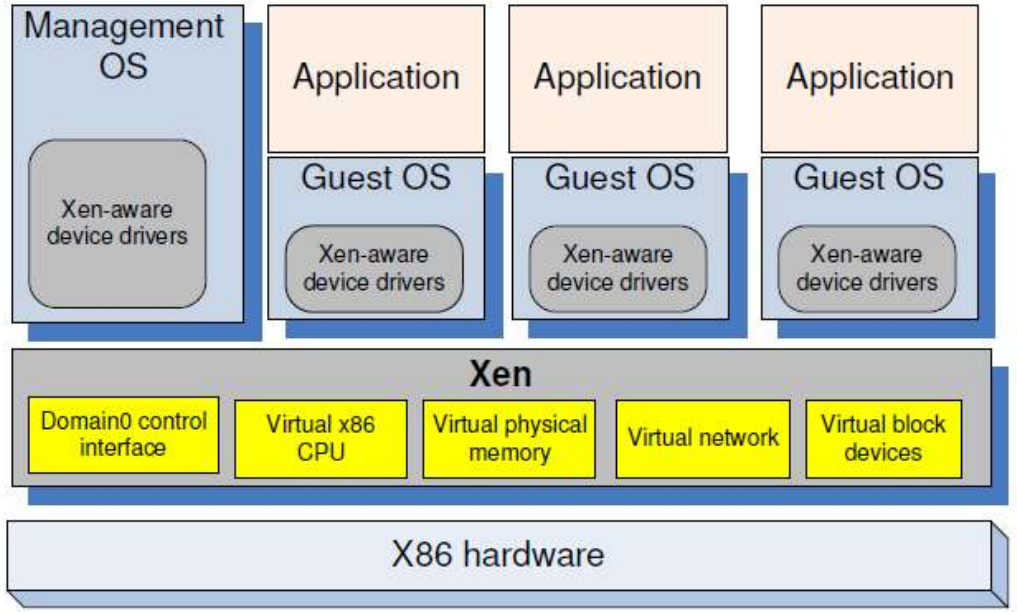
Types of VMs – Application Containment

- Some goals of virtualization are segregation of apps, performance and resource management, easy start, stop, move, and management of them
- Can do those things without full-fledged virtualization
  - If applications compiled for the host operating system, don't need full virtualization to meet these goals
- Oracle **containers** / **zones** for example create virtual layer between OS and apps
  - Only one kernel running – host OS
  - OS and devices are virtualized, providing resources within zone with impression that they are only processes on system
  - Each zone has its own applications; networking stack, addresses, and ports; user accounts, etc
  - CPU and memory resources divided between zones
    - Zone can have its own scheduler to use those resources

Case study: Xen

- Xen is a Virtual Machine Monitor (VMM) or hypervisor developed by the Computing Laboratory at the University of Cambridge, UK, in 2003
- Since 2010 Xen is a free software, developed by the community of users and licensed under the GNU General Public License
- Several operating systems including Linux, Minix, NetBSD, FreeBSD,NetWare, and OZONE can operate as paravirtualised Xen guest OSs running on x86, x86-64, Itanium, and ARM architectures
- Rackspace and Amazon both use Xen

Xen for X86 Architecture



Xen for X86 Architecture

- Xen used the concept of domain (Dom) to refer to the ensemble of address spaces hosting a guest OS and address spaces for applications running under this guest OS
- Each domain runs on a virtual x86 CPU
- Dom0 is dedicated to the execution of Xen control functions and privileged instructions
- DomU is a user domain

Xen for X86 Architecture

- The x86 Intel architecture supports four protection rings or privilege levels
- virtually all OS kernels run at Level 0, the most privileged one, and applications at Level 3
- In Xen the VMM runs at Level 0, the guest OS at Level 1, and applications at Level 3

Side effects of virtualisation

- *Performance penalty* and the *hardware costs*
- All privileged operations of a virtual machine must be trapped and validated by the Virtual Machine Monitor which, ultimately, controls the system behaviour; the increased overhead has a negative impact on the performance.
- The cost of the hardware for a virtual machine is higher than the cost for a system running a traditional operating system because the physical hardware is shared among a set of guest OSs and it is typically configured with faster and/or multi-core processors with more memory, larger disks, and additional network interfaces as compared with a system running a traditional operating system

References

- Operating System Concepts 10th book official slides by Abraham Silberschatz, Greg Gagne, Peter B. Galvin
- Cloud Computing: Theory and Practice by Dan C. Marinescu
- http://www.hardwaresecrets.com/

  everything-you-need-to-know-about-the-intel-virtualization-technology/
- http://www.vmware.com/virtualization.html
- VMWare white paper “Understanding Full Virtualization, Paravirtualization, and Hardware Assist”