Module-3: Virtualization

1 Introduction

2 Characteristics of virtualized environments

3 Taxonomy of virtualization techniques

3.1 Execution Virtualization

1. Machine reference model

2 Hardware-level virtualizations

a. Hypervisors

3 Hardware virtualization techniques

a. Hardware-assisted Virtualization

b. Full virtualization

c. Paravirtualization

d. Partial virtualization

4 Operating system-level virtualizations

5 Programming language-level virtualization

6 Application-level virtualizations

a. Interpretation

b. Binary Translation

3.2 Other types of Virtualizations

1. Storage Virtualization
2. Network Virtualization
3. Desktop Virtualization
4. Application-Server Virtualization

4 Virtualization and cloud computing

5 Pros and cons of virtualization

6 Technology examples

1. Xen: Para virtualization
2. VMware: Full virtualization
3. Microsoft Hyper – V

1. Introduction

Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure-based services. Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are untrusted, without affecting other users' applications. The basis of this technology is the ability of a computer program—or a combination of software and hardware—to emulate an executing environment separate from the one that hosts such programs.

Virtualization is a large umbrella of technologies and concepts that are meant to provide an abstract environment—whether virtual hardware or an operating system—to run applications. The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing.

Virtualization technologies have gained renewed interested recently due to the confluence of several phenomena:

(a) Increased performance and computing capacity.

The high-end side of the PC market, where supercomputers can provide immense compute power that can accommodate the execution of hundreds or thousands of virtual machines.

(b) Underutilized hardware and software resources.

Hardware and software underutilization is occurring due to (1) increased performance and computing capacity, and (2) the effect of limited or sporadic use of resources.

Computers today are so powerful that in most cases only a fraction of their capacity is used by an application or the system. Using these resources for other purposes after hours could improve the efficiency of the IT infrastructure.

(c) Lack of space.

Companies such as Google and Microsoft expand their infrastructures by building data centers as large as football fields that are able to host thousands of nodes. Although this is viable for IT giants, in most cases enterprises cannot afford to build another data center to accommodate additional resource capacity. This condition, along with hardware underutilization, has led to the diffusion of a technique called server consolidation

(d) Greening initiatives.

Maintaining a data center operation not only involves keeping servers on, but a great deal of energy is also consumed in keeping them cool. Infrastructures for cooling have a significant impact on the carbon footprint of a data center. Hence, reducing the number of servers through server consolidation will definitely reduce the impact of cooling and power consumption of a data center. Virtualization technologies can provide an efficient way of consolidating servers.

(e) Rise of administrative costs.

The increased demand for additional capacity, which translates into more servers in a data center, is also responsible for a significant increment in administrative costs. Computers—in particular, servers—do not operate all on their own, but they require care and feeding from system administrators.

These are labor-intensive operations, and the higher the number of servers that have to be managed, the higher the administrative costs. Virtualization can help reduce the number of required servers for a given workload, thus reducing the cost of the administrative personnel.

2. Characteristics of virtualized environments

Virtualization is a broad concept that refers to the creation of a virtual version of something, whether hardware, a software environment, storage, or a network. In a virtualized environment there are three major components: guest, host, and virtualization layer. The guest represents the system component that interacts with the virtualization layer rather than with the host, as would normally happen. The host represents the original environment where the guest is supposed to be managed. The virtualization layer is responsible for recreating the same or a different environment where the guest will operate (see Figure 3.1).

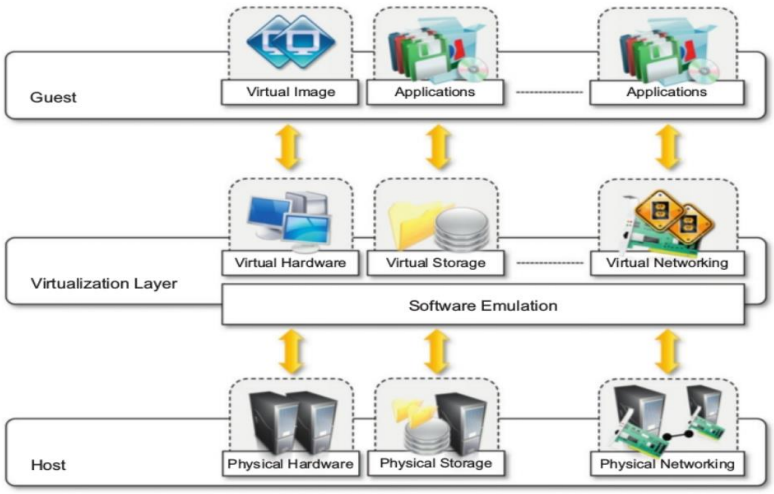

FIGURE 3.1
The virtualization reference model.

The characteristics of virtualized solutions are:

1 Increased security

2 Managed executions

3 Portability

1. Increased security

The virtual machine represents an emulated environment in which the guest is executed. All the operations of the guest are generally performed against the virtual machine, which then translates and applies them to the host. This level of indirection allows the virtual machine manager to control and filter the activity of the guest, thus preventing some harmful operations from being performed. For example, applets downloaded from the Internet run in a sandboxed 3 version of the Java Virtual Machine (JVM), which provides them with limited access to the hosting operating system resources. Both the JVM and the .NET runtime provide extensive security policies for customizing the execution environment of applications.

2 Managed executions

Virtualization of the execution environment not only allows increased security, but a wider range of features also can be implemented. In particular, sharing, aggregation, emulation, and isolation are the most relevant features (see Figure 3.2).

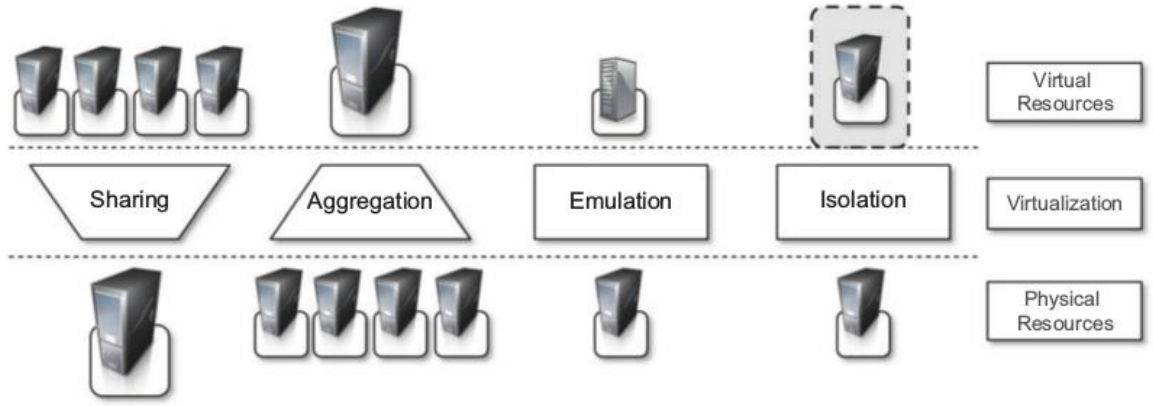

FIGURE 3.2

Functions enabled by managed execution.

(a) Sharing - Virtualization allows the creation of a separate computing environments within the same host. In this way it is possible to fully exploit the capabilities of a powerful guest, which would otherwise be underutilized.

(b)Aggregation - Not only is it possible to share physical resource among several guests, but virtualization also allows aggregation, which is the opposite process. A group of separate hosts can be tied together and represented to guests as a single virtual host.

(c)Emulation - Guest programs are executed within an environment that is controlled by the virtualization layer, which ultimately is a program. This allows for controlling and tuning the environment that is exposed to guests. For instance, a completely different environment with respect to the host can be emulated, thus allowing the execution of guest programs requiring specific characteristics that are not present in the physical host.

(d) Isolation - Virtualization allows providing guests—whether they are operating systems, applications, or other entities—with a completely separate environment, in which they are executed. The guest program performs its activity by interacting with an abstraction layer, which provides access to the underlying resources.

3 Portability

The concept of portability applies in different ways according to the specific type of virtualization considered. In the case of a hardware virtualization solution, the guest is packaged into a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines.

In the case of programming-level virtualization, as implemented by the JVM or the .NET runtime, the binary code representing application components (jars or assemblies) can be run without any recompilation on any implementation of the corresponding virtual machine. This makes the application development cycle more flexible and application deployment very straightforward: One version of the application, in most cases, is able to run on different platforms with no changes.

3. Taxonomy of virtualization techniques

Virtualization covers a wide range of emulation (computing action of a different computer, software system) techniques that are applied to different areas of computing.

A classification of these techniques helps us better understand their characteristics and use (see Figure 3.3).

The first classification discriminates against the service or entity that is being emulated.

Virtualization is mainly used to emulate execution environments, storage, and networks. Among these categories, execution virtualization constitutes the oldest, most popular, and most developed area. Therefore, it deserves major investigation and a further categorization

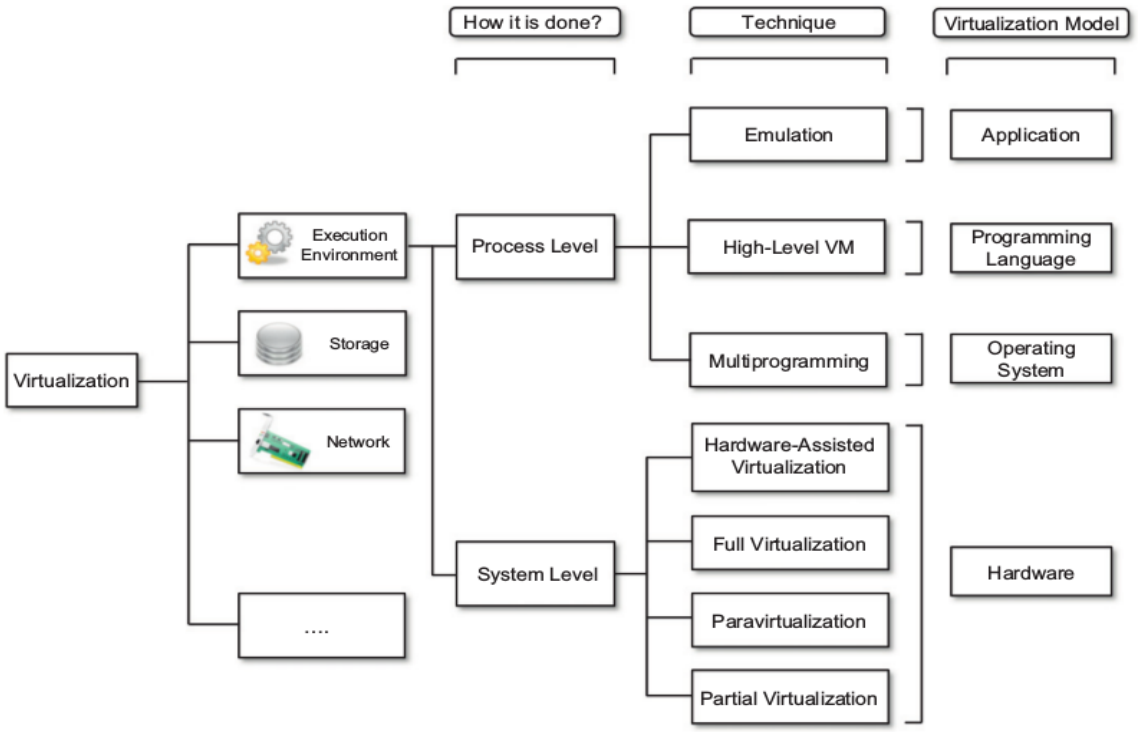

FIGURE 3.3
A taxonomy of virtualization techniques.

3.1 Execution virtualization

Execution virtualization includes all techniques that aim to emulate an execution environment that is separate from the one hosting the virtualization layer.

All these techniques concentrate their interest on providing support for the execution of programs, whether these are the operating system, a binary specification of a program compiled against an abstract machine model, or an application.

1 Machine reference model
2 Hardware-level virtualizations
    a. Hypervisors
3 Hardware virtualization techniques
    a. Hardware-assisted Virtualization
    b. Full virtualization
    c. Paravirtualization
    d. Partial virtualization
4 Operating system-level virtualizations
5 Programming language-level virtualization
6 Application-level virtualizations
    a. Interpretation
    b. Binary Translation

1 Machine reference model

Modern computing systems can be expressed in terms of the reference model described in Figure 3.4.

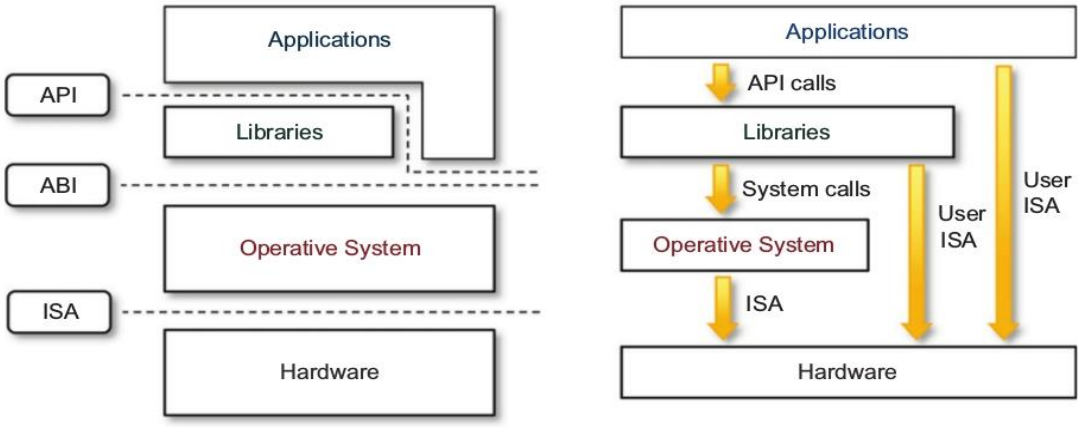

FIGURE 3.4
A machine reference model.

- At the bottom layer, the model for the hardware is expressed in terms of the **Instruction Set Architecture (ISA)**, which defines the instruction set for the processor, registers, memory, and interrupt management.
- **ISA** is the interface between hardware and software, and it is important to the operating system (OS) developer (System ISA) and developers of applications that directly manage the underlying hardware (User ISA).
- The **application binary interface (ABI)** separates the operating system layer from the applications and libraries, which are managed by the OS.
- ABI covers details such as low-level data types, alignment, and call conventions and defines a format for executable programs.
- The highest level of abstraction is represented by the **application programming interface (API)**, which interfaces applications to libraries and/or the underlying operating system.

The **instruction set (ISA)** exposed by the hardware has been divided into two different parts as **privileged** and **nonprivileged** instructions.

- **Privileged instructions** are those that are executed under specific **restrictions** and are mostly used for **sensitive operations**, which expose (behavior-sensitive) or modify (control-sensitive) the privileged state.

- **Nonprivileged instructions** are those instructions that can be used **without interfering** with other tasks because they **do not access shared resources**.

A possible implementation features a hierarchy of privileges (see Figure 3.5) in the form of ring-based security: Ring 0, Ring 1, Ring 2, and Ring 3;

- Ring 0 is in the most privileged level and Ring 3 in the least privileged level.
- **Ring 0** is used by the **kernel of the OS** (Supervisor)**.**
- **Rings 1 and 2** are used by the **OS-level services** (hypervisor) and
- **Ring 3** is used by the user.

Recent systems support only two levels, with Ring 0 for supervisor mode and Ring 3 for user mode.

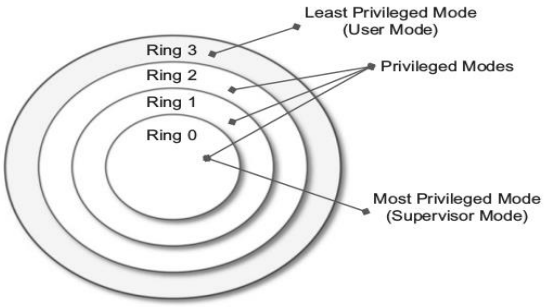

FIGURE 3.5
Security rings and privilege modes.

2 Hardware-level virtualizations

- It provides an abstract execution environment in terms of **computer hardware** on **top** of which a guest **operating system** can be run.
- The guest is represented by the operating system, the host by the physical computer hardware, the virtual machine by its emulation, and the virtual machine manager by the **hypervisor** (see Figure 3.6).
- The hypervisor is generally a program or a combination of software and hardware that allows the abstraction of the underlying physical hardware.
- **Hardware-level virtualization is also called system virtualization**, since it provides **ISA** to **virtual machines**, which is the representation of the hardware interface of a system.

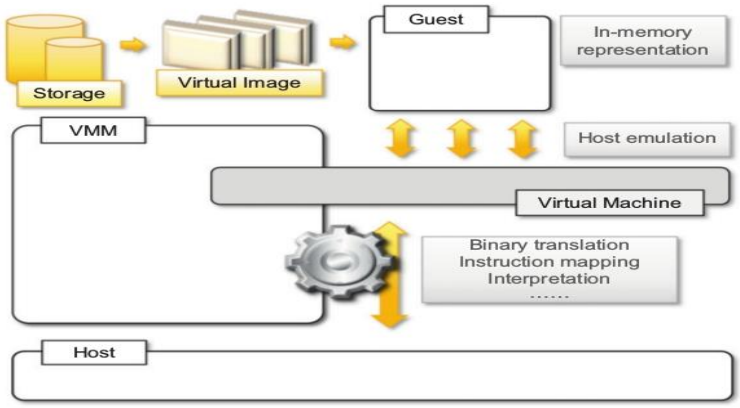

FIGURE 3.6
A hardware virtualization reference model.

a. Hypervisors

A fundamental element of hardware virtualization is the **hypervisor**, or virtual machine manager (**VMM**). It recreates a hardware environment in which guest operating systems are installed. There are two major types of hypervisors: Type I and Type II (see Figure 3.7).

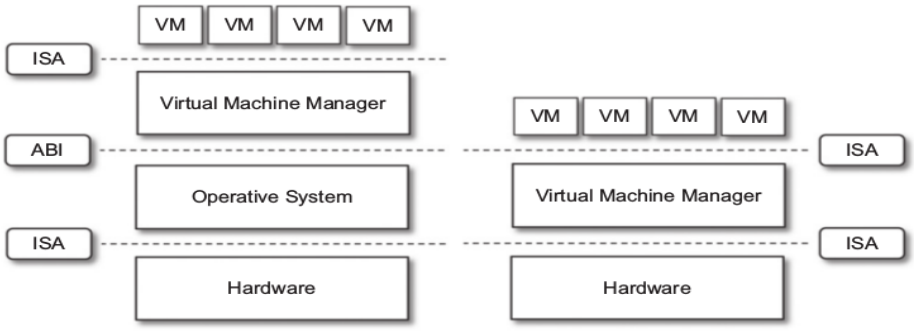

FIGURE 3.7
Hosted (left) and native (right) virtual machines. This figure provides a graphical representation of the two types of hypervisors.

- **Type I hypervisors (native virtual machine)** It run directly on top of the hardware. Therefore, they take the place of the operating systems and interact directly with the ISA interface exposed by the underlying hardware, and they emulate this interface to allow the management of guest operating systems. This type of hypervisor is also called a **native virtual machine** since it runs natively on hardware.
- **Type II hypervisors (hosted virtual machine)** It requires the support of an operating system to provide virtualization services. This means that they are programs managed by the operating system, which interact with it through the ABI and emulate the ISA of virtual hardware for guest operating systems. This type of hypervisor is also called a **hosted virtual machine** since it is hosted within an operating system.
- **Virtual machine manager (VMM)** is internally organized as described in Figure 3.8.

Three main modules, **dispatcher**, **allocator**, and **interpreter**, coordinate their activity to emulate the underlying hardware.

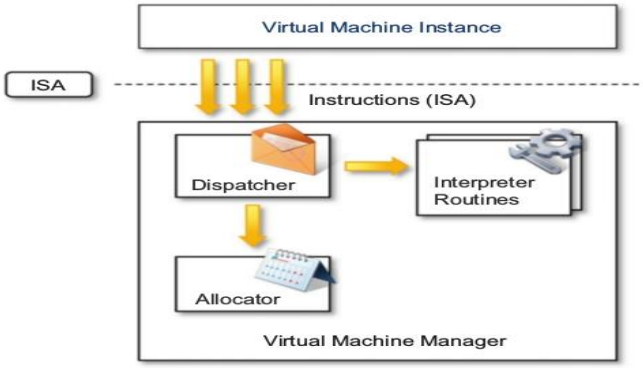

FIGURE 3.8
A hypervisor reference architecture.

- The **dispatcher** constitutes the entry point of the monitor and reroutes the instructions issued by the virtual machine instance to one of the two other modules.
- The **allocator** is responsible for deciding the **system resources to be provided to the VM**.
- The **interpreter module** consists of interpreter routines. These are executed whenever a virtual machine executes a privileged instruction.

The design and architecture of a virtual machine manager, together with the underlying hardware design of the host machine, determine the full realization of hardware virtualization, where a guest operating system can be transparently executed on top of a VMM as though it were run on the underlying hardware. The criteria that need to be met by a virtual machine manager to efficiently support virtualization were established by Goldberg and Popek in 1974 [23].

Three properties must be satisfied:

1. Equivalence - A guest running under the control of a virtual machine manager should exhibit the same behaviour as when it is executed directly on the physical host.

2. Resource control - The virtual machine manager should be in complete control of virtualized resources.

3. Efficiency - A statistically dominant fraction of the machine instructions should be executed without intervention from the virtual machine manager.

Popek and Goldberg provided a classification of the instruction set and proposed three theorems that define the properties that hardware instructions need to satisfy in order to efficiently support virtualization.

THEOREM 3.1

For any conventional third-generation computer, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

This theorem establishes that all the instructions that change the configuration of the system resources should generate a trap in user mode and be executed under the control of the virtual machine manager.

THEOREM 3.2

A conventional third-generation computer is recursively virtualizable if:

- It is virtualizable and
- A VMM without any timing dependencies can be constructed for it.

Recursive virtualization is the ability to run a virtual machine manager on top of another virtual machine manager. This allows nesting hypervisors as long as the capacity of the underlying resources can accommodate that. Virtualizable hardware is a prerequisite to recursive virtualization.

THEOREM 3.3

A hybrid VMM may be constructed for any conventional third-generation machine in which the set of user-sensitive instructions is a subset of the set of privileged instructions.

There is another term, hybrid virtual machine (HVM), which is less efficient than the virtual machine system. In the case of an HVM, more instructions are interpreted rather than being executed directly. All instructions in virtual supervisor mode are interpreted. Whenever there is an attempt to execute a behavior-sensitive or control-sensitive instruction, HVM controls the execution directly or gains the control via a trap. Here all sensitive instructions are caught by HVM that are simulated.

3 Hardware virtualization techniques (System Level techniques)

- Hardware-assisted virtualization
- Full virtualization
- Paravirtualization
- Partial virtualization

- **Hardware-assisted virtualization (Hardware – Processors)**
  This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation.

  This technique was originally introduced in the IBM System/370. **Examples** of hardware-assisted virtualization are the extensions to the **x86-64**bit architecture introduced with Intel VT, and AMD.

Intel and AMD introduced processor extensions, and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others.

- **Full virtualization (Running Operating System – no modified OS)**
  Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware.

  To make this possible, virtual machine manager are required to provide a complete emulation of the entire underlying hardware. The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform.

- **Paravirtualization (Thin virtual machine - modified OS)**
  Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, therefore, guests need to be modified. The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host

- **Partial virtualization**
  Partial virtualization provides a partial emulation of the underlying hard- ware, thus not allowing the complete execution of the guest operating system in complete isolation. Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported.

4 Operating system-level virtualizations

- Operating system-level virtualization offers the opportunity to **create different and separated execution environments for applications that are managed concurrently**.
- It is Different from hardware virtualization - there is no virtual machine manager or hypervisor, and the **virtualization is done within a single operating system, where the OS kernel allows for multiple isolated user space instances**.
- The **kernel is also responsible for sharing the system resources** among instances and for limiting the impact of instances on each other.

5 Programming language-level virtualization

- Programming language-level virtualization is mostly used to achieve ease of **deployment of applications, managed execution, and portability across different platforms and operating systems**.
- It consists of a virtual machine **executing the byte code of a program**, which is the result of the compilation process. Compilers implemented and used this technology to produce a binary format representing the machine code for an abstract architecture.

- Programming language-level virtualization has a long trail in computer science history and originally was used in 1966 for the implementation **of Basic Combined Programming Language (BCPL**), a language for writing compilers and one of the ancestors of the C programming language.

6 Application-level virtualizations

- Application-level virtualization is a technique allowing applications to be **run in runtime environments that do not natively support all the features required by such applications**. In this scenario, **applications are not installed in the expected runtime environment but are run as though they were.**
- Emulation can also be used to execute program binaries compiled for different hardware architectures. In this case, one of the following strategies can be implemented:

a. Interpretation. In this technique every source instruction is interpreted by an emulator for executing native ISA instructions, leading to poor performance. Interpretation has a minimal start-up cost but a huge overhead, since each instruction is emulated.

b. Binary translation. In this technique every source instruction is converted to native instructions with equivalent functions. After a block of instructions is translated, it is cached and reused. Binary translation has a large initial overhead cost, but over time it is subject to better performance, since previously translated instruction blocks are directly executed.

3.2 Other types of virtualizations

Other than execution virtualization, other types of virtualizations provide an abstract environment to interact with. These mainly cover storage, networking, and client/server interaction.

1 Storage virtualization

Storage virtualization is a system administration practice that allows decoupling the physical organization of the hardware from its logical representation. Using this technique, users do not have to be worried about the specific location of their data, which can be identified using a logical path. Storage virtualization allows us to harness a wide range of storage facilities and represent them under a single logical file system. There are different techniques for storage virtualization, one of the most popular being network-based virtualization by means of **storage area networks (SANs)**.

2 Network virtualization

Network virtualization combines hardware appliances and specific software for the creation and management of a virtual network. Network virtualization can aggregate different physical networks into a single logical network (external network virtualization) or provide network-like functionality to an operating system partition (internal network virtualization). The result of external network virtualization is generally a virtual LAN (VLAN).

3 Desktop virtualization

Desktop virtualization abstracts the desktop environment available on a personal computer to provide access to it using a client/server approach. Desktop virtualization provides the same outcome of hardware virtualization but serves a different purpose. Similarly, to hardware virtualization, desktop virtualization makes accessible a different system as though it were natively installed on the host, but this system is remotely stored on a different host and accessed through a network connection. Moreover, desktop virtualization addresses the problem of making the same desktop environment accessible from everywhere.

4 Application server virtualization

Application server virtualization abstracts a collection of application servers that provide the same services as a single virtual application server by using load-balancing strategies and providing a high-availability infrastructure for the services hosted in the application server. This is a particular form of virtualization and serves the same purpose of storage virtualization: providing a better quality of service rather than emulating a different environment.

4. Virtualization and Cloud Computing

Virtualization plays an important role in cloud computing since it allows for the appropriate degree of customization, security, isolation, and manageability that are fundamental for delivering IT ser- vices on demand. Particularly important is the role of virtual computing environment and execution virtualization techniques. Among these, hardware and programming language virtualization are the techniques adopted in cloud computing systems.

Besides being an enabler for computation on demand, virtualization also gives the opportunity to design more efficient computing systems by means of consolidation, which is performed transparently to cloud computing service users

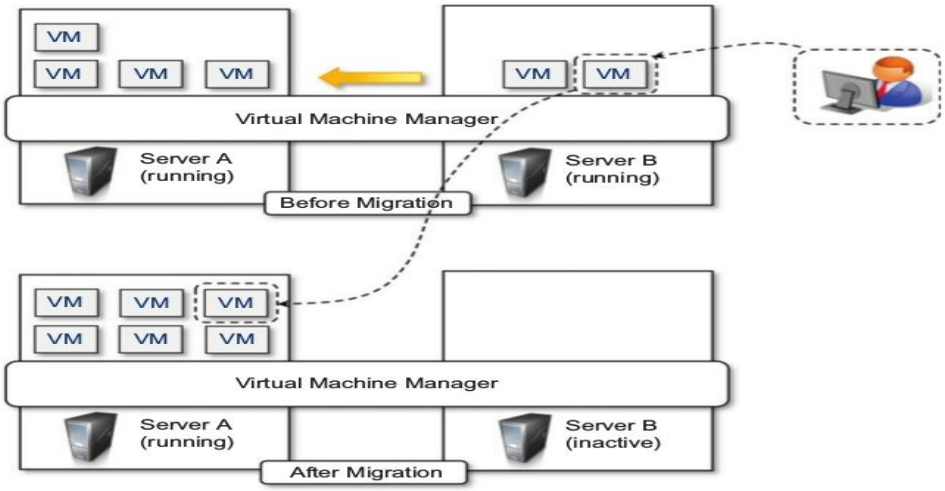

FIGURE 3.10
Live migration and server consolidation.

Since virtualization allows us to create isolated and controllable environments, it is possible to serve these environments with the same resource without them interfering with each other. This opportunity is particularly attractive when resources are underutilized, because it allows reducing the number of active resources by aggregating virtual machines over a smaller number of resources that become fully utilized. This practice is also known as server consolidation, while the movement of virtual machine instances is called virtual machine migration (see Figure 3.10). Because virtual machine instances are controllable environments, consolidation can be applied with a minimum impact, either by temporarily stopping its execution and moving its data to the new resources or by performing a finer control and moving the instance while it is running. This second techniques are known as live migration and in general is more complex to implement but more efficient since there is no disruption of the activity of the virtual machine instance.

5. Pros and Cons of Virtualization

Virtualization has now become extremely popular and widely used, especially in cloud computing. Today, the capillary diffusion of the Internet connection and the advancements in computing technology have made virtualization an interesting opportunity to deliver on-demand IT infrastructure and services.

1. Advantages of virtualization

1. Managed execution and isolation are perhaps the most important advantages of virtualization. In the case of techniques supporting the creation of virtualized execution environments, these two characteristics allow building secure and controllable computing environments.
2. Portability is another advantage of virtualization, especially for execution virtualization techniques. Virtual machine instances are normally represented by one or more files that can be easily transported with respect to physical systems.
3. Portability and self-containment also contribute to reducing the costs of maintenance, since the number of hosts is expected to be lower than the number of virtual machine instances. Since the guest program is executed in a virtual environment, there is very limited opportunity for the guest program to damage the underlying hardware.
4. Finally, by means of virtualization it is possible to achieve a more efficient use of resources. Multiple systems can securely coexist and share the resources of the underlying host, without interfering with each other.

2 The other side of the coin: disadvantages of virtualization

(a) Performance degradation

Performance is one of the major concerns in using virtualization technology. Since virtualization interposes an abstraction layer between the guest and the host, the guest can experience increased latencies (delays).

For instance, in the case of hardware virtualization, where the intermediate emulates a bare machine on top of which an entire system can be installed, the causes of performance degradation can be traced back to the overhead introduced by the following activities:

- Maintaining the status of virtual processors
- Support of privileged instructions (trap and simulate privileged instructions)
- Support of paging within VM
- Console functions

(b) Inefficiency and degraded user experience

Virtualization can sometime lead to an inefficient use of the host. Some of the specific features of the host cannot be exposed by the abstraction layer and then become inaccessible. In the case of hardware virtualization, this could happen for device drivers: The virtual machine can sometime simply provide a default graphic card that maps only a subset of the features available in the host. In the case of programming-level virtual machines, some of the features of the underlying operating systems may become inaccessible unless specific libraries are used.

(c) Security holes and new threats

Virtualization opens the door to a new and unexpected form of phishing. The capability of emulating a host in a completely transparent manner led the way to malicious programs that are designed to extract sensitive information from the guest. The same considerations can be made for programming-level virtual machines: Modified ver- sions of the runtime environment can access sensitive information or monitor the memory locations utilized by guest applications while these are executed.

6. Technology Examples

1. Xen: Paravirtualization

2. VMware: Full Virtualization

a. Full virtualization and Binary Translation
b. Virtualization Solutions
   i. End-user (Desktop) Virtualization
   ii. Server Virtualization
   iii. Infrastructure Virtualization and Cloud-Computing Solutions

3. Microsoft Hyper - V

a. Architecture
   i. Hypervisor
   ii. Enlightened I/O and Synthetic Devices
   iii. Parent Partition
   iv. Children Partitions
b. Cloud Computing and Infrastructure Management

1. Xen: Paravirtualization

Xen is an open-source **hypervisor** based on **paravirtualization**. It is the most popular application of paravirtualization. Xen has been extended to compatible with full virtualization using hardware-assisted virtualization. It enables high performance to execute **guest operating system**.

This is probably done by **removing the performance loss while executing the instructions requiring significant handling and by modifying portion of the guest operating system executed by Xen,** with reference to the execution of such instructions.

Hence this especially support **x86**, which is the most used architecture on commodity machines and servers

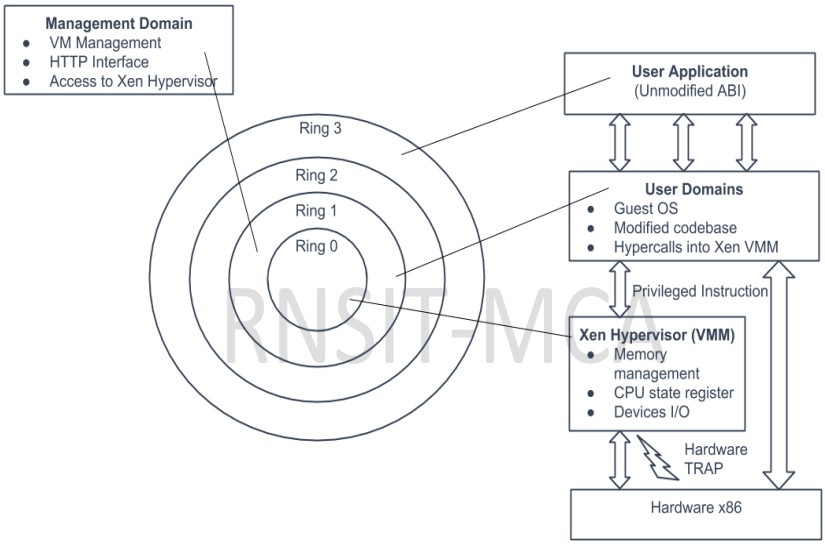

Figure – Xen Architecture and Guest OS Management

Above figure describes the Xen Architecture and its mapping onto a **classic x86 privilege model**. A Xen based system is handled by Xen hypervisor, which is executed in the most privileged mode and maintains the access of guest operating system to the basic hardware. Guest operating system are run between domains, which represents virtual machine instances.

In addition, **particular control software**, which has **privileged access** to the host and handles all other guest OS, runs in a special domain called Domain 0. This the only one loaded once the virtual machine manager has fully booted and hosts an HTTP server that delivers requests for virtual machine creation, configuration, and termination.

This component establishes the primary version of a shared virtual machine manager (VMM), which is a necessary part of Cloud computing system delivering Infrastructure-as-a-Service (IaaS) solution.

Various x86 implementation support four distinct security levels, termed as rings, i.e., **Ring 0, Ring 1, Ring 2, Ring 3.**

Here, Ring 0 represents the level having most privilege and Ring 3 represents the level having least privilege. Almost all the frequently used Operating system uses only two levels i.e., Ring 0 for the Kernel code and Ring 3 for user application and non-privilege OS program. This provides a chance to the Xen to implement paravirtualization. This enables Xen to control unchanged the Application Binary Interface (ABI) thus allowing a simple shift to Xen-virtualized solutions, from an application perspective.

Due to the structure of x86 instruction set, some instructions allow code execution in Ring 3 to switch to Ring 0 (Kernel mode). Such an operation is done at hardware level, and hence between a virtualized environment, it will lead to a TRAP or a silent fault, thus preventing the general operation of the guest OS as it is now running in Ring 1.

This condition is basically occurred by a subset of system calls. To eliminate this situation, implementation in operating system requires a modification and all the sensitive system calls needs re-implementation with hypercalls. Here, hypercalls are the particular calls revealed by the virtual machine (VM) interface of Xen and by use of it, Xen hypervisor tends to catch the execution of all the sensitive instructions, manage them, and return the control to the guest OS with the help of a supplied handler.

Paravirtualization demands the OS codebase be changed, and hence all operating systems cannot be referred to as guest OS in a Xen-based environment. This condition holds where hardware-assisted virtualization cannot be free, which enables to run the hypervisor in Ring 1 and the guest OS in Ring 0. Hence, Xen shows some limitations in terms of legacy hardware and in terms of legacy OS.

In fact, these are not possible to modify to be run in Ring 1 safely as their codebase is not reachable, and concurrently, the primary hardware hasn't any support to execute them in a more privileged mode than Ring 0. Open source OS like Linux can be simply modified as its code is openly available, and Xen delivers full support to virtualization, while components of Windows are basically not compatible with Xen, unless hardware-assisted virtualization is available. As new releases of OS are designed to be virtualized, the problem is getting resolved and new hardware supports x86 virtualization.

Pros:

a) Xen server is developed over open-source Xen hypervisor and it uses a combination of hardware-based virtualization and paravirtualization. This tightly coupled collaboration between the operating system and virtualized platform enables the system to develop lighter and flexible hypervisor that delivers their functionalities in an optimized manner.

b) Xen supports balancing of large workload efficiently that capture CPU, Memory, disk input-output and network input-output of data. It offers two modes to handle this workload: Performance enhancement, and For handling data density.

c) It also comes equipped with a special storage feature that we call Citrix storage link. Which allows a system administrator to uses the features of arrays from Giant companies- Hp, Netapp, Dell Equal logic etc.

d) It also supports multiple processors, live migration one machine to another, physical server to virtual machine or virtual server to virtual machine conversion tools, centralized multi server management, real time performance monitoring over window and Linux.

Cons:

a) Xen is more reliable over linux rather than on window.

b) Xen relies on 3rd-party component to manage the resources like drivers, storage, backup, recovery & fault tolerance.

c) Xen deployment could be a burden some on your Linux kernal system as time passes.

d) Xen sometimes may cause increase in load on your resources by high input-output rate and and may cause starvation of other Vm's.

2. VMware: Full Virtualization

In full virtualization primary hardware is replicated and made available to the guest operating system, which executes unaware of such abstraction and no requirements to modify. Technology of VMware is based on the key concept of Full Virtualization. Either in desktop environment, with the help of type-II hypervisor, or in server environment, through type-I hypervisor, VMware implements full virtualization. In both the cases, full virtualization is possible through the direct execution for non-sensitive instructions and binary translation for sensitive instructions or hardware traps, thus enabling the virtualization of architecture like x86.

(a) Full Virtualization and Binary Translation

VMware is widely used as it tends to virtualize x86 architectures, which executes unmodified on-top of their hypervisors. With the introduction of hardware-assisted virtualization, full virtualization is possible to achieve by support of hardware. But earlier, x86 guest operating systems unmodified in a virtualized environment could be executed only with the use of dynamic binary translation.

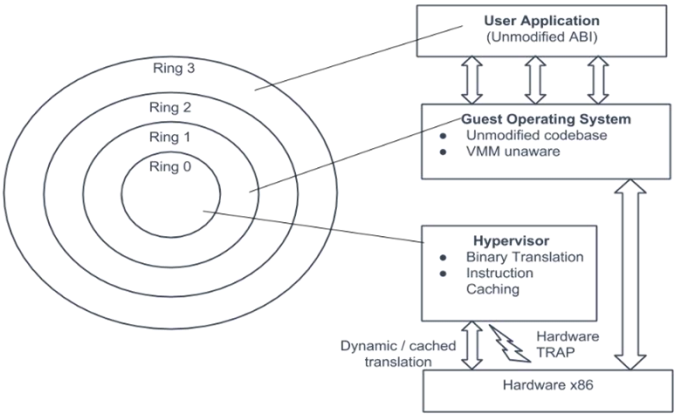

Figure – Full Virtualization Reference Model

The major benefit of this approach is that guests can run unmodified in a virtualized environment, which is an important feature for operating system whose source code does not exist. Binary translation is portable for full virtualization. As well as translation of instructions at runtime presents an additional overhead that is not existed in other methods like

paravirtualization or hardware-assisted virtualization. Contradict, binary translation is only implemented to a subset of the instruction set, while the others are managed through direct execution on the primary hardware. This depletes somehow the impact on performance of binary translation.

Advantages of Binary Translation –

1. This kind of virtualization delivers the best isolation and security for Virtual Machine.
2. Truly isolated numerous guest OS can execute concurrently on the same hardware.
3. It is only implementation that needs no hardware assist or operating system assist to virtualize sensitive instruction as well as privileged instruction.

Disadvantages of Binary Translation –

1. It is time consuming at run-time.
2. It acquires a large performance overhead.
3. It employs a code cache to stock the translated most used instructions to enhance the performance, but it increases memory utilization along with the hardware cost.
4. The performance of full virtualization on the x86 architecture is 80 to 95 percent that of the Host machines.

(b) Virtualization Solutions

i. End-user (Desktop) Virtualization

VMware supports virtualization of operating system environments and single applications on end-user's computers. Specific VMware software – VMware Workstation, for windows operating systems and VMware Fusion, for Mac OS X environments – is installed in the host operating system to create virtual machines and manage their execution. Besides the creation of an isolated computing environment, the two products allow a guest operating system to leverage the resources of the host machine (USB devices, folder sharing and integration with the GUI of the host operating system. Fig. 3.13 provides an overview of the architecture of these systems.

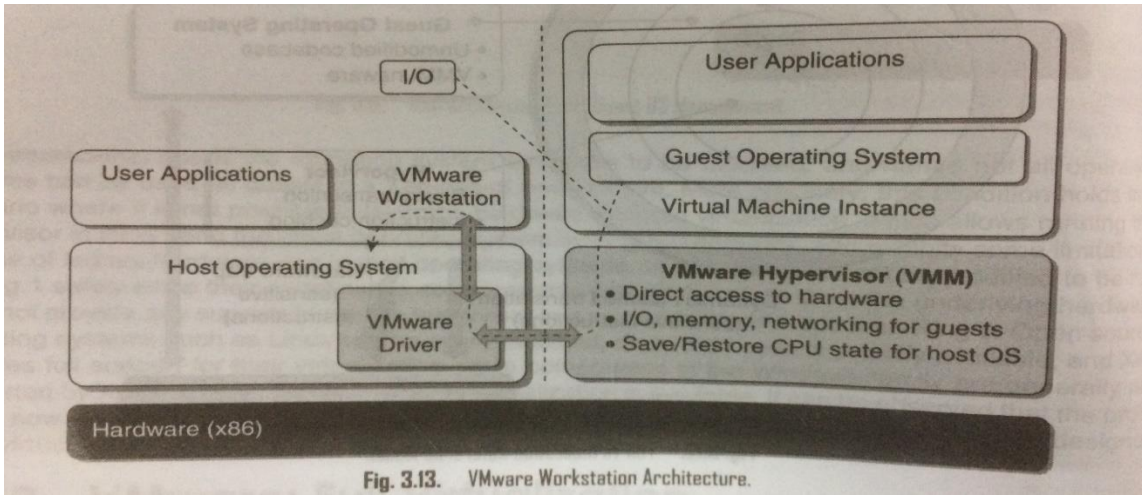

Fig. 3.13. VMware Workstation Architecture.

The virtualization environments is created by an application installed in the guest operating system, which provides the guest operating system with full hardware virtualization of the underlying hardware. This is done by installing a specific driver in the operating system that provides two main services.

- It deploys a virtual machine manager that can run in privileged mode.
- It provides hooks for the VMware application to process specific I/O requests eventually by relaying such requests to the host operating system via system calls.

ii. Server Virtualization

Initial support for server virtualization was provided by VMware GSX server, which replicates the approach used for end-user computers and introduces remote management and scripting capabilities. The architecture of VMware GSX server is depicted in Fig. 3.14.

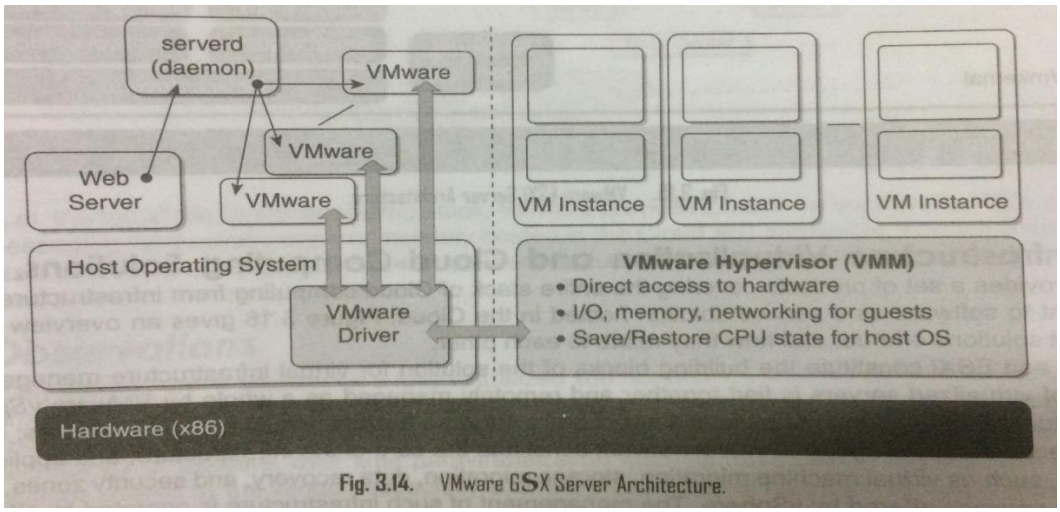

Fig. 3.14. VMware GSX Server Architecture.

The architecture is designed to serve the virtualization of web servers. A daemon process called served, controls, and manages VMware application processes. These applications are then connected to the virtual machines instances by means of the VMware driver installed on the host operating system. Virtual machine instances are managed by the VMM as described previously. User request for virtual machine management and provisioning are routed from the web server through the VMM by means of served.

iii. Infrastructure virtualization and cloud computing solutions

VMware provides a set of products covering the entire stack of cloud computing from infrastructure management to software as a service solution hosted in the cloud. Fig. 3.16 gives an overview of the different solutions offered and how they relate to each other.

A collection of virtualized data centers are turned into a infrastructure-as-a-service cloud by VMware **vCloud**, which allows service providers to make available to end users a virtual computing environments, on demand, on a pay-per-use basis.

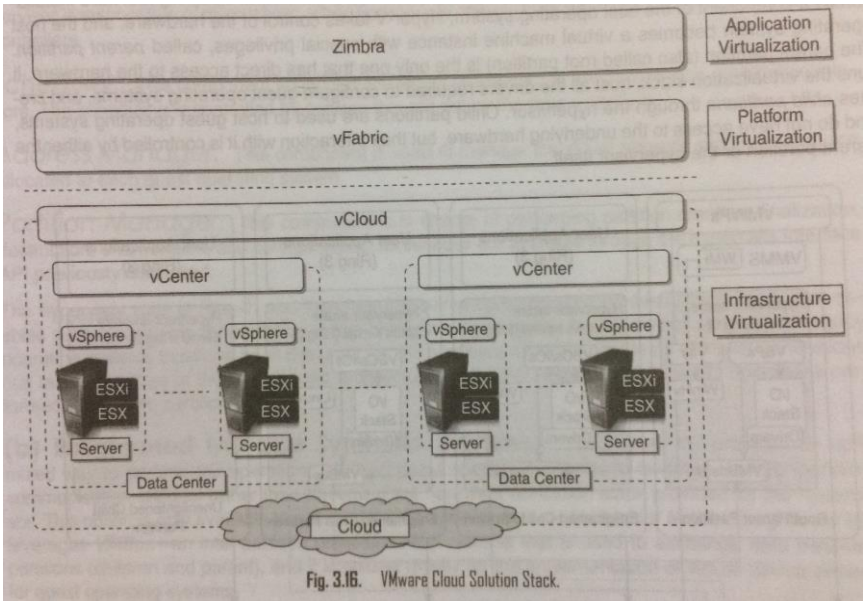

Fig. 3.16. VMware Cloud Solution Stack.

vFabric is a collection of components for application monitoring, scalable data management, and scalable execution and provisioning of java web applications.

Zimra a solution for office automation, messaging, and collaboration completely hosted in the cloud and accessible from anywhere. This is a SaaS solution that integrates together different features into a single software platform, providing email and collaboration management.

3. Microsoft Hyper - V

Hyper – V is an infrastructure virtualization solution developed by Microsoft for server virtualization. As the name recalls, it uses a hypervisor-based approach for hardware virtualization, which leverages several techniques to support a variety 2008 R2 that installs the hypervisor as a role within the server.

1. Architecture

Hyper – V supports multiple and concurrent execution of the guest operating system by means of partitions. A partition is a completely isolated environment in which an operating system is installed and run.

Fig. 3.17 provides an overview of the architecture of Hyper – V. Hyper – V takes control of the hardware, and the host operating system becomes a virtual machine instance with special privileges, called parent partition. The parent partition (also called root partition) is the only one that has direct access to the hardware, it runs the virtualized stack, host all the drivers required to configure guest operating systems and creates child partitions through the hypervisor. Child partitions are used to host guest operating systems and do not have access to the underlying hardware, but their interaction with it is controlled by either the parent partition or the hypervisor itself.

(a) Hypervisor: the hypervisor is the component that directly manages the underlying hardware (processors and memory). It is logically defined by the following components:

- Hypercalls interface
- Memory service routines (MSRs)
- Advanced programming interrupt controller (APIC)
- Schedular
- Address manager
- Partition manager

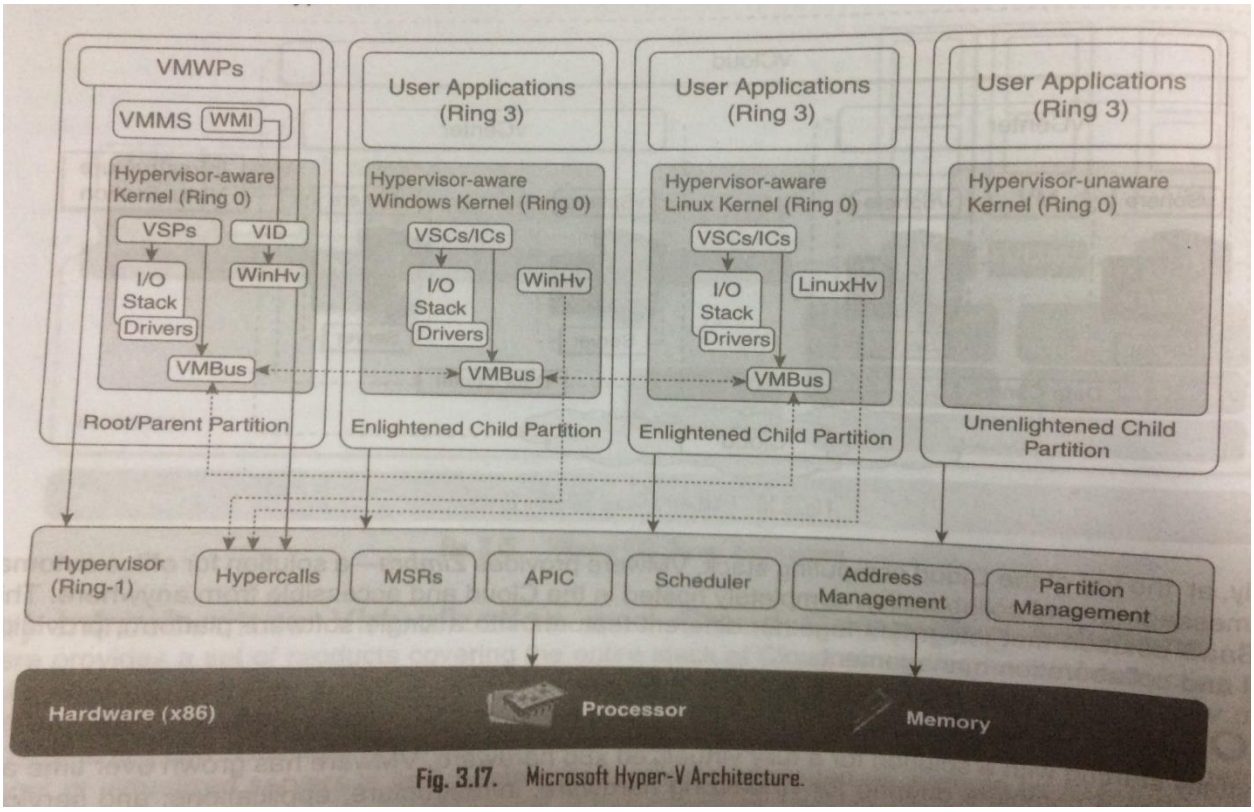

Fig. 3.17. Microsoft Hyper-V Architecture.

- **Hypercalls interface**
  This is the entry point for all the partitions for the execution of sensible instructions. This interface is used by drivers in the partitioned operating system to contact the hypervisor by using the standard windows calling convention. The parent partition also uses interface to create children partitions.

- **Memory service routines (MSRs)**
  These are the set of functionalities that control the memory, and its access from partitions. By leveraging hardware-assisted virtualization, the hypervisor uses the input output memory management unit (I/O MMU or IOMMU) to fast track the access to devices from partitions, by translating virtual memory addresses.

- **Advanced programming interrupt controller (APIC)**
  It represents the synthetic interrupt controller, which manages the virtual processor signals coming from the underlying hardware when some event occurs. The hypervisor is responsible of dispatching, when appropriate, the physical interrupts to the synthetic interrupt controller.

- **Schedular**
  It schedules the virtual processors to run on available physical processors. The scheduling is controlled by policies that are set by the parent partition.

- **Address manager**
  It is used to manage the virtual network addresses that are allocated to each guest operating system.

- **Partition manager**
  It creates the partition, finalization, destruction, enumeration, and configurations. Its services are available through the hypercalls interface API.

(b) Enlightened I/O and Synthetic Devices:

It provides inter-partition communication channel rather than traversing the hardware emulation stack provided by the hypervisor.

VMBus - an inter-partition communication channel that is used to exchange data between partitions for guest operating system.

VSPs (Virtual Service Providers)– these are kernel level drivers that are deployed in the parent partition and provides access to the corresponding hardware devices.

VSCs – these VSPs interact with VSCs which represent the virtual device drivers (synthetic drivers) seen by the guest operating systems in the children partitions.

(c) Parent Partition:

The parent partition is also the one that manages the creation, execution and destruction of children partitions.

VID (virtualization Infrastructure Driver) – which controls the access to the hypervisor and also allows the management of virtual processor and memory.

VMWPs (Virtual Machine Worker Process) – which manages the children partition by interacting with the hypervisor through the VID.

(d) Children Partition:

Children partitions are used to execute guest operating systems. These are isolated environments, which allow a secure and controlled execution of guests.

2. Cloud Computing and Infrastructure Management

Hyper-V constitutes the basic building block of Microsoft virtualization infrastructure. It contributes to create a ful-featured platform for server virtualization.

The basic features offered by Hyper-V with management capabilities includes:

- Management portal for the creation and management of virtual instances
- Virtual to virtual (V2V) and physical to virtual (P2V) conversions
- Delegated administration
- Library functionality and deep PowerShell integration
- Intelligent placement of virtual machines in the managed environment and
- Host capacity management.