Cloud Computing

Concept, Technology & Architecture

Unit-4

Virtualization of Clusters and Data Centers

IMPLEMENTATION LEVELS OF VIRTUALIZATION

This can be implemented at various operational levels,

- Instruction set architecture (ISA) level,
- Hardware level,
- Operating system level,
- Library support level, and
- Application level

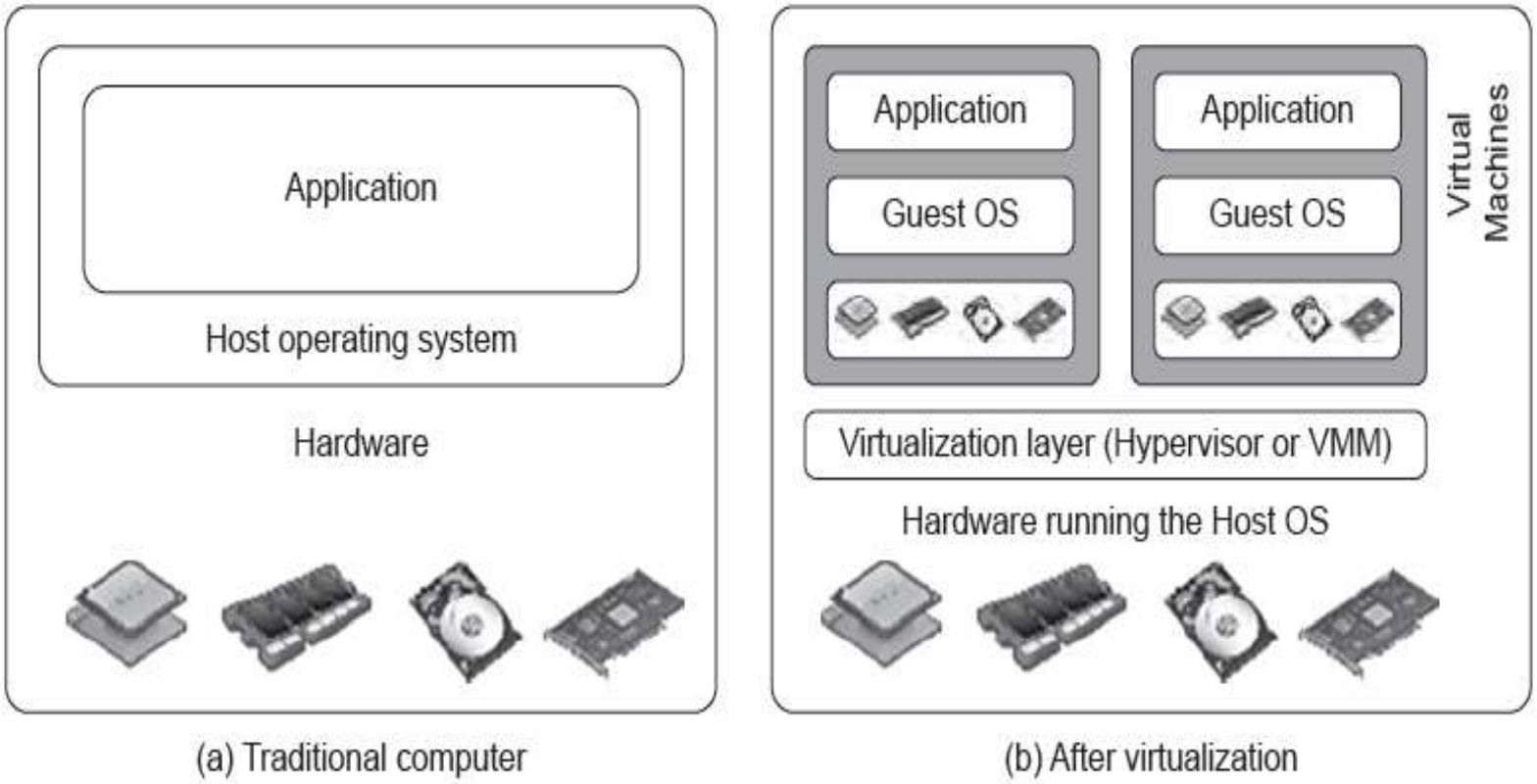

FIGURE 3.1

The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.

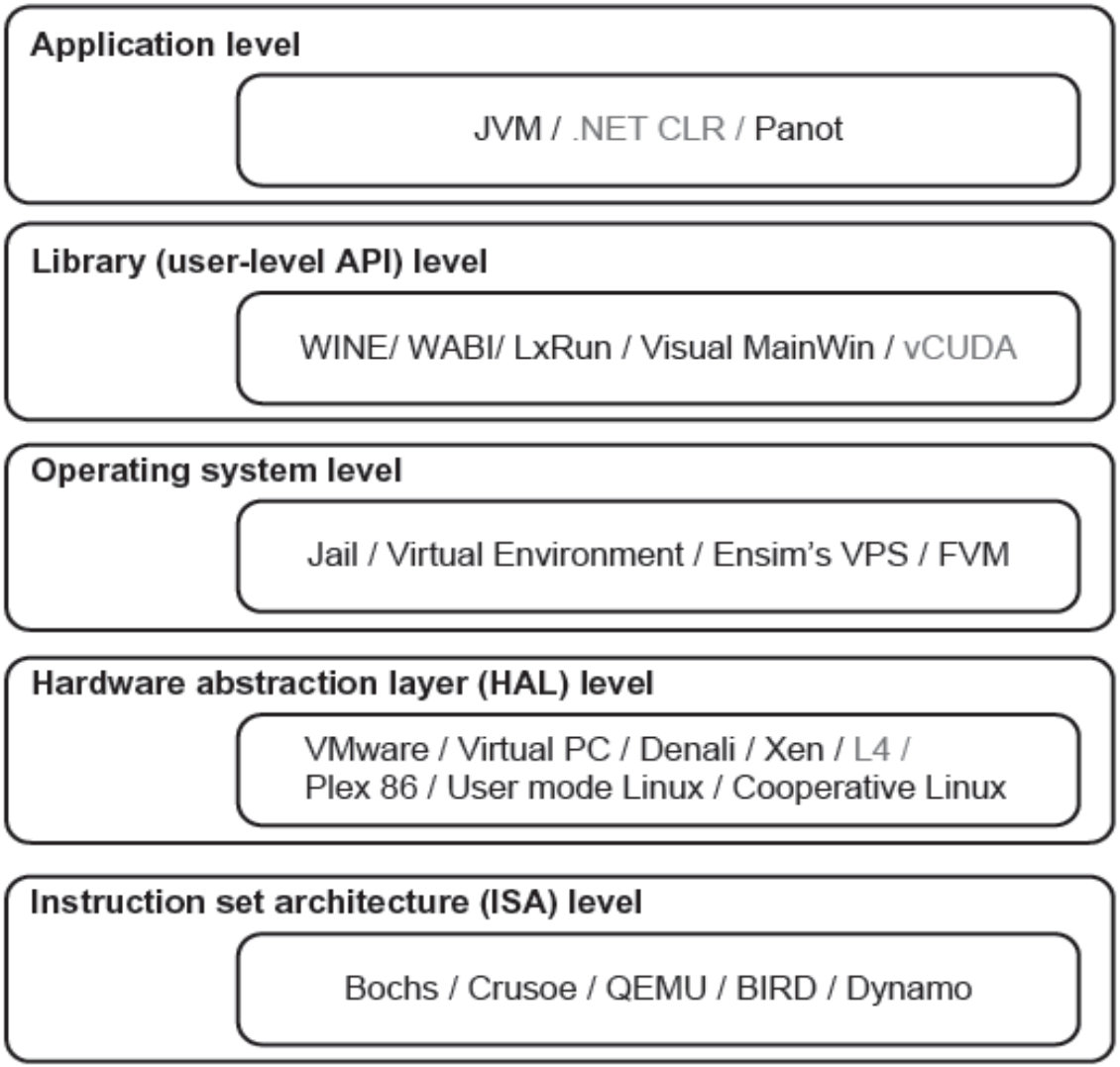

FIGURE 3.2

Virtualization ranging from hardware to applications in five abstraction levels.

1. Instruction Set Architecture Level

-At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine.

-For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation.

-With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine.

-The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one.

- One source instruction may require tens or hundreds of native target instructions to perform its function.

.

Instruction Set Architecture Level

- A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

-One source instruction may require tens or hundreds of native target instructions to perform its function.

-Dynamic binary translation translates basic blocks of dynamic source instructions to target instructions.

-. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

2. Hardware Abstraction Level

Hardware-level virtualization is performed right on top of the bare hardware.

-The idea is to virtualize a computer's resources, such as its **processors, memory, and I/O devices.**

-The intention is to upgrade the hardware utilization rate by multiple users concurrently.

- The Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

3. Operating System Level

-This refers to an abstraction layer between traditional OS and user applications.

-OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers.

-The containers behave like real servers.

- OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

4. Library Support Level

-Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS.

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

- Example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

5. User-Application Level

-Virtualization at the application level virtualizes an application as a VM.

-On a traditional OS, an application often runs as a process.

-Therefore, application-level virtualization is also known as process-level virtualization.

-The most popular approach is to deploy high level language (HLL)VMs.

6. Relative Merits of Different Approaches

-In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition.

- Any program written in the HLL and compiled for this VM will be able to run on it. Java Virtual Machine (JVM) is a good examples of this class of VM.

6. Relative Merits of Different Approaches

-The number of X's in the table cells reflects the advantage points of each implementation level.

- Five X's implies the best case and one X implies the worst case.
-Overall, hardware and OS support will yield the highest performance. However, the hardware and application levels are also the most expensive to implement.

-User isolation is the most difficult to achieve.

- ISA implementation offers the best application flexibility.

6. Relative Merits of Different Approaches

-Table 3.1 compares the relative merits of implementing virtualization at various levels.

-“Implementation Complexity” implies the cost to implement that particular virtualization level.

- “Application Isolation” refers to the effort required to isolate resources committed to different VMs.

6. Relative Merits of Different Approaches

Table 3.1 Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

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 resource not explicitly allocated to it;

(3) it is possible under certain circumstances for a VMM to regain control of resources already allocated.

- If a processor is not designed to support virtualization primarily, it is necessary to modify the hardware to satisfy the three requirements for a VMM. This is known as hardware-assisted virtualization.

Table 3.2 Comparison of Four VMM and Hypervisor Software Packages

Provider and References	Host CPU	Host OS	Guest OS	Architecture
VMware Workstation [71]	x86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, Netware, OS/2, SCO, BeOS, Darwin	Full Virtualization
VMware ESX Server [71]	x86, x86-64	No host OS	The same as VMware Workstation	Para-Virtualization
Xen [7,13,42]	x86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, Windows XP and 2003 Server	Hypervisor
KVM [31]	x86, x86-64, IA-64, S390, PowerPC	Linux	Linux, Windows, FreeBSD, Solaris	Para-Virtualization

The vCUDA for Virtualization of General-Purpose GPUs

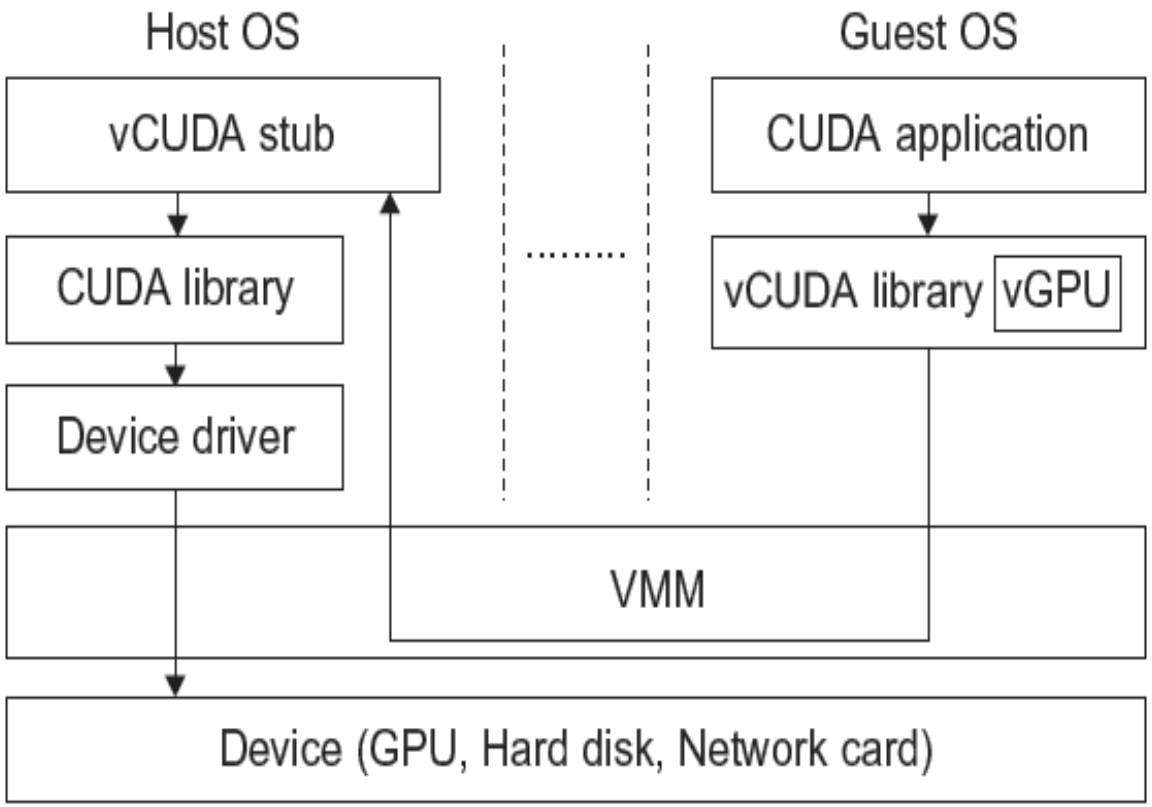

FIGURE 3.4

Basic concept of the vCUDA architecture.

-vCUDA virtualizes the CUDA library and can be installed onguest OSes.

- When CUDA applications run on a guest OS and issue a call to the CUDA API, vCUDA intercepts the call and redirects it to the CUDA API running on the host OS.

Figure 3.4 shows the basicconcept of the vCUDA architecture .
The vCUDA employs a client-server model to implement CUDA virtualization.
It consists of three userspace components:

the vCUDA library,
a virtual GPU in the guest OS (which acts as a client), and the
vCUDA stub in the host OS (which acts as a server).

- The vCUDA library resides in the guest OS as a substitute for the standard CUDA library.
- It is responsible for intercepting and redirecting API calls from the client to the stub.
- Besides these tasks, vCUDA also creates vGPUs and manages them.

The functionality of a vGPU is threefold: It abstracts the GPU structure and gives applications 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 application and notify the remote stub to allocate the
real device memory, and the vGPU is responsible for storing the CUDA API flow.

-The vCUDA stub receives and interprets remote requests and creates a corresponding execution context for the API calls from the guest OS, then returns the results to the guest OS.

- The vCUDA stub also manages actual physical resource allocation.

The functionality of a vGPU is threefold: It abstracts the GPU structure and gives applications 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 application and notify the remote stub to allocate the
real device memory, and the vGPU is responsible for storing the CUDA API flow.

-The vCUDA stub receives and interprets remote requests and creates a corresponding execution context for the API calls from the guest OS, then returns the results to the guest OS.

- The vCUDA stub also manages actual physical resource allocation.

3.2 VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

-Depending on the position of the virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, paravirtualization, and host-based virtualization.

- The hypervisor is also known as the VMM (Virtual Machine Monitor). They both perform the same virtualization operations.

3.2.1 Hypervisor and Xen Architecture

3.2 VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

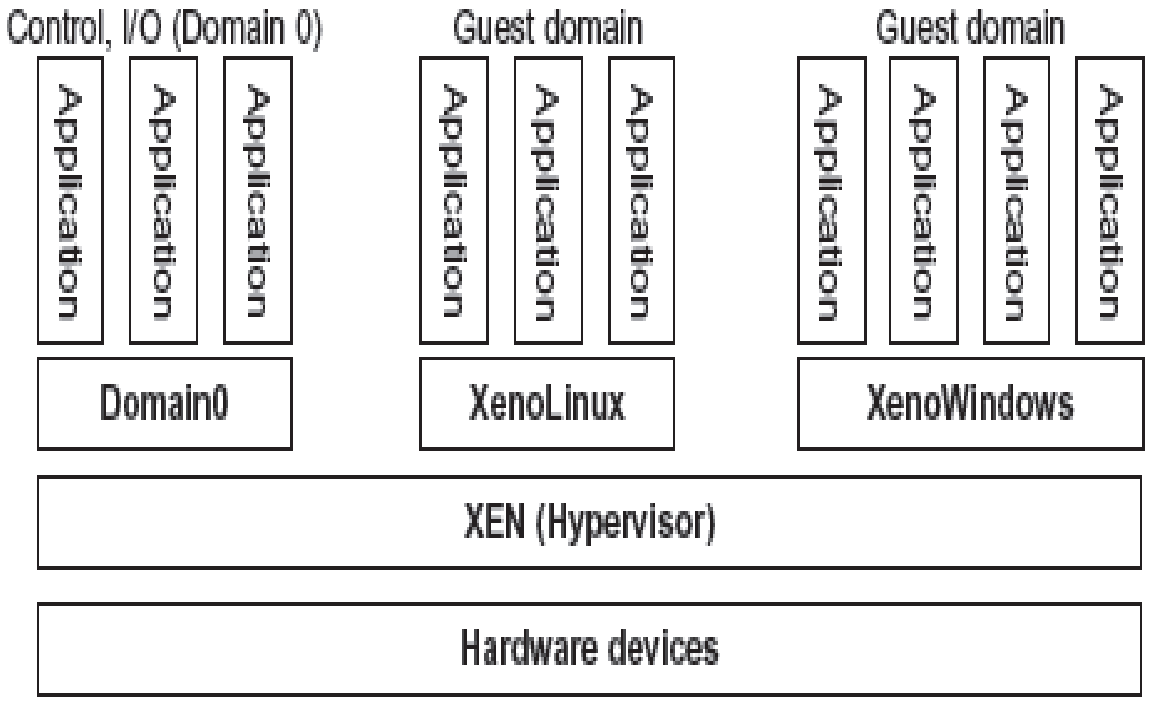

FIGURE 3.5

The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

3.2 VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

The Xen Architecture

-Xen is an open source hypervisor program developed by Cambridge University. Xen is a microkernel.

hypervisor, which separates the policy from the mechanism.

-The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in Figure 3.5.

- 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.
- As a result, the size of the Xen hypervisor is keptrather small.
- Xen provides a virtual environment located between the hardware and the OS.
- A number of vendors are in the process of developing commercial Xen hypervisors, among them
are Citrix XenServer and Oracle VM.
-

3.2 VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

The core components of a Xen system are the hypervisor, kernel, and applications.

-The guest OS, which has control ability, is called Domain 0, and theothers are called Domain U. Domain 0 is a privileged guest OS of Xen. It is first loaded when Xenboots without any file system drivers being available.

-Domain 0 is designed to access hardwaredirectly and manage devices. Therefore, one of the responsibilities of Domain 0 is to allocate andmap hardware resources for the guest domains (the Domain U domains).

- Domain 0, behaving as a VMM, allows users tocreate, copy, save, read, modify, share, migrate, and roll back VMs as easily as manipulating a file,which flexibly provides tremendous benefits for users.