



Virtualization

DR. AVTAR SINGH

Virtualization




Virtualization technologies have increased a change of interest in recent times because of coming together with various phenomena

- ▶ Increased performance and computing capacity
- ▶ Underutilized hardware and software resources
- ▶ Lack of space
- ▶ Greening initiatives.
- ▶ Rise of administrative costs.


Characteristics of Virtualized Environment

- ▶ Increased security
- ▶ Managed Execution
 - Sharing
 - Aggregation
 - Emulation
 - Isolation
- ▶ Portability

- ▶ **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. Sharing is a most significant feature in virtualized data centers, where this simple feature is used to reduce the number of active servers and limit power consumption.
- ▶ **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 connected and represented to guests as a single virtual host. This function is naturally implemented in middleware for distributed computing, example shown in cluster management software, which harnesses the physical resources of a homogeneous group of machines and represents them as a single resource.

- 
- **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 example, 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. This feature becomes very useful for testing purposes, where a specific guest has to be validated against different platforms or architectures and the wide range of options is not easily accessible during development.

Again, hardware virtualization solutions are able to provide virtual hardware and emulate a particular kind of device such as Small Computer System Interface (SCSI) devices for file I/O, without the hosting machine having such hardware installed. Old and legacies that does not meet the requirements of current systems can be run on emulated hardware without any need to change the code. This is possible either by emulating the required hardware architecture or within a specific operating systems and box, such as the MS-DOS mode in Windows 95/98. Another example of emulation is an arcade-game emulator that allows us to play arcade games on a normal personal computer.

- 
- ▶ **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 programmer forms its activity by interacting with an abstraction layer, which provides access to the underlying resources. Isolation brings several benefits; for example, it allows vmultiple guest storunon the same host without interfering with each other. Second, it provides a separation between the host and the guest. The virtual machine can filter the activity of the guest and prevent harmful operations against the host.

Virtualization Reference Model

Virtualization is a wide concept that refers to the creation of a virtual version of something, whether it is hardware or software environment, storage, or a network.

In a virtualized environment there are three major components: **guest, host, and virtualization layer**.

guest signifies the System component that interacts with the virtualization layer rather than with the host, as would normally happen.

host represents the original environment where the guest is supposed to be managed.

virtualization layer is responsible for recreating the same or a different environment where the guest will operate shown figure next slide.

The main common characteristic of all these different implementations is the fact that the virtual environment is created by means of a software program.

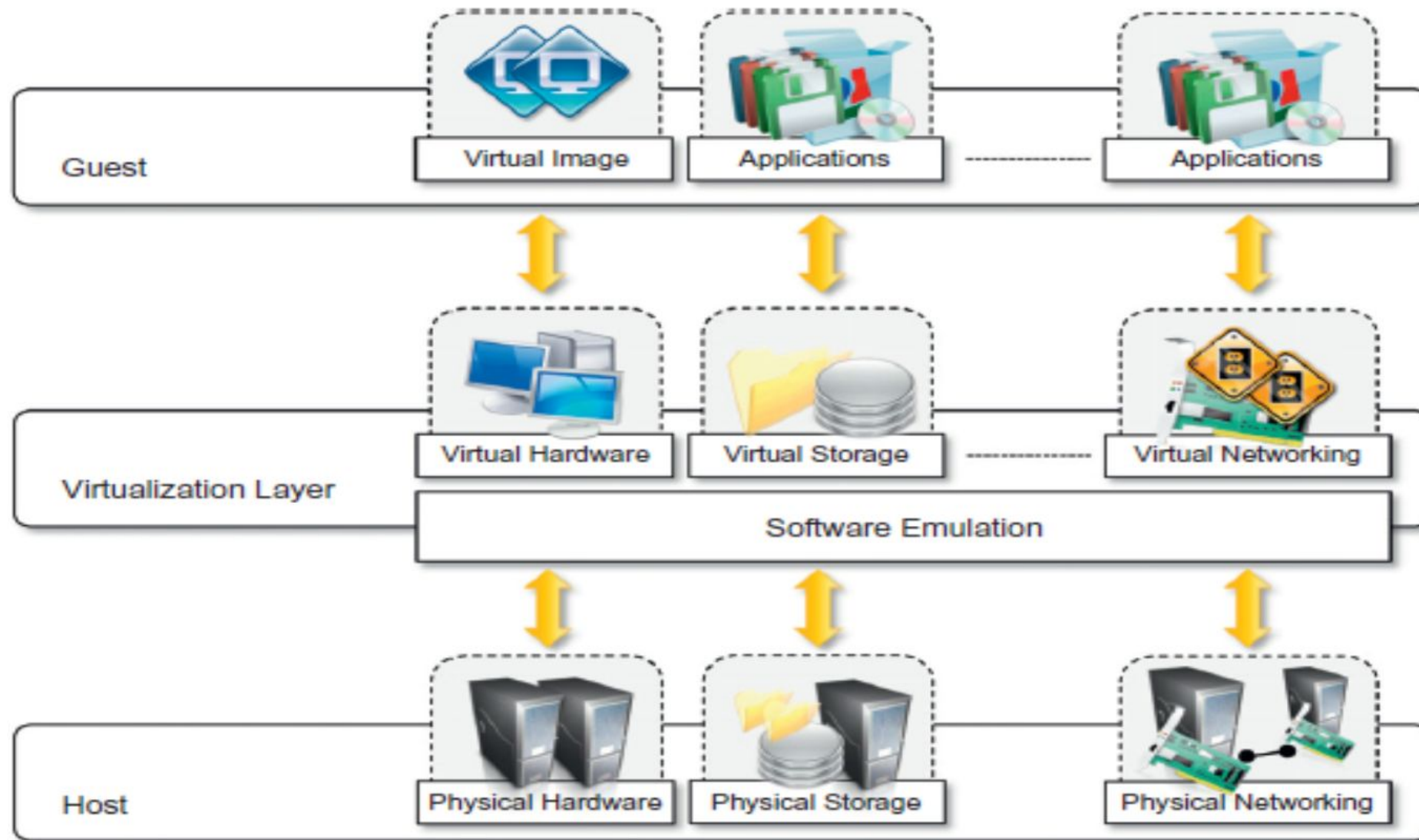


Fig. The virtualization reference model

Taxonomy of Virtualization Techniques

Virtualization covers a wide range of emulation techniques that are applied to different areas of computing. A organization of these techniques helps us better understand their characteristics and use (see Figure next slide). The first taxonomy distinguishes 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. Hence, it deserves major investigation and a further categorization.

It can divide these execution virtualization techniques into two major categories by considering the type of host they require.

Process-level techniques are implemented on top of an existing operating system, which has full control of the hardware.

System-level techniques are implemented directly on Hardware and do not require—or require a minimum of support from—an existing operating system.

These two groups can list various techniques that offer the guest a different type of virtual computation environment: bare hardware, operating system resources, low-level Programming language, and application libraries.

Taxonomy of virtualization techniques

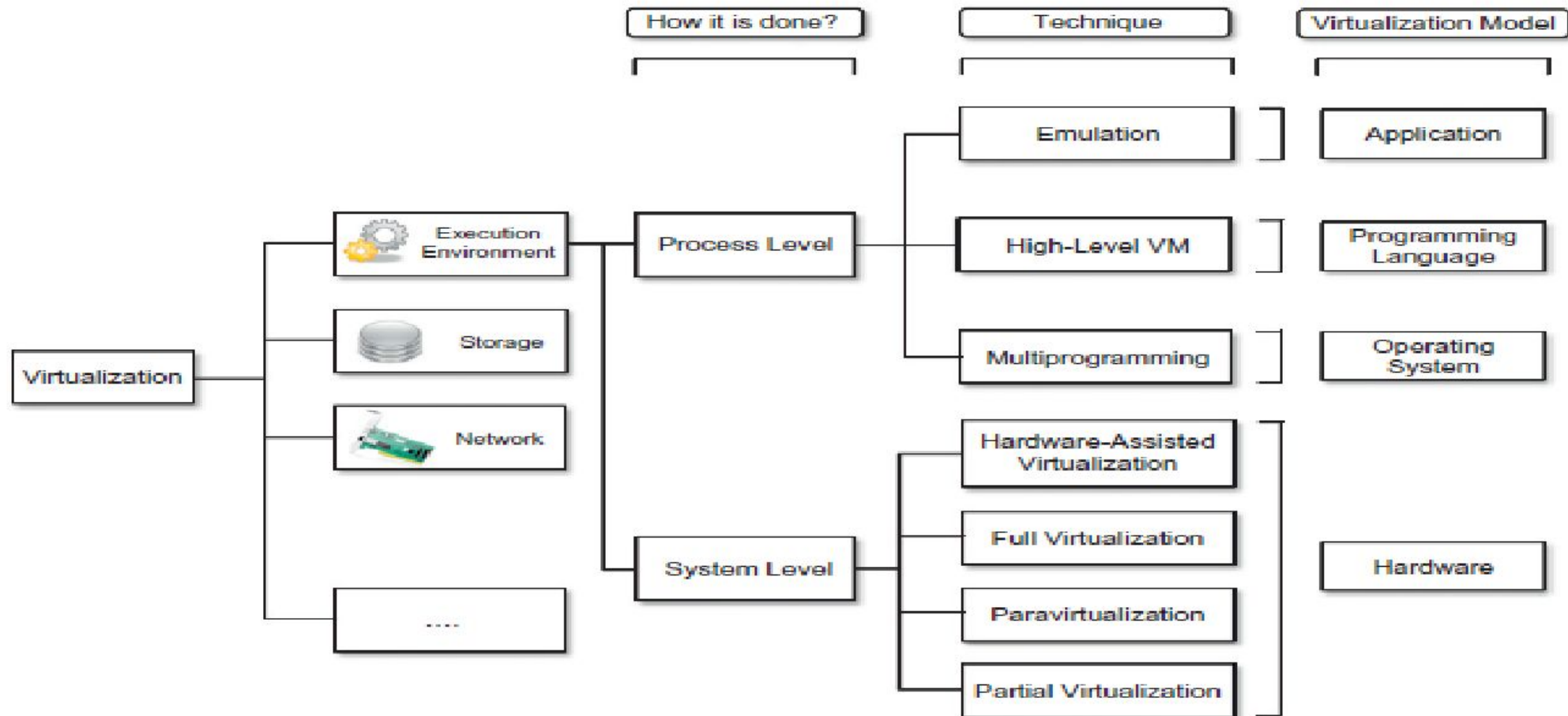


FIGURE 3.3

A taxonomy of virtualization techniques.

Hardware-assisted virtualization

- ▶ 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 extensions to the x86-64bit architecture introduced with IntelVT (formerly known as Vander pool) and AMDV (formerly known as Pacifica). These extensions, which differ between the two vendors, are meant to reduce the performance penalties experienced by emulating x86 hardware with hypervisors.
- ▶ Before the introduction of hardware-assisted virtualization, software emulation of x86 hardware was significantly costly from the performance point of view.
- ▶ The reason for this is that by design the x86 architecture did not meet the formal requirements Introduced by Popek and Goldberg, and early products were using binary translation to trap some Sensitive instructions and provide an emulated version. VMware Virtual Platform was Introduced in 1999 by VMware, which pioneered the field of x86 virtualization, were based on this technique. After 2006, Intel and AMD introduced processor extensions, and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), Virtual Box, Xen, VMware, Hyper-V, SunxVM, Parallels, and others.

Hardware level virtualization Model

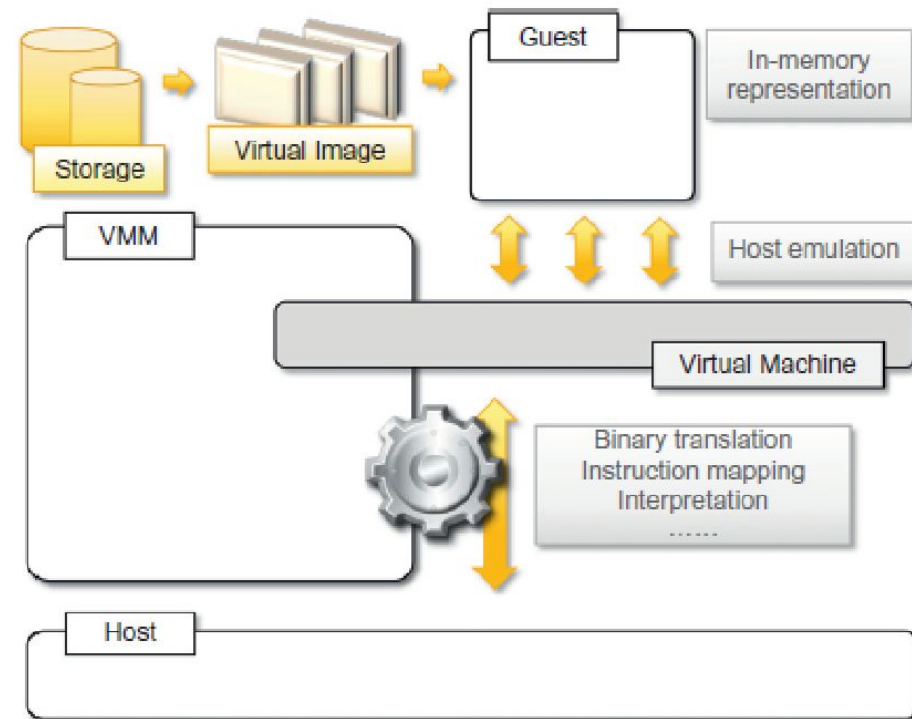


FIGURE 3.6

A hardware virtualization reference model.

Full virtualization

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 their hardware. To make this possible, (VMM) virtual machine managers 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. Whereas it is a desired goal for many virtualization solutions, full virtualization poses important concerns related to Performance and technical implementation.

A key challenge is the interception of privileged instructions namely I/O instructions: Since they change the state of the resources exposed by the host, they have to be contained within the virtual machine manager. A simple solution to achieve full virtualization is to provide a virtual environment for all the instructions, thus posing some limits on performance.

A successful and efficient implementation of full virtualization is obtained with a combination of hardware and software, not allowing potentially harmful instructions to be executed directly on the host. This is what is accomplished through hardware-assisted virtualization.

Full virtualization model

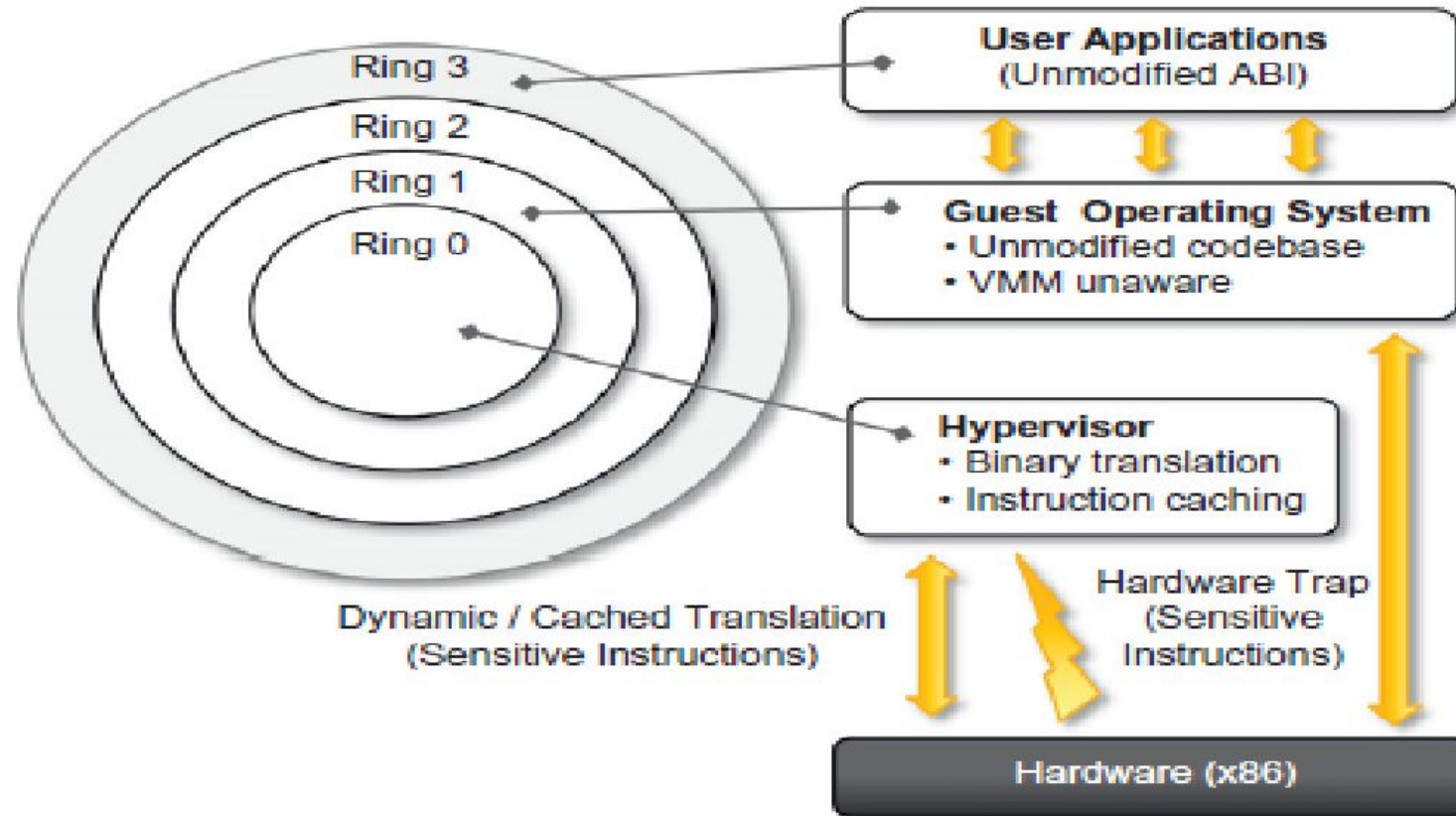


FIGURE 3.12

A full virtualization reference model.

Paravirtualization

This is a not-transparent virtualization solution that allows implementing thin virtual machine managers. Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, 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, thus preventing performance losses that would otherwise be experienced in managed execution. This allows a simpler implementation of virtual machine managers that have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host. To take advantage of such an opportunity, guest operating systems. This technique has been successfully used by Xen for providing virtualization solutions for Linux-based operating systems specifically ported to run on Xen hypervisors.

Operating systems that cannot be ported can still take advantage of paravirtualization by using adhoc device drivers that remap the execution of critical instructions to the paravirtualization APIs exposed by the hypervisor .

Xen provides this solution for running Windows-based OS on x86 architectures.

Other solutions using paravirtualization include VMWare, Parallels, and some solutions for embedded and real-time environments such as TRANGO, WindRiver, and XtratuM.

Operating system-level virtualization

- ▶ On the other hand, operating system-level virtualization does not expose the same Flexibility of hardware virtualization, since all the user space instances must share the same operating system.
- ▶ This technique is an efficient solution for server consolidation scenarios in which multiple Application servers share the same technology: operating system, application server framework, and other components. When different servers are aggregated into one physical server, each server is run in a different user space, completely isolated from the others.
- ▶ Examples of operating system-level virtualizations are FreeBSD Jails, IBM Logical Partition (LPAR), Solaris Zones and Containers, Parallels Virtuozzo Containers, OpenVZ, iCoreVirtual Accounts, Free Virtual Private Server (FreeVPS), and others.
- ▶ The services offered by these technologies differ, and most of them are available on Unix-based systems. Some of them, such as Solaris and OpenVZ, allow for different versions of the same operating system to operate concurrently.

Operating system-level virtualization

- ▶ Operating system-level virtualization offers the opportunity to create different and separated execution environments for applications that are managed concurrently. Differently from hardware virtualization, there is no virtual machine manager or hypervisor, and the virtualization is done within a Single OS, 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. A user space instance in general contains a proper view of the file system, which is completely isolated, and separate IP addresses, software configurations, and access to devices.
- ▶ OS supporting this type of virtualization are general-purpose, time shared operating systems with the capability to provide stronger namespace and resource isolation.
- ▶ This virtualization technique can be considered an evolution of the chroot mechanism in Unix systems. The chroot operation changes the file system root directory for a process and its children to a specific directory. As a result, the process and its children cannot have access to other portions of the file system than those accessible under the new root directory.

Programming language-level virtualization

- ▶ Programming language-level virtualization has lag behind 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.
- ▶ It 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 bytecode 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.
- ▶ The characteristics of this architecture vary from implementation to implementation. Normally these virtual machines constitute a simplification of the underlying hardware instruction set and provide some high-level Instructions that map some of the features of the languages compiled for them. At runtime, the byte code can be either interpreted or compiled on the fly—or JITted—against the underlying hardware instruction set.

Programming language-level virtualization

- ▶ Other examples to use this technology have been the UCSD Pascal and Smalltalk. Virtual machine programming languages become popular with Sun's introduction of the Java Platform in 1996. Originally created as a platform for developing Internet applications, Java became one of the technologies of choice for enterprise applications, and a large community of developers formed around it. **The Java virtual machine was originally designed for the execution of programs written in the Java language, but other languages such as Python, Pascal, Groovy, and Ruby were made available.** The ability to support multiple programming languages has been one of the key elements of the Common Language Infrastructure (CLI), which is the specification behind .NET Framework. Currently, the Java platform and .NET Framework are the most popular Technologies for enterprise application development.
- ▶ **Both Java and the CLI are stack-based virtual machines:** The reference model of the abstract architecture is based on an execution stack that is used to perform operations. The bytecode generated by compilers for these architectures contains a set of instructions that load operands on the stack, perform some operations, and put the result on the stack. Additionally, specific instructions for invoking methods and managing objects and classes are included.
- ▶ Stack-based virtual machines possess the property of being easily interpreted and executed simply by lexical analysis and hence are easily portable over different architectures. An alternative solution is offered by **register-based virtual machines**, in which the reference model is based on registers.

Programming language-level virtualization

- ▶ This kind of virtual machine is closer to the underlying architecture we use today. An example of a register-based virtual machine is Parrot, a programming-level virtual machine that was originally designed to support the execution of PERL and then generalized to host the execution of dynamic languages.
- ▶ The main advantage of these machines, also called process virtual machines, has the ability to provide a uniform execution environment across various platforms.
- ▶ Programs compiled into bytecode can be executed on any OS and platform for which a virtual machine able to execute that code has been provided.
- ▶ This simplifies the development from life-cycle point of view and deployment efforts since it is not necessary to provide different versions of the same code. The implementation of the virtual machine for different platforms is still a costly task, but it is done once and not for any application.

Programming language-level virtualization

- ▶ Moreover, process virtual machines allow for more control over the execution of programs since they do not provide direct access to the memory.
- ▶ Security is another advantage of managed programming languages; by filtering the I/O operations, the process virtual machine can easily support and boxing of applications.
- ▶ As an example, **both Java and .NET provide** an infrastructure for pluggable security policies and code access security frameworks. All these advantages come with a price: performance. Virtual machine programming languages generally expose an inferior performance compared to languages compiled against the real architecture. This performance difference is getting smaller, and the high compute power available on average processors makes it even less important.
- ▶ Implementations of this model are also called high-level virtual machines, since high-level programming languages are compiled to a conceptual ISA, which is further interpreted or dynamically translated against the specific instruction of the hosting platform.

Application-level virtualization

- ▶ This technique allow 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. In general, these techniques are mostly concerned with partial file systems, libraries, and operating system component emulation. Such emulation is performed by a thin layer—a program or an operating system component—that is in charge of executing the application.
- ▶ 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:
 - ▶ 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 Startup cost but a huge overhead, since each instruction is emulated.
 - ▶ 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.

Application-level virtualization

- ▶ Emulation, as described, is different from hardware-level virtualization. The former simply allows the execution of a program compiled against a different hardware, whereas the latter emulates a complete hardware environment where an entire operating system can be installed.
- ▶ Application virtualization is a good solution in the case of missing libraries in the host operating system; in this case a replacement library can be linked with the application, or library calls can be remapped to existing functions available in the host system.
- ▶ The advantage is that in this case the virtual machine manager is much lighter since it provides a partial emulation of the runtime environment compared to hardware virtualization. Moreover, this technique allows incompatible applications to run together. Compared to programming-level virtualization, which works across all the applications developed for that virtual machine, application-level virtualization works for a specific environment: It supports all the applications that run on top of a specific environment.

Application-level virtualization

- ▶ The most popular solution simple meeting application virtualization is Wine, which is a Software application allowing Unix-like operating systems to execute programs written for the Microsoft Windows platform.
- ▶ Wine features a software application acting as a container for the guest application and a set of libraries, called Winelib that developers can use to compile applications to be ported on Unixsystems.
- ▶ Wine takes its inspiration from a similar product from Sun, Windows Application Binary Interface (WABI), which implements the Win16API specifications on Solaris.
- ▶ A similar solution for the MacOSX environment is CrossOver, which allows running Windows applications directly on the MacOSX operating system. VMware ThinApp, another Product in this area, allows capturing the setup of an installed application and packaging it in to an Executable image isolated from the hosting operating system.

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.
- ▶ Execution virtualization can be implemented directly on top of the hardware by the operating system, an application, or libraries dynamically or statically linked to an application image.

Machine Reference Model



Virtualizing an execution environment at different levels of the computing stack requires a reference model that defines the interfaces between the levels of abstractions, which hide implementation details. From this perspective, virtualization techniques actually replace one of the layers and intercept the calls that are directed toward it. Therefore, a clear separation between layers simplifies their implementation, which only requires the emulation of the interfaces and a proper interaction with the underlying layer.

Machine Reference Model

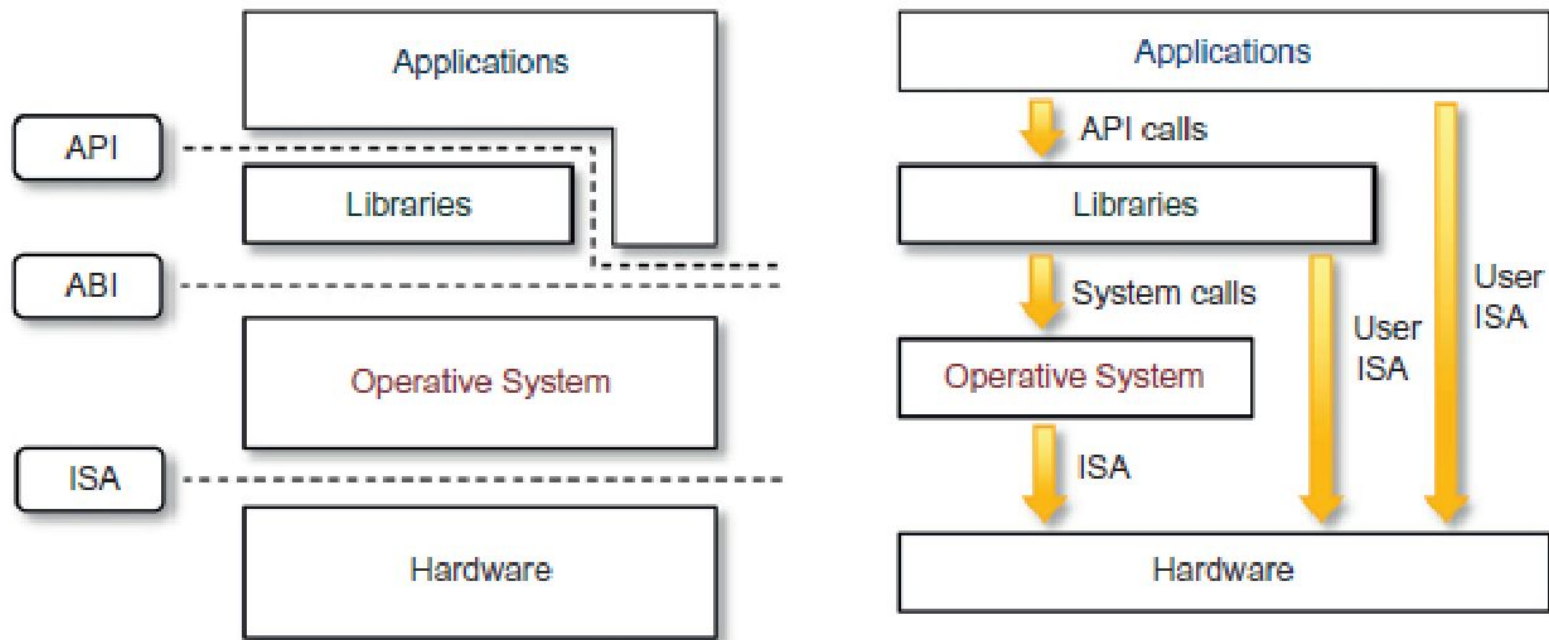


FIGURE 3.4

A machine reference model.

Machine reference model

Nonprivileged instructions are those instructions that can be used without interfering with other tasks because they do not access shared resources. This category contains, for example, all the floating, fixed-point, and arithmetic 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. For instance, *behavior sensitive* instructions are those that operate on the I/O, where as *control-sensitive* instructions alter the state of the CPU registers. Some types of architecture feature more than one class of privileged instructions and implement a finer control of how these instructions can be accessed. For instance, a possible implementation features a hierarchy of privileges (see Fig. next slide) in the form of ring-based security: Ring 0, Ring 1, Ring 2, and Ring 3; Ring0 is in the most privileged level and Ring 3 in the least privileged level. Ring 0 is used by the kernel of the OS, rings1 and 2 are used by the OS-level services, and Ring 3 is used by the user. Recent systems support only two levels, with Ring 0 for supervisor mode and Ring3 for user mode.

Machine reference model

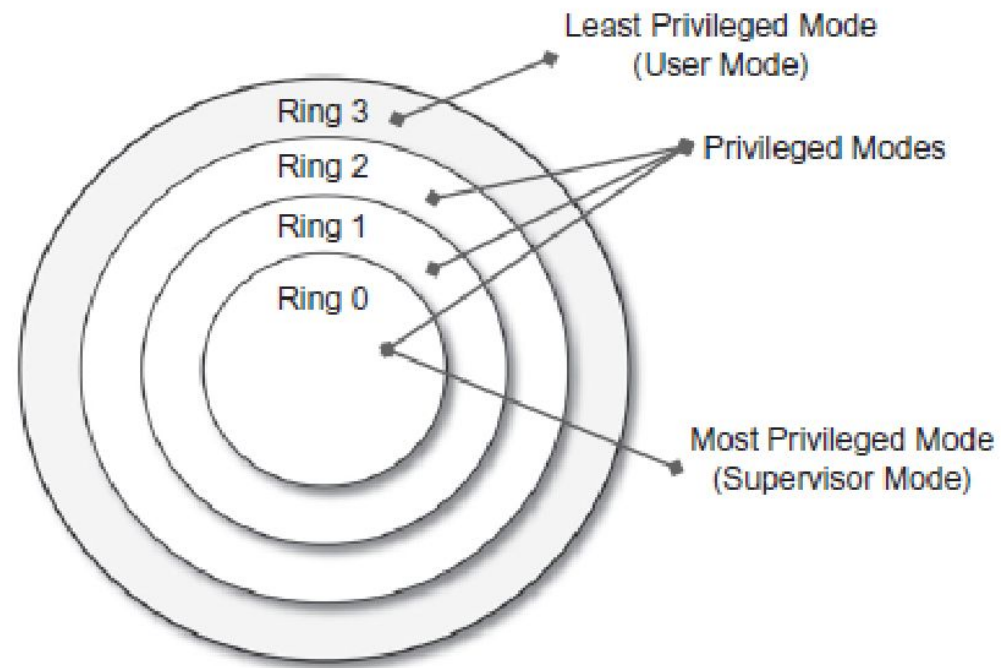


FIGURE 3.5

Security rings and privilege modes.

Virtual Machine with Host

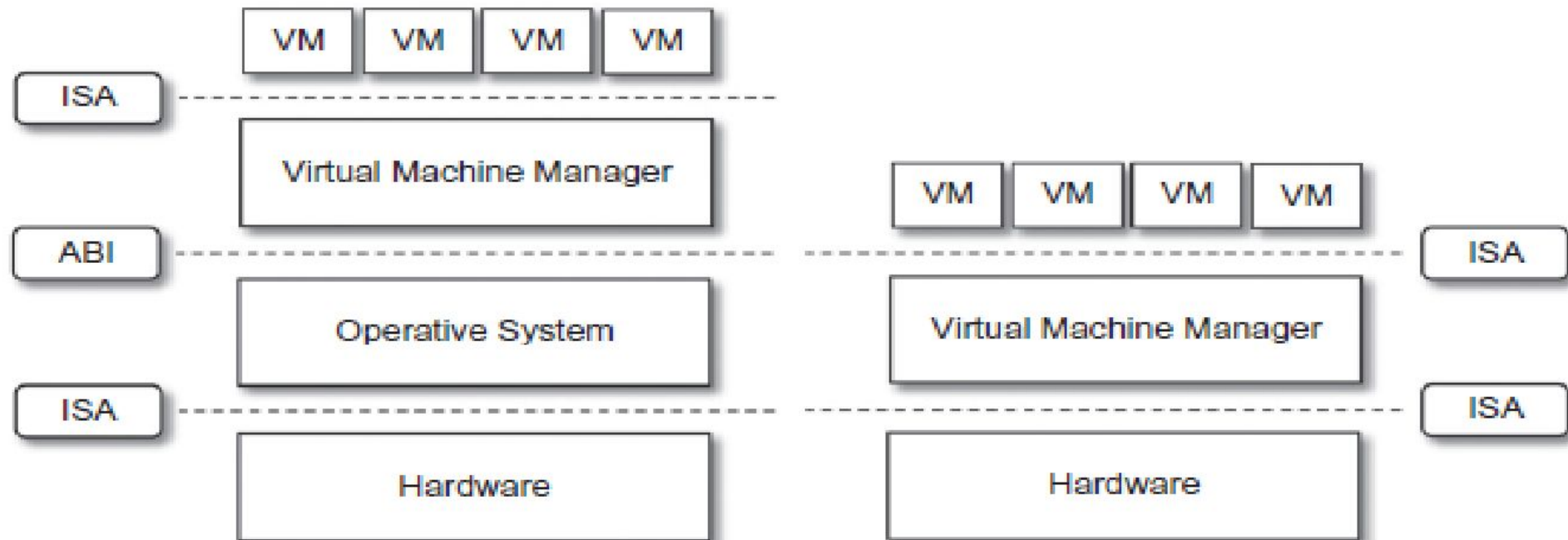
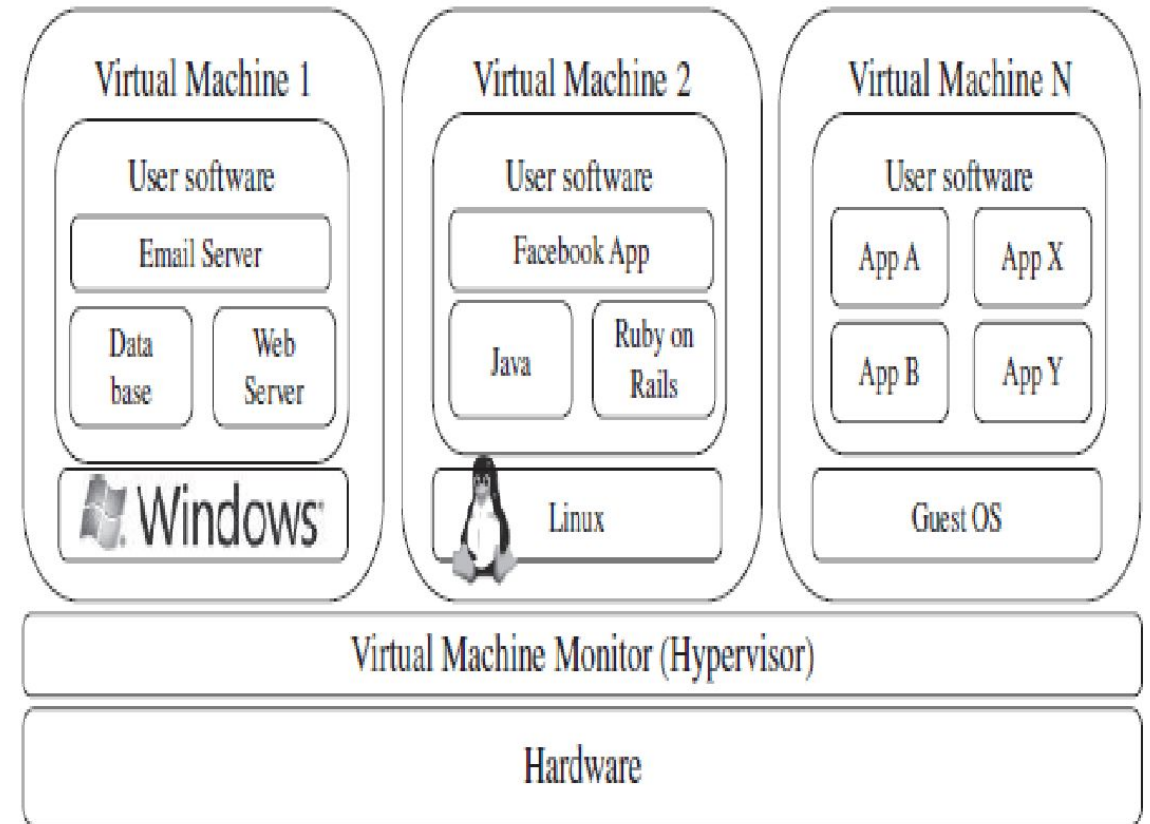


FIGURE 3.7

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

Virtual Machine Monitor

Hardware virtualization allows running multiple operating systems and software stacks on a single physical platform. As shown in Fig., a software layer, the virtual machine monitor (VMM), also called a hypervisor, mediates access to the physical hardware presenting to each guest operating system a virtual machine (VM), which is a set of virtual platform interfaces



Other Virtualization Techniques

A number of VMM platforms exist that are the basis of many utility or cloud computing environments. The most notable ones, VMWare, Xen, and KVM, are outlined in the following sections.

VMWare ESXi. VMware is a pioneer in the virtualization market. Its ecosystem of tools ranges from server and desktop virtualization to high-level management tools [24]. ESXi is a VMM from VMWare. It is a bare-metal hypervisor, meaning that it installs directly on the physical server, whereas others may require a host operating system. It provides advanced virtualization techniques of processor, memory, and I/O. Especially, through memory ballooning and page sharing, it can overcommit memory, thus increasing the density of VMs inside a single physical server.

Other Virtualization Techniques

Xen. The Xen hypervisor started as an open-source project and has served as a base to other virtualization products, both commercial and open-source. It has pioneered the para-virtualization concept, on which the guest operating system, by means of a specialized kernel, can interact with the hypervisor, thus significantly improving performance. In addition to an open-source distribution [25], Xen currently forms the base of commercial hypervisors of a number of vendors, most notably Citrix XenServer [26] and Oracle VM [27].

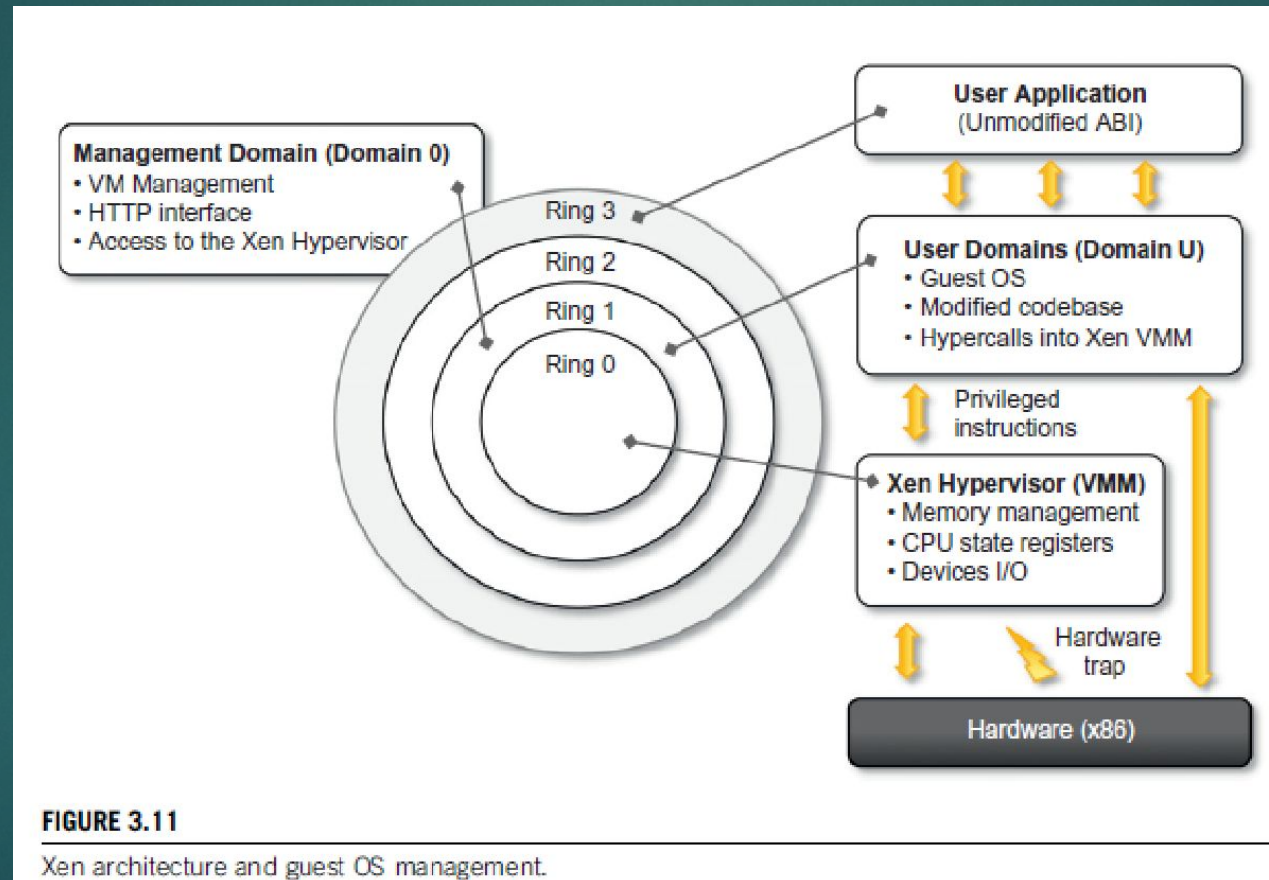
KVM. The kernel-based virtual machine (KVM) is a Linux virtualization subsystem. It has been part of the mainline Linux kernel since version 2.6.20, thus being natively supported by several distributions. In addition, activities such as memory management and scheduling are carried out by existing kernel features, thus making KVM simpler and smaller than hypervisors that take control of the entire machine [28].

KVM leverages hardware-assisted virtualization, which improves performance and allows it to support unmodified guest operating systems [29]; currently, it supports several versions of Windows, Linux, and UNIX [28].

Technology Examples

- ▶ Xen is the most popular implementation of paravirtualization, which in contrast with fullvirtualization, allows high performance execution of guest operating systems. This is made possible by eliminating the performance loss while executing instructions that require special management.
- ▶ Figure in next Slide describes the architecture of Xen and its mapping on to a classic x86 privilege model. A Xen-based system is managed by the Xen hypervisor, which runs in the highest privileged mode and controls the access of guest operating system to the underlying hardware.

Xen Architecture and guest OS management



Virtualization solutions

- ▶ End user (desktop) virtualization

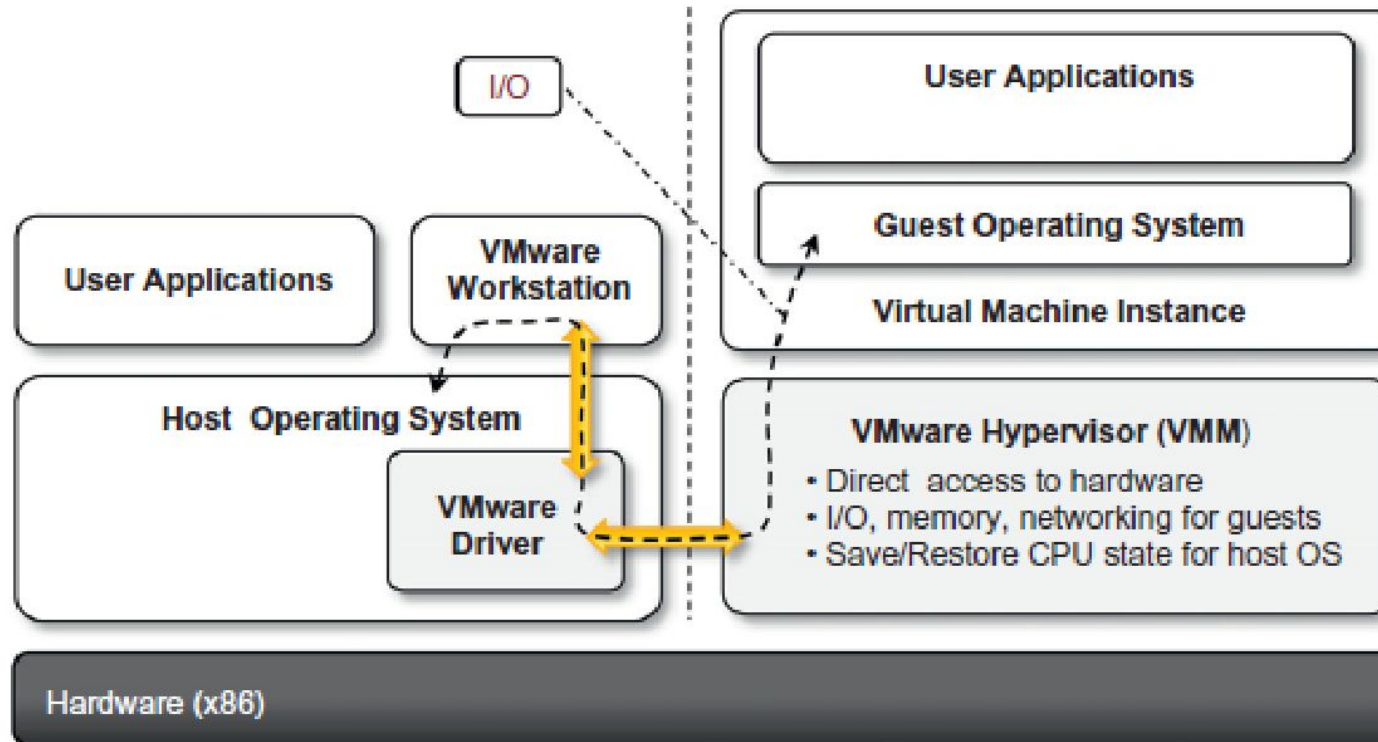


FIGURE 3.13

VMware workstation architecture.

Virtualization Solutions

This is done by installing a specific driver in the host 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.

Using this architecture — also called Hosted Virtual Machine Architecture—it is possible to both isolate virtual machine instances within the memory space of a single application and provide reasonable performance, since the intervention of the VMware application is required only for instructions, such as device I/O, that require binary translation. Instructions that can be directly executed are managed by the virtual machine manager, which takes control of the CPU and the MMU and alternates its activity with the hostOS. Virtual machine images are saved in a collection of files on the host filesystem, and both VMware Workstation and VMware Fusion allow creation of new images, pause their execution, create snapshots, and undo operations by rolling back to a previous state of the virtual machine.

Cont..

- ▶ VMware Player is a reduced version of VMware Workstation that allows creating and playing virtual machines in a Windows or Linux operating environment.
- ▶ VMWare ESXi Server, are examples of the hypervisor based approach. Both can be installed on bare metal servers and provide services for virtual machine management.

VMware Server Architecture

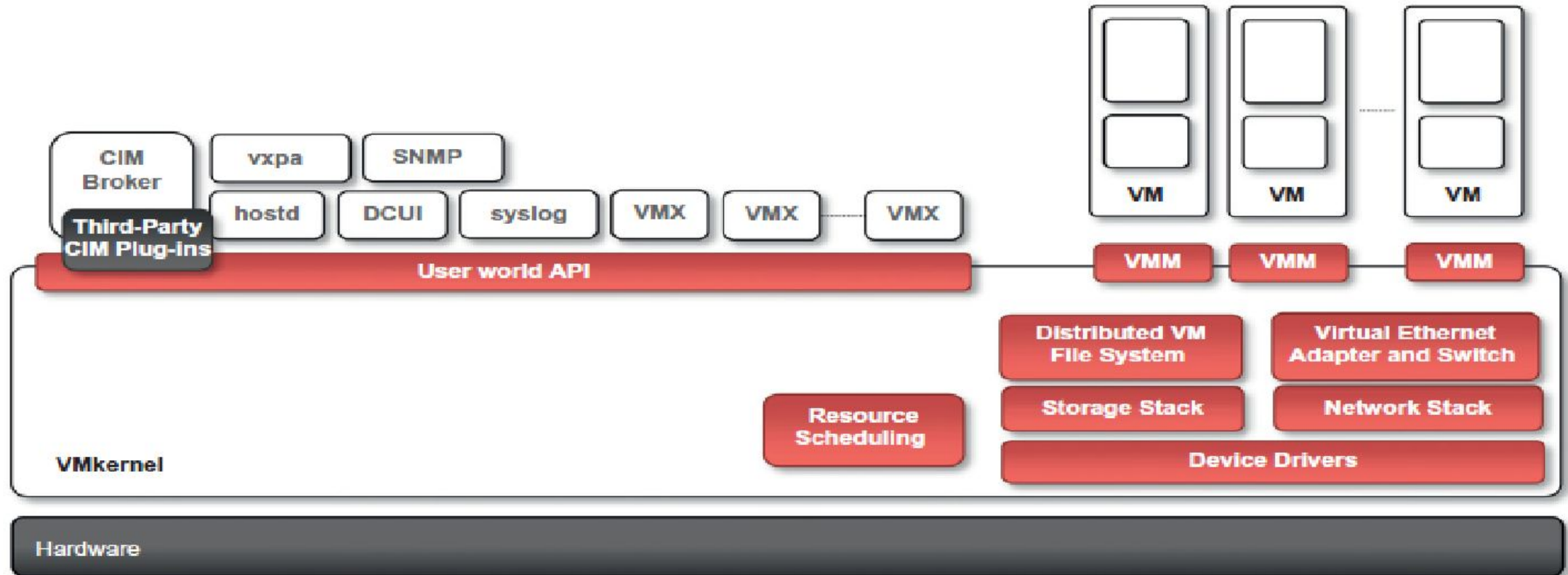


FIGURE 3.15

VMware ESXi server architecture.

VMware Cloud Solution Stack

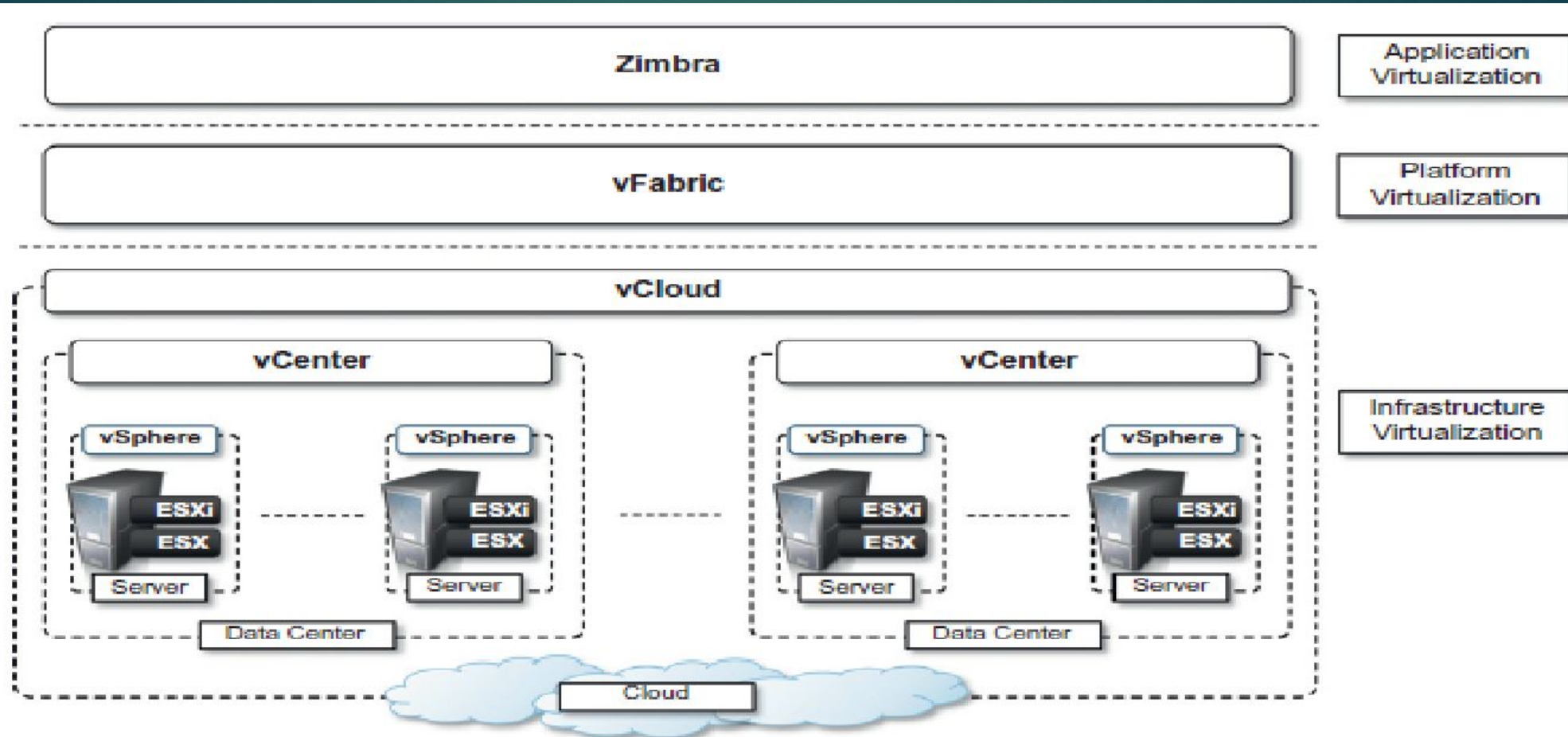


FIGURE 3.16

VMware Cloud Solution stack.

Hypervisor

- ▶ Hypercalls interface
- ▶ Memory service routines (MSRs)
 - ▶ I/O Memory Management Unit
- ▶ Advanced programmable interrupt controller (APIC)
 - ▶ Event occurs (timer expired, I/O ready, exception and trap)
- ▶ Scheduler
- ▶ Address Manager
- ▶ Partition Manager

Hypervisor reference Architecture

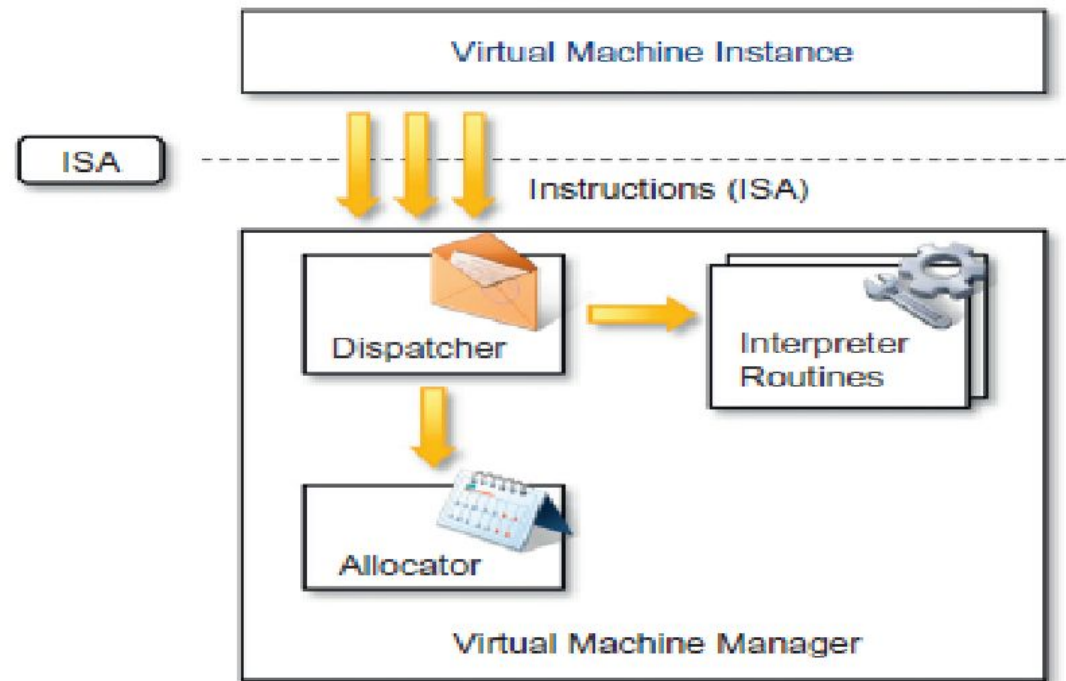


FIGURE 3.8

A hypervisor reference architecture.

Microsoft Hyper-V

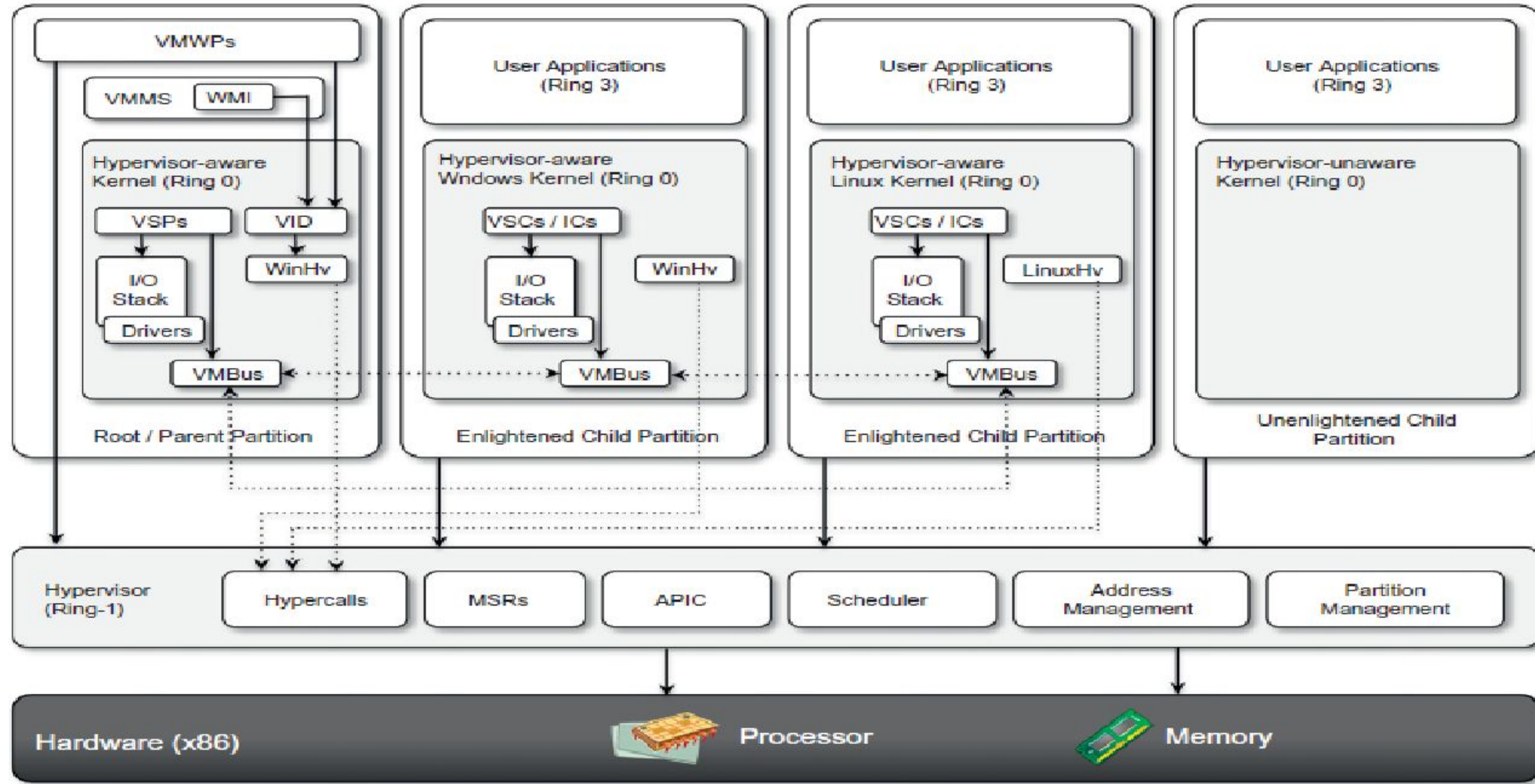


FIGURE 3.17

Microsoft Hyper-V architecture.