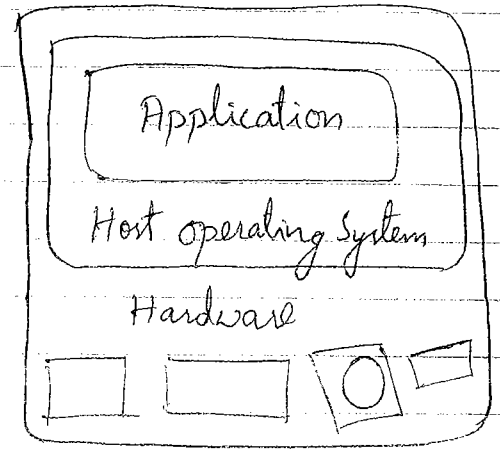
UNIT IV - VIRTUALIZATION

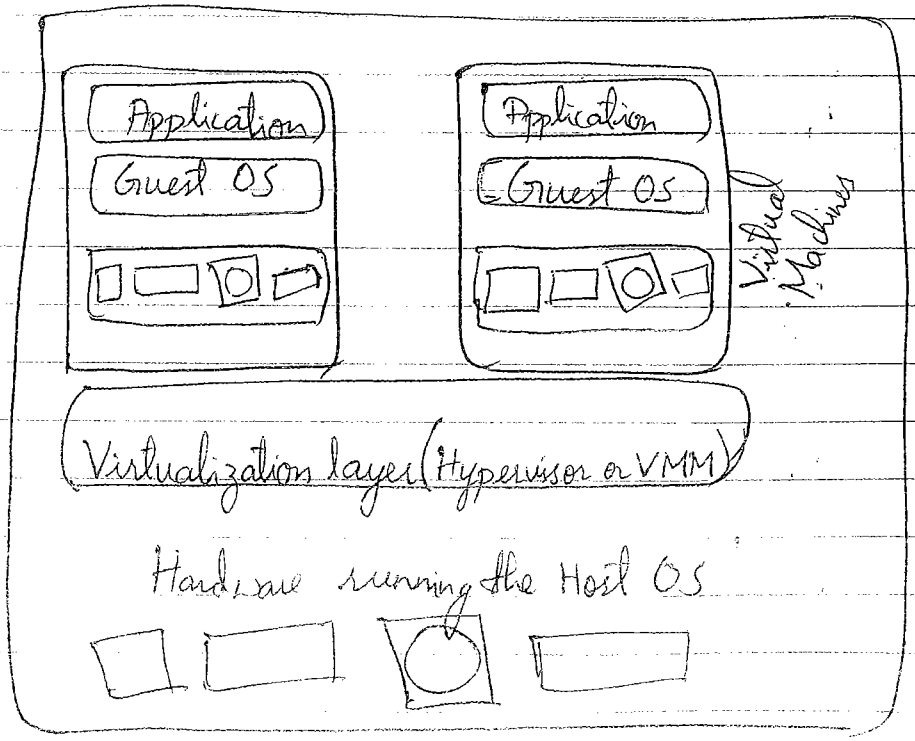
Implementation Levels of Virtualization

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.

Levels of Virtualization Implementation



(a) Traditional computer.

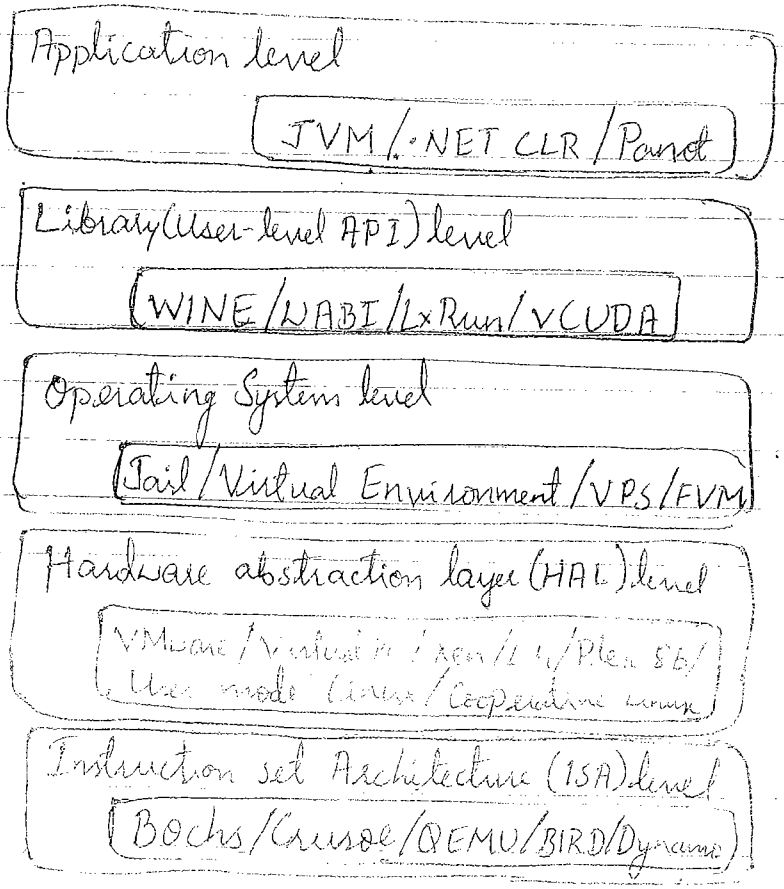


(b) After Virtualization

- The different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This software is called virtualization layer or hypervisor or virtual machine monitor (VMM).

- The main function of hypervisor is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs exclusively.

Common virtualization layers include
- instruction set architecture (ISA) level
- hardware level
- operating system level
- application level



QEMU Quick Emulator

Fig: Virtualization ranging from hardware to appl^{n} in

Instruction Set Architecture level

- Virtualization is performed by emulating given ISA by the ISA of the host machine.
- Ex- MIPS (Microprocessor without Interlocked Pipeline Stages) binary code can run on an x86-based host machine with help of ISA emulation.
- The basic emulation ~~method~~ is through code interpretation.
- An interpreter program interprets the source instructions to target inst= one by one
- One source instruct= may require tens or hundreds of native target instructions to perform its function.
- Dynamic binary translation approach translates basic blocks of dynamic source instructions to target instructions.
- A virtual instruction set architecture (V-ISA) requires adding a processor-specific software translation layer to the compiler.

Hardware Abstraction Level

Operating System level

- This refers to an abstraction layer bet traditional OS & user applications.
- OS-level virtualizat creates isolated containers on a single physical server & the OS instances to utilize the hardware & software in data centers.
- The containers behave like real servers.
- This is used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

Library Support Level

Virtualizat with library interfaces is possible by controlling the communication link between applications & the rest of a system through API hooks.

Ex: The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts.

User-Application Level

Applicat-level virtualization is also known as process-level virtualization.

- Java Virtual Machine (JVM) is ex.

Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming.

Relative Merits of Different Approaches

Level of Implementation	Higher Performance	Application Flexibility	Implement^n Complexity	Appl^n Isolat^n
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualizat^n	XXXXX	XXX	XXXXX	XXXX
OS-level virtualizat^n	XXXXX	XX	XXX	XX
Runtime library support	XX X	XX	XX	XX
User applicat^n level	XX	XX	XXXXX	XXXXX

Implementation Complexity implies the cost to implement that particular virtualization level
Application Isolation refers to the efforts required to isolate resources committed to different VMs.

VMM Design Requirements & Providers

There are three requirements for a VMM

1) VMM should provide an environment for programs which is essentially identical to the original machine
2) Programs run in this environment should show, at worst, only minor decrease in speed
3) a VMM should be in complete control of the system resources.

Complete control of these resources by a VMM includes the following aspects

1) the VMM is responsible for allocating hardware resources for programs
2) it is not possible for a program to access any resources not explicitly allocated to it
3) it is possible under certain circumstances for a VMM to regain control of resources already allocated.

Comparison of Four VMM & Hypervisor Software Packages.

	Provider	Host CPU	Host OS	Guest OS	Architecture
1.	VMware Workstation	x86, x86-64	Windows, Linux	Windows, Linux, Solaris FreeBSD, Netware, OS/2, SCO, Darwin	Full Virtualization
2	VMware ESX Server	x86, x86-64	No host OS	11	Para-Virtualization
3	Xen	x86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD Linux, Solaris Windows XP & 2003 Server	Hypervisor
4	KVM	x86, x86-64, IA-64, S390, PowerPC	Linux	Linux, windows FreeBSD, Solaris	Para-Virtualization

Virtualization Support at the O.S level

The cloud computing has atleast two challenges.
- First is the ability to use a variable number of physical machines & VM instances depending on the need of a problem.
- The second challenge concerns the slow operation of instantiating new VMs. New VMs originate either as fresh boots or as replicates of a template VM, unaware of the current appl^n state.

Why OS-level Virtualization?

- It is slow to initialize a hardware-level VM because each VM creates its own image from scratch.
- In cloud environment thousands of VMs need to be initialized simultaneously.
- Slow operation, storing the VM images also becomes an issue. VM images have repeated content.

- O.S level of virtualizat^n insert a virtualizat^n layer inside an operating system to partition a machine's physical resources.
- It enables multiple isolated VMs within a single operating system kernel. This kind of VM is called virtual execution environment (VE), Virtual Private System (VPS), or simply containers.

- VE has its own set of processes, file system, user accounts, network interfaces with IP address, routing table, firewall rules, other settings.

Advantages of OS Extensions

1) VMs at the operating system level have minimal startup/shutdown costs, low resources requirements, & high scalability.
2) for an OS level VM, it is possible for a VM & its host environment to synchronize state changes when necessary.

These benefits can be achieved via two mechanisms of OS-level virtualization:

1) All OS-level VMs on the same physical machine share a single operating system kernel.
2) the virtualization layer can be designed in a way that allows processes in VMs to access as many resources of the host machine as possible, [illegible]

Disadvantages of OS Extensions

- All the VMs at operating system level on a single container must have the same kind of guest operating system.
- That is, although different OS-level VMs may have different operating system distributions, they must pertain to the same operating system family.

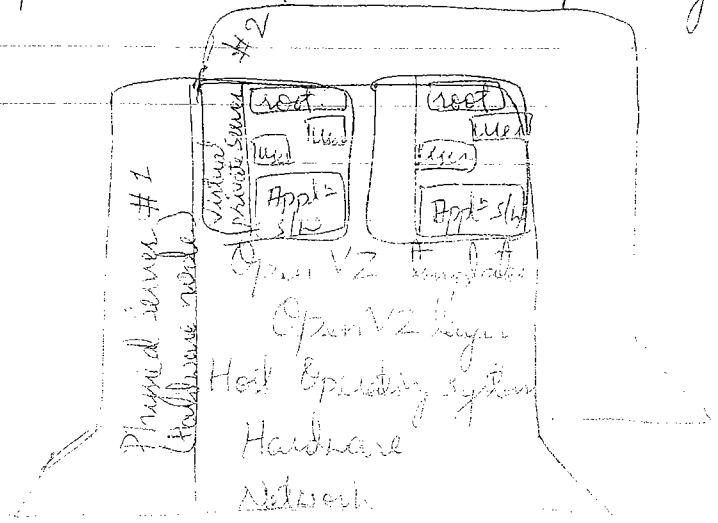


Fig OpenVZ virtualization layer inside the host OS, which provides some OS images to create VMs quickly.

Virtualization on Linux or Windows Platforms

Virtualization Support & Source of Info	Brief Intro on Functionality & Applicat= Platforms
1) Linux vServer for Linux platforms	Extends Linux kernels to implement a security mechanism to help build VMs by setting resources limits & file attributes & changing the root environment for VM isolation.
2) OpenVZ for Linux platforms	Supports virtualization by creating virtual private servers (VPSes); VPS has its own files, users, process tree, virtual devices, checkpointing & live migration are supported.
3) FVM (Feather-Weight Virtual Machines) for virtualizing the Windows NT	Uses system call interfaces to create VMs at the NT kernel space; multiple VMs are supported by virtualized namespace & copy-on-write.

Middleware Support for Virtualization

- Library-level virtualization is also known as user-level Application Binary Interfaces (ABI) or API emulation
- This type of virtualization can create execution environment for running alien programs on a platform rather than creating a VM to run the entire operating system.
- API call interception & remapping are the key functions performed

The vCUDA for Virtualization of General-Purpose GPUs

CUDA - Compute Unified Device Architecture

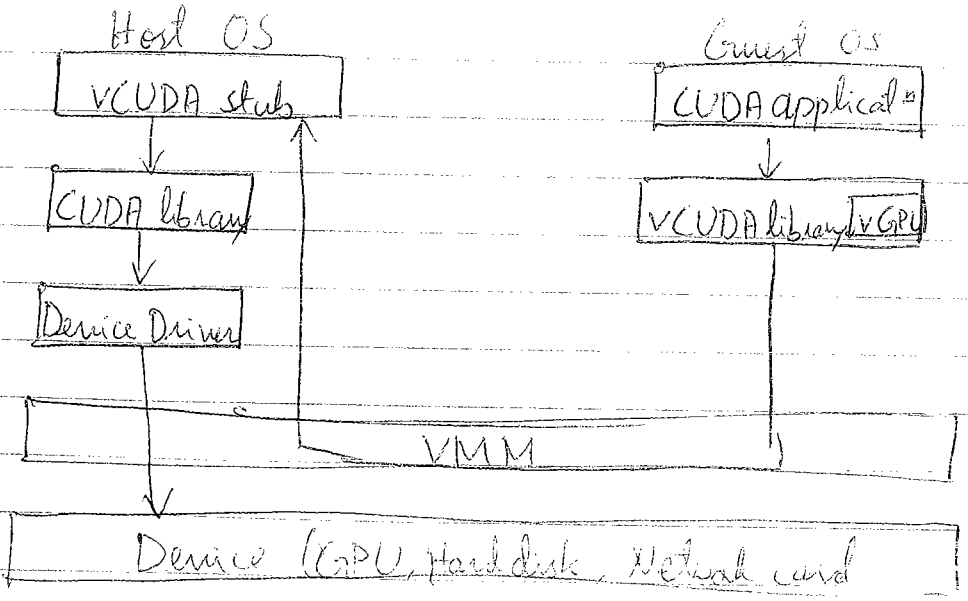


Fig Basic concept of the vCUDA architecture

- It is difficult to run CUDA applications on hardware-level VMs directly
- vCUDA virtualizes the CUDA library and can be installed on guest OS

- When CUDA appl^n run on a guest OS & issue a cal to the CUDA API, vCUDA intercepts the call & redirects it to the CUDA API running on the host OS.

- The vCUDA employs a client-server model to implement CUDA virtualization.

- vCUDA library resides in the guest OS as a substitute for standard CUDA library.
- It intercepts & redirect API calls from client to the stub.

Functions of vGPU are.
- It abstracts the GPU structure & gives appl^n a uniform view of the underlying hardware
- When a CUDA application in the guest OS allocates a device's memory the vGPU can return a local virtual address to the appl^n & notify the remote stub to allocate the real device memory.
- vGPU is responsible for storing the CUDA API flow

Virtualization Structures/Tools & Mechanisms

Depending on the position of the virtualization layer, there are basically three classes of VM architecture.

1) hypervisor architecture (Virtual machine monitor)
2) para-virtualization
3) host-based virtualization

Hypervisor & Xen Architecture

- Hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory & disk & network interfaces.
- The hypervisor provides hypercalls for the guest OS

Depending on functionality hypervisor

micro-kernel architecture
- includes only the basic & unchanging functions like physical memory management & processor scheduling
- size is smaller

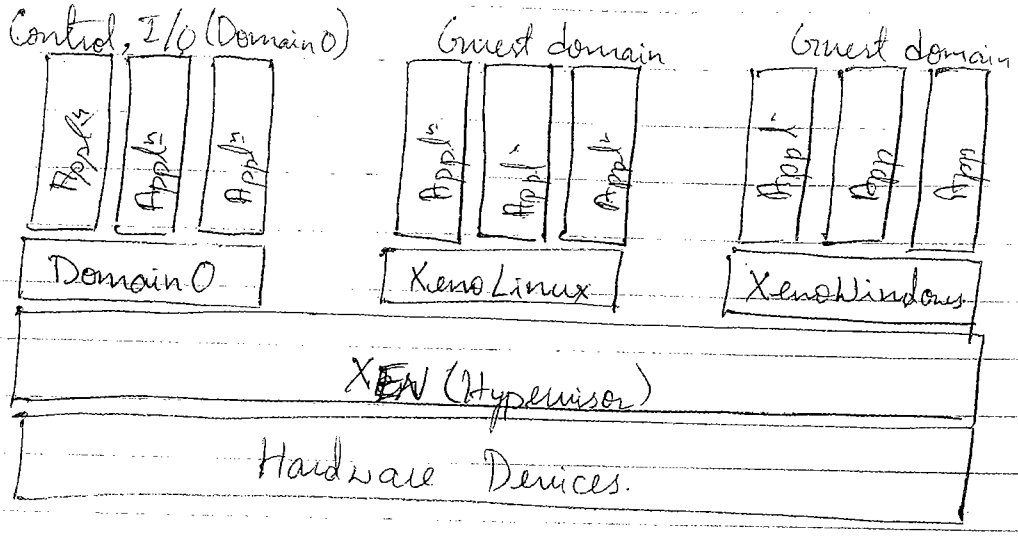
Ex: Microsoft Hyper-V

monolithic hypervisor architecture
- includes all functions including device drivers

Ex: VMware ESX

The Xen Architecture

- Xen is micro-kernel hypervisor.



Binary Translation with Full Virtualization.

Depending on implementation technologies hardware virtualization can be classified

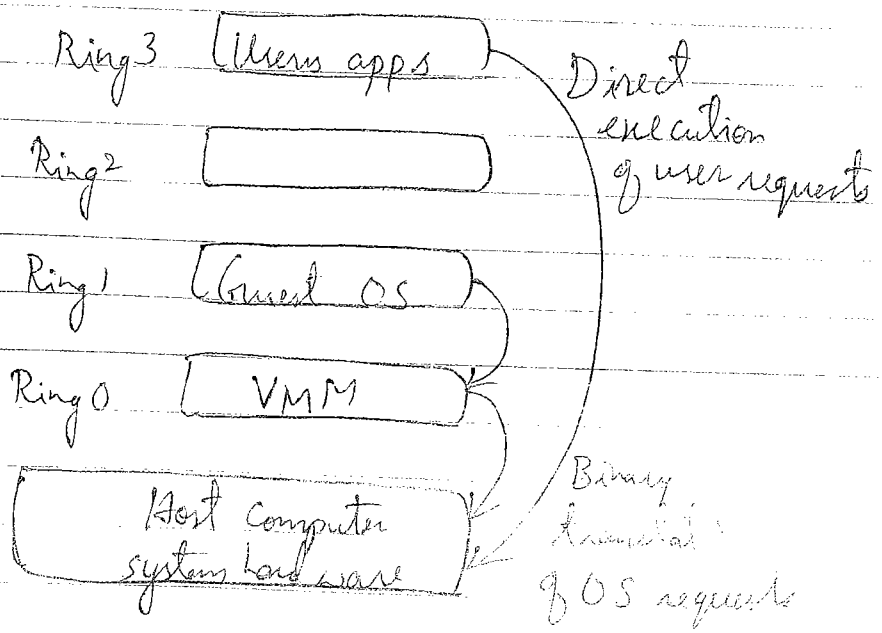
full virtualizat= - does not need to modify the host OS

host-based virtualizat= - ~~modify~~ the

Full virtualization

- Noncritical instructions run on the hardware directly
- critical instruct= are discovered & replaced with traps into the VMM to be emulated by software.

Binary Translation of Guest OS Requests using a VMM



Host-Based Virtualization

- In this virtualization layer is installed on top of the host OS. This host OS is responsible for managing the hardware.

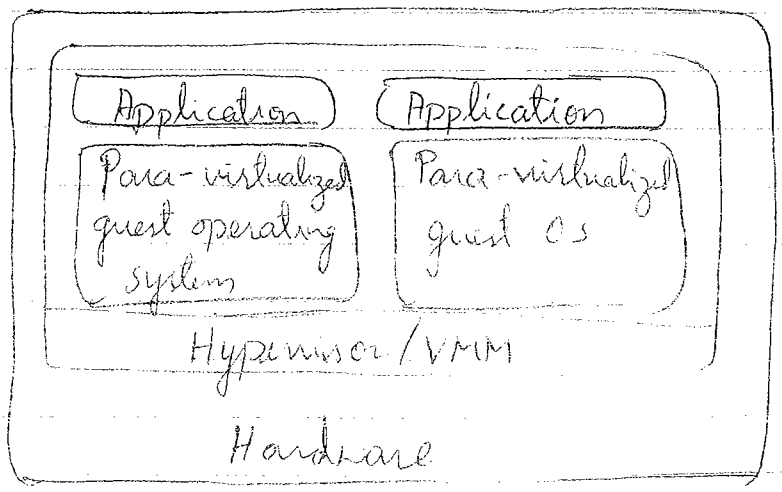
Advantages

1) The user can install this VM architecture without modifying the host OS. The virtualizing software can rely on host OS to provide device drivers & other low-level services.

2) The host-based approach appeals to many host machine configurations. This is more flexible.

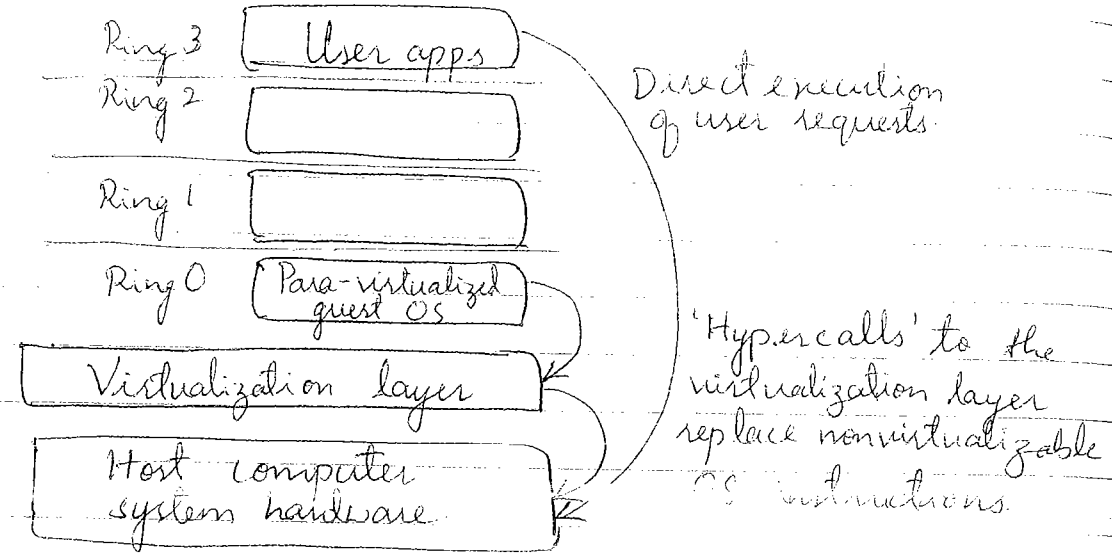
Para-Virtualization with Compiler Support

- Para-virtualization needs to modify the guest operating systems.
- A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications.
- Para-virtualization attempts to reduce the virtualization overhead, & thus improve performance by modifying only the guest OS kernel.



Para-virtualized VM architecture.

Para-Virtualization Architecture



Virtualization Structures / Tools & Mechanisms

- Depending on the position of virtualization layer, there are several classes of VM architecture
  - hypervisor architecture
  - para-virtualization
  - host-base virtualizat=

Hypervisor and Xen Architecture

(VMM)

- supports hardware-level virtualizat= - on bare metal devices like CPU, memory, disk & network interface.
- provides hypercalls for guest OSes & applications.

depending on functionality hypervisor are of two types
- micro-kernel architecture
  Ex: Microsoft Hyper-V
- monolithic hypervisor architecture
  Ex: VMware ESX

micro-kernel hypervisor
- includes only the basic & unchanging functions like physical memory management & processor scheduling
- device drivers & other changeable components are outside the hypervisor.

monolithic hypervisor - implements all physical memory management, processor scheduling, & device drivers.

Size of micro-k...

Xen Architecture

- Open source hypervisor developed by Cambridge University.
- Xen is a micro-kernel hypervisor
- separates policy from the mechanism.
- Xen does not include any device drivers natively.
- It just provides a mechanism by which a guest OS can have direct access to the physical devices.
- Size is small
- provides virtual environment located bet= hardware & OS.
- commercial Xen hypervisors – Citrix XenServer
  - Oracle VM.

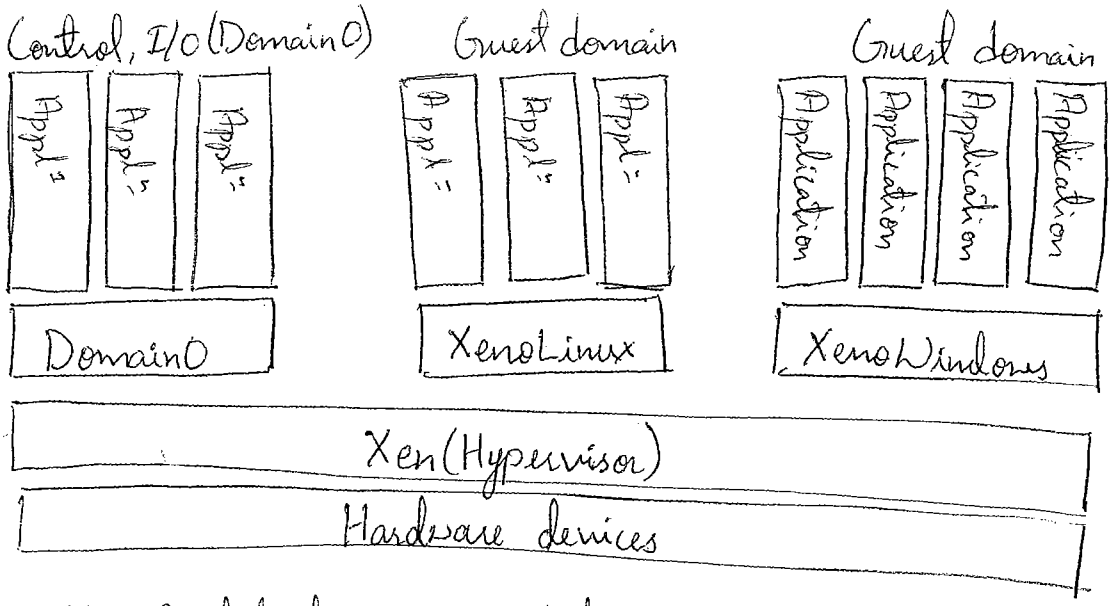


Fig: Xen Architecture's special domain 0 for control & I/O.

- Core components of Xen system - hypervisor, kernel & applications.
- guest OS run on top of hypervisor.
- not all guest OSes are created equal, & one in particular controls the other
- The guest OS which has control ability is called Domain0 & others are called DomainU.

- Domain 0 is first loaded when Xen boots without any file system drivers being available.
- Domain 0 is designed to access hardware directly & manage devices
- Domain 0 is responsible to allocate & map hardware resources for guest domains.
- Domain 0, behaving as a VMM, allows users to create, copy, save, read, modify, share, migrate & roll back VMs as easily as manipulating a file.

Binary Translation with Full Virtualization

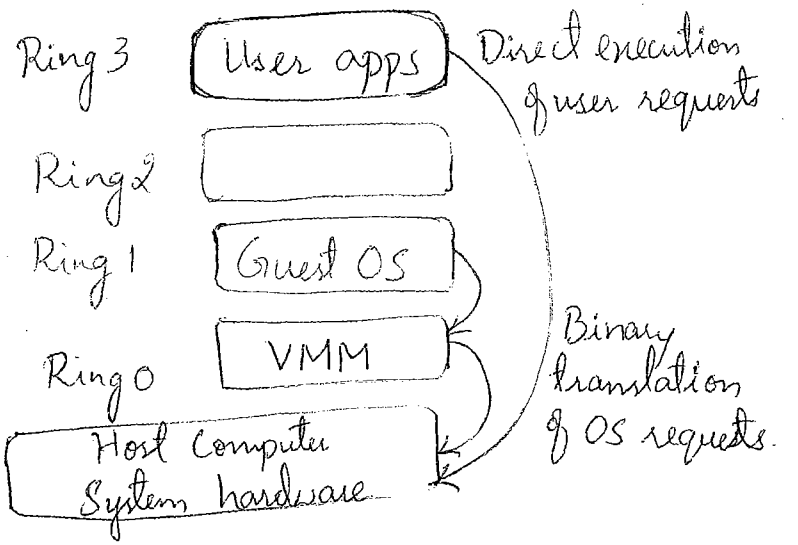
Depending on implementation technologies, hardware virtualization is classed
- full virtualization
- host-based virtualization

Full Virtualization

- It does not need to modify the host OS.
- It relies on binary translation to trap & to virtualize the execution of certain sensitive, nonvirtualizable instructions.
- The guest OSes & their applications consists of noncritical & critical instructions.
- Non-critical instructions run on the hardware directly
- critical instructions are discovered & replaced with traps into the VMM to be emulated by software.

- Why are only critical instructions trapped into the VMM? This is because binary translation can incur a large performance overhead.
- Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do.
- Running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

Binary Translation of Guest OS Requets Using a VMM



VMware puts VMM at Ring 0, the guest OS at Ring 1.

- VMM scans the instruction stream & identifies the privileged, control- and behavior-sensitive instructions.
- When these instructions are identified, they are trapped into VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation.
- Full virtualization combines binary translation & direct execution.
- Guest OS is completely decoupled from underlying H/w.
- I/O-intensive applications is really big challenge.
- Binary translation employs a code cache to store translated hot

Host-Based Virtualization

- Virtualizat- layer is installed on top of the host OS.
- Dedicated appl-s may run on the VMs. Some other appl- can also run with the host OS directly.

Adv - (1) User can install VM architecture without modifying host OS. The virtualizing s/w can rely on host OS to provide device drivers & other low-level services. This will simplify VM design & ease its deployment.

Para-Virtualization with Compiler Support

- Para-virtualization needs to modify the guest OS.
- Para-virtualized VM provides special APIs requiring substantial OS modifications in user applications.
- Para-virtualizat- attempts to reduce the virtualization overhead & thus improve performance by modifying the only the guest OS kernel.
- The guest OS systems are para-virtualized. They are assisted by
- an intelligent compiler to replace the nonvirtualizable OS instructions by hypercalls.
- The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2 and 3.
- The lower the ring number, the higher the privilege of instruction being executed.
- The OS is responsible for managing the hardware & privileged instructions to execute at Ring 0, while user-level applications runs at Ring 3

Para-Virtualization Architecture

- When x86 processor is virtualized, a virtualization layer is inserted between the hardware & OS.
- According to x86 ring definit=, virtualizat= layer should be installed at Ring 0S.
- When guest OS kernel is modified for virtualization, it can no longer run on hardware directly.

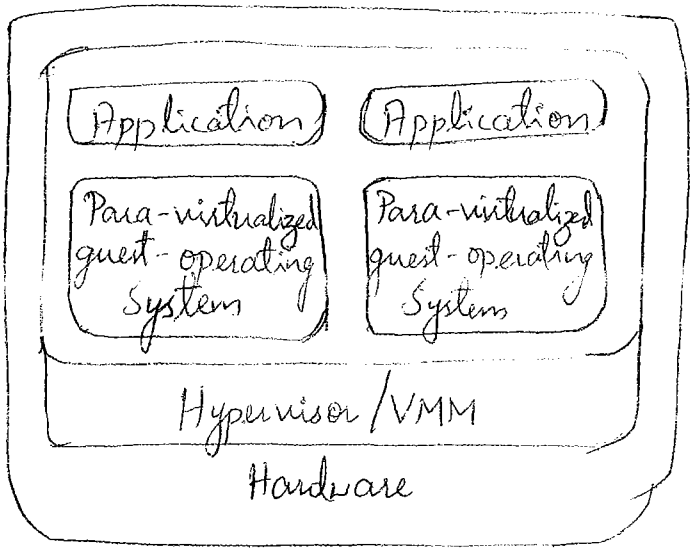
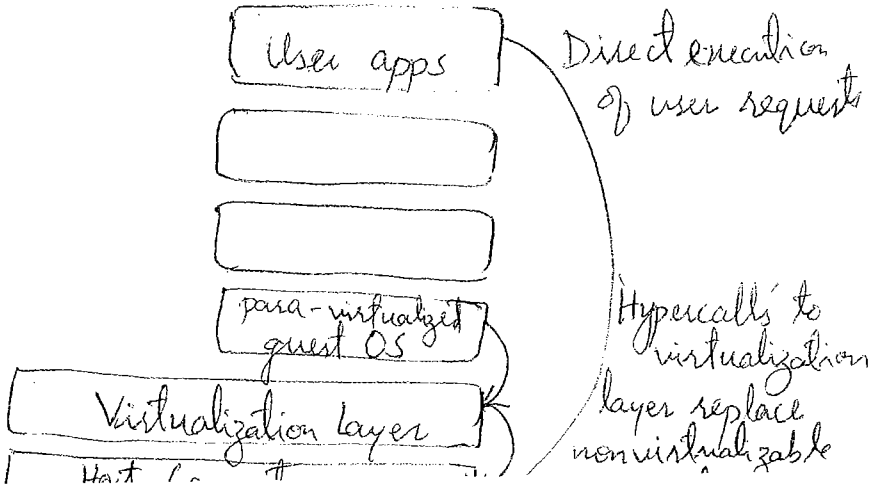


Fig: Para-virtualized VM architecture.

- The guest OS are para-virtualized.
- They are assisted by an intelligent compiler to replace the nonvirtualizable OS instructions by hypercalls.



Disadvantages (Problems)

- its compatibility & portability may be in doubt, because it must support the unmodified OS as well.
- cost of maintaining para-virtualized OSs is high, because they may requires deep OS kernel modifications.

Adv.

- Compared with full virtn, para-virtn is relatively easy & more practical.
- In full virtn performance is low bcz of binary translatn.

KVM (Kernel-Based VM)

- This is a linux para-virtualizatn system - a part of linux version 2.6.20 kernel.
- Memory mngt & scheduling are carried out by existing linux kernel. Rest activities are done by KVM.
- KVM is hard-ware-assisted para-virtualizatn tool, which improve performance & supports unmodified guest OSes such as Windows, Linux, solaris & other UNIX variants.

VMware ESX Server for Para-Virtualization

- ESX is a VMM or a hypervisor for bare-metal x86 symmetric multiprocessing (SMP) servers.
- It accesses hardware resource such as I/O directly & has complete resource management control.

An ESX-enabled server consists of four components:
- a virtualization layer
- a resource manager
- hardware interface components
- a service console.

- In this VMkernel interacts directly with H/W without involving host OS

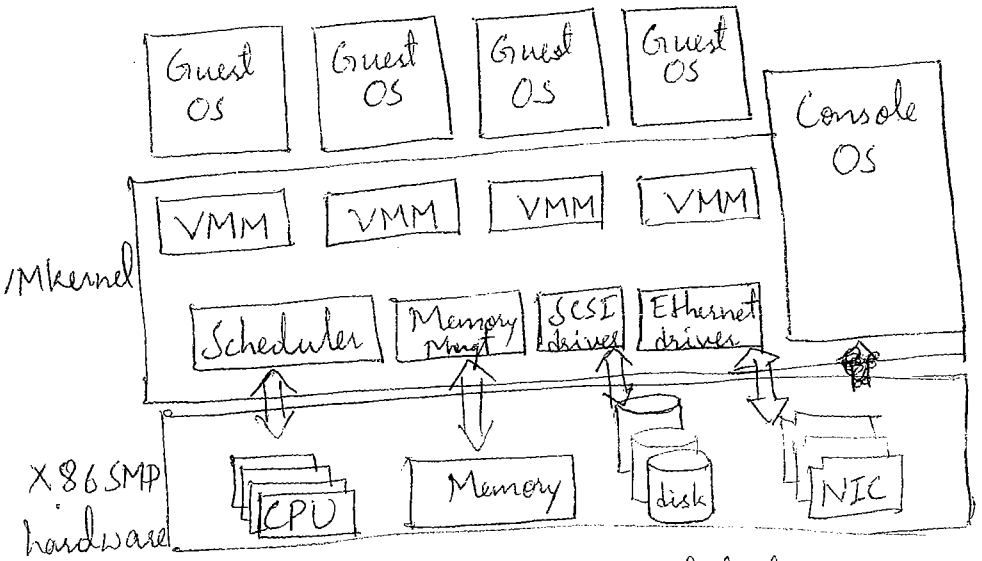


Fig: VMware ESX Server architecture using para-virtualization.

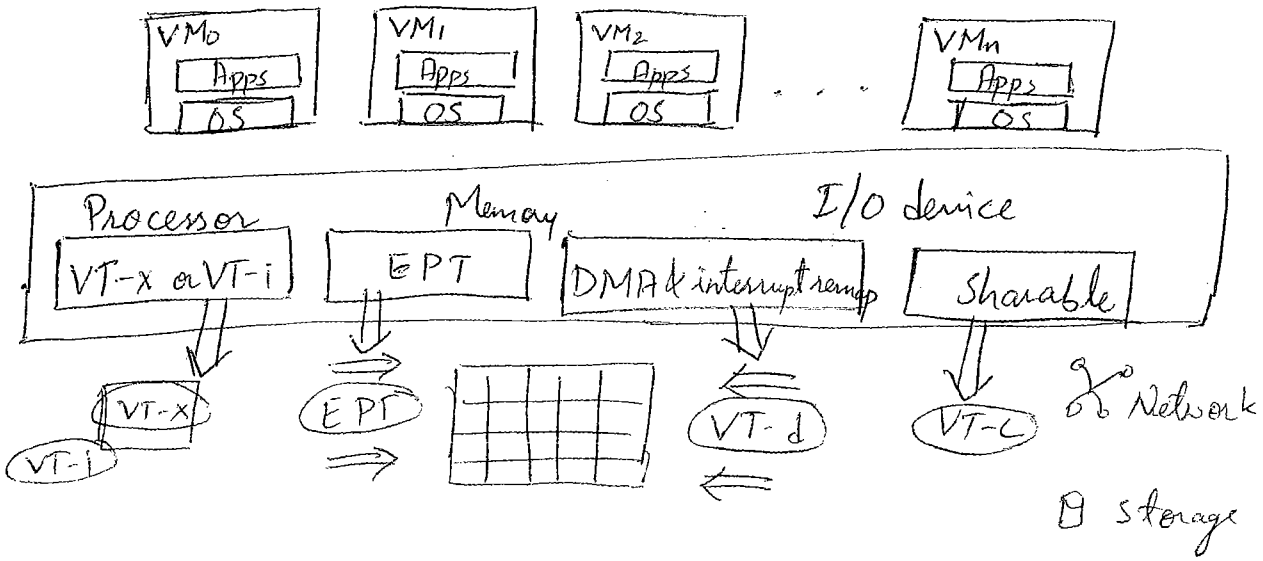
- The resource manager allocates CPU, memory disk, & network bandwidth & maps them to virtual hardware resource set of each VM created.
- Hardware interface components are device drivers & the VMware ESX Server File System.
- The service console is responsible for booting the system, & initiating the execution of VMM & resource manager & relinquishing control to those layers.
- It also facilitates the process for system administrators.

Virtualization of CPU, Memory & I/O Devices

- User mode - unprivileged inst=
- Supervisor mode - privileged instructions

Hardware Support for Virtualizat= in Intel x86 processor

- H/w assisted virt=
- Processor virt=, VT-x or VT-i technique.
- VT-x adds privileged mode (VMX Root mode)
  Some inst= to processor
  - traps all sensitive inst= in VMM
- For memory virt=, Intel offers EPT, - translates the virtual address to machine's physical address.
- For I/O virtualizat=, VT-d & VT-c to support.



CPU Virtualization

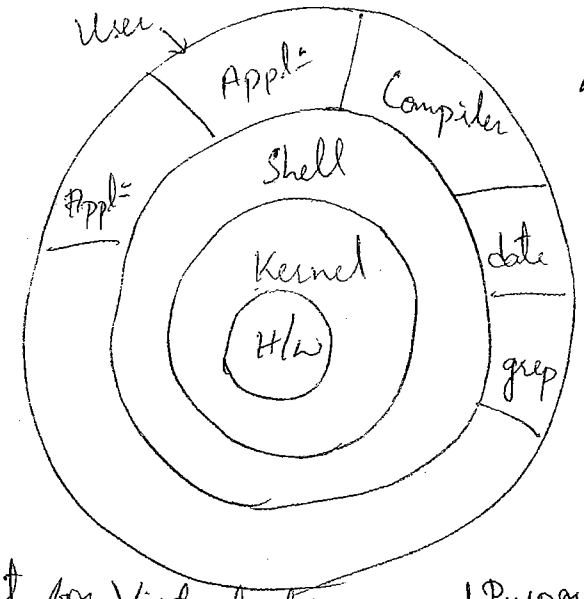
- unprivileged inst= of VM run directly on host M/C

critical instructions
- privileged inst= executed in — privileged mode
- control-sensitive inst= — change the config of resources used
- behavior-sensitive " ↓ different behaviours depending on config.

- A CPU architecture is virtualizable if it supports the ability to run the VM's privileged & unprivileged inst= in CPU's user mode while VMM run in supervisor mode.
- When privileged instructions including control & behavior sensitive instructions of VM are executed, they are traped in VMM.
  - VMM acts as unified mediator for H/w access from different VMs
- RISC CPU architecture can be virtualized naturally, because all control- & behavior-sensitive inst= are privileged inst=.
- x86 CPU architecture are not primarily designed to support virt=. bcz abt 10 sensitive inst= are not privileged inst=. when these inst= execute in virt=, they cannot be trapped in VMM.

Resource management subsystem of OpenVZ

- Two level disk allocation – 1st level - amount of disk space by VM
  – 2nd level – Each VM acts as a std Linux system – for each use & group.
- Two-level CPU scheduler
  – 1st level – which VM to give the time slice to
  – 2nd " – same as that of Linux
- resource controller



Windows System Calls
↓ WABI
Solaris system calls

Middleware Support for Virtualizat= General Purpose.

vCUDA for Virtualizat= of a GPU.

vCUDA library, virtual GPU, vCUDA stub.
↓
guest OS
as
substitute
for std CUDA lib

- CUDA is library for general-purpose GPUs.
- difficult to run CUDA applicat= on hardware-level VMs directly.
- vCUDA virtualizes the CUDA library & can be installed on guest OS.

- When CUDA appl^n runs on a guest OS & issue call to CUDA API, vCUDA intercepts the call & redirects it to CUDA API running on the host OS.
- vCUDA employs a client-server model to implement CUDA virtualizat^n.
- It consists of three user space components
  - vCUDA library
  - virtual GPU in the guest OS. (client)
  - vCUDA stub in the host OS (server)

vCUDA library
- resides in guest OS as substitute for std CUDA library.
- responsible for intercepting & redirecting API calls from client to the stub.
- creates vGPU & manages them.

vGPU
- it abstracts GPU structure & gives appl^n a uniform view of underlying hardware.
- When a CUDA applicat^n in guest OS allocates a device's memory, the vGPU can return a local virtual address to application & notify the remote stub to allocate real device memory.
- vGPU is responsible for storing the CUDA API flow.

vCUDA stub
- receives & interprets remote requests and create a corresponding execution context for the API calls from guest OS & return results
- manages actual physical resource allocation