

3.6.1 Xen: paravirtualization

Xen is an open-source initiative implementing a virtualization platform based on paravirtualization. Initially developed by a group of researchers at the University of Cambridge in the United Kingdom, Xen now has a large open-source community backing it. Citrix also offers it as a commercial solution, XenSource. Xen-based technology is used for either desktop virtualization or server virtualization, and recently it has also been used to provide cloud computing solutions by means of Xen Cloud Platform (XCP). At the basis of all these solutions is the Xen Hypervisor, which constitutes the core technology of Xen. Recently Xen has been advanced to support full virtualization using hardware-assisted virtualization.

Xen is the most popular implementation of *paravirtualization*, which, in contrast with full virtualization, allows high-performance execution of guest operating systems. This is made possible by eliminating the performance loss while executing instructions that require special management. This is done by modifying portions of the guest operating systems run by Xen with reference to the execution of such instructions. Therefore it is not a transparent solution for implementing virtualization. This is particularly true for x86, which is the most popular architecture on commodity machines and servers.

Figure 3.11 describes the architecture of Xen and its mapping onto 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. Guest

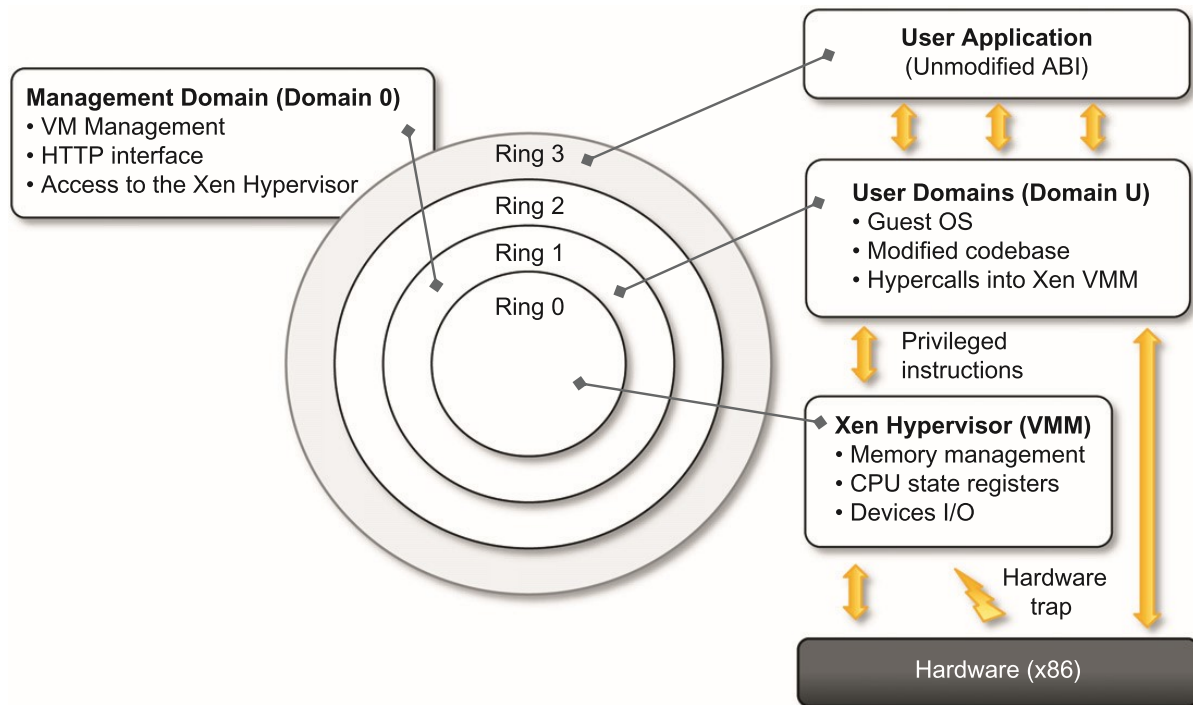


FIGURE 3.11

Xen architecture and guest OS management.

operating systems are executed within *domains*, which represent virtual machine instances. Moreover, specific control software, which has privileged access to the host and controls all the other guest operating systems, is executed in a special domain called *Domain 0*. This is the first one that is loaded once the virtual machine manager has completely booted, and it hosts a HyperText Transfer Protocol (HTTP) server that serves requests for virtual machine creation, configuration, and termination. This component constitutes the embryonic version of a distributed virtual machine manager, which is an essential component of cloud computing systems providing Infrastructure-as-a-Service (IaaS) solutions.

Many of the x86 implementations support four different security levels, called *rings*, where Ring 0 represent the level with the highest privileges and Ring 3 the level with the lowest ones. Almost all the most popular operating systems, except OS/2, utilize only two levels: Ring 0 for the kernel code, and Ring 3 for user application and nonprivileged OS code. This provides the opportunity for Xen to implement virtualization by executing the hypervisor in Ring 0, Domain 0, and all the other domains running guest operating systems—generally referred to as *Domain U*—in Ring 1, while the user applications are run in Ring 3. This allows Xen to maintain the ABI unchanged, thus allowing an easy switch to Xen-virtualized solutions from an application point of view. Because of the structure of the x86 instruction set, some instructions allow code executing in Ring 3 to jump into Ring 0 (kernel mode). Such operation is performed at the hardware level and therefore within a virtualized environment will result in a *trap* or *silent fault*, thus preventing the normal operations of the guest operating system, since this is now running in Ring 1. This condition is generally triggered by a subset of the system calls. To avoid this situation, operating systems need to be changed in their implementation, and the sensitive system calls need to be reimplemented with *hypercalls*, which are specific calls exposed by the virtual machine interface of Xen. With the use of hypercalls, the Xen hypervisor is able to catch the execution of all the sensitive instructions, manage them, and return the control to the guest operating system by means of a supplied handler.

Paravirtualization needs the operating system codebase to be modified, and hence not all operating systems can be used as guests in a Xen-based environment. More precisely, this condition holds in a scenario where it is not possible to leverage hardware-assisted virtualization, which allows running the hypervisor in Ring -1 and the guest operating system in Ring 0. Therefore, Xen exhibits some limitations in the case of legacy hardware and legacy operating systems. In fact, these cannot be modified to be run in Ring 1 safely since their codebase is not accessible and, at the same time, the underlying hardware does not provide any support to run the hypervisor in a more privileged mode than Ring 0. Open-source operating systems such as Linux can be easily modified, since their code is publicly available and Xen provides full support for their virtualization, whereas components of the Windows family are generally not supported by Xen unless hardware-assisted virtualization is available. It can be observed that the problem is now becoming less and less crucial since both new releases of operating systems are designed to be virtualization aware and the new hardware supports x86 virtualization.

3.6.2 VMware: full virtualization

VMware's technology is based on the concept of *full virtualization*, where the underlying hardware is replicated and made available to the guest operating system, which runs unaware of such abstraction layers and does not need to be modified. VMware implements full virtualization either in the

desktop environment, by means of *Type II* hypervisors, or in the server environment, by means of *Type I* hypervisors. In both cases, full virtualization is made possible by means of *direct execution* (for nonsensitive instructions) and *binary translation* (for sensitive instructions), thus allowing the virtualization of architecture such as x86.

Besides these two core solutions, VMware provides additional tools and software that simplify the use of virtualization technology either in a desktop environment, with tools enhancing the integration of virtual guests with the host, or in a server environment, with solutions for building and managing virtual computing infrastructures.

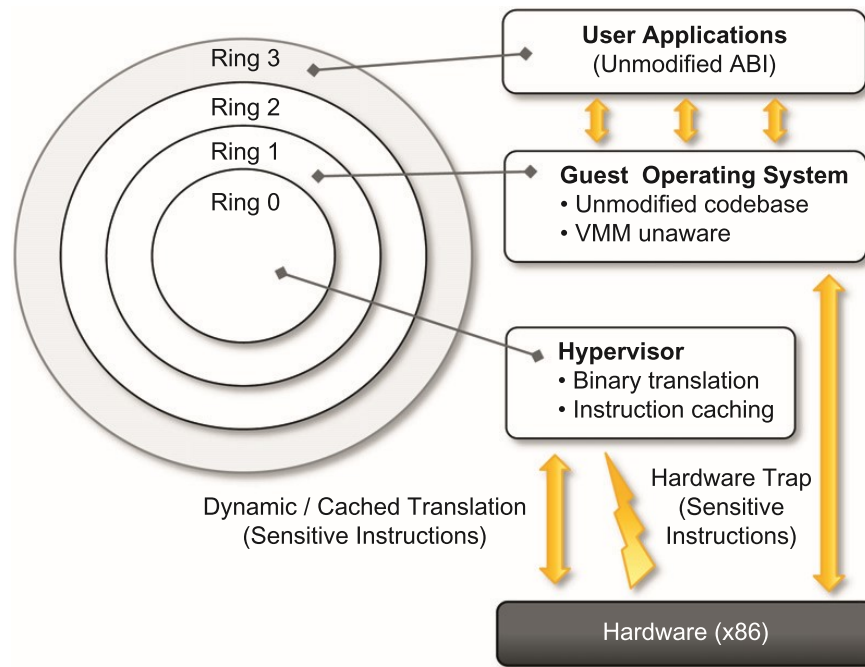
3.6.2.1 Full virtualization and binary translation

VMware is well known for the capability to virtualize x86 architectures, which runs unmodified on top of their hypervisors. With the new generation of hardware architectures and the introduction of *hardware-assisted virtualization* (Intel VT-x and AMD V) in 2006, full virtualization is made possible with hardware support, but before that date, the use of *dynamic binary translation* was the only solution that allowed running x86 guest operating systems unmodified in a virtualized environment.

As discussed before, x86 architecture design does not satisfy the first theorem of virtualization, since the set of sensitive instructions is not a subset of the privileged instructions. This causes a different behavior when such instructions are not executed in Ring 0, which is the normal case in a virtualization scenario where the guest OS is run in Ring 1. Generally, a trap is generated and the way it is managed differentiates the solutions in which virtualization is implemented for x86 hardware. In the case of dynamic binary translation, the trap triggers the translation of the offending instructions into an equivalent set of instructions that achieves the same goal without generating exceptions. Moreover, to improve performance, the equivalent set of instruction is cached so that translation is no longer necessary for further occurrences of the same instructions. [Figure 3.12](#) gives an idea of the process.

This approach has both advantages and disadvantages. The major advantage is that guests can run unmodified in a virtualized environment, which is a crucial feature for operating systems for which source code is not available. This is the case, for example, of operating systems in the Windows family. Binary translation is a more portable solution for full virtualization. On the other hand, translating instructions at runtime introduces an additional overhead that is not present in other approaches (paravirtualization or hardware-assisted virtualization). Even though such disadvantage exists, binary translation is applied to only a subset of the instruction set, whereas the others are managed through direct execution on the underlying hardware. This somehow reduces the impact on performance of binary translation.

CPU virtualization is only a component of a fully virtualized hardware environment. VMware achieves full virtualization by providing virtual representation of memory and I/O devices. Memory virtualization constitutes another challenge of virtualized environments and can deeply impact performance without the appropriate hardware support. The main reason is the presence of a *memory management unit (MMU)*, which needs to be emulated as part of the virtual hardware. Especially in the case of *hosted hypervisors* (Type II), where the virtual MMU and the host-OS MMU are traversed sequentially before getting to the physical memory page, the impact on performance can be significant. To avoid nested translation, the *translation look-aside buffer (TLB)* in the virtual MMU directly maps physical pages, and the performance slowdown only occurs in case of a TLB miss.

**FIGURE 3.12**

A full virtualization reference model.

Finally, VMware also provides full virtualization of I/O devices such as network controllers and other peripherals such as keyboard, mouse, disks, and universal serial bus (USB) controllers.

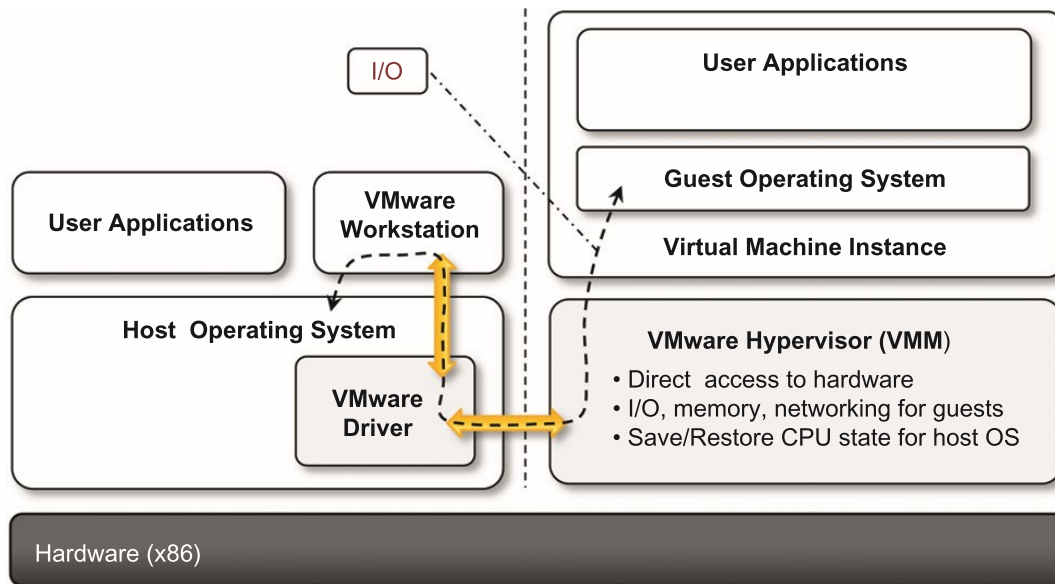
3.6.2.2 Virtualization solutions

VMware is a pioneer in virtualization technology and offers a collection of virtualization solutions covering the entire range of the market, from desktop computing to enterprise computing and infrastructure virtualization.

End-user (desktop) virtualization

VMware supports virtualization of operating system environments and single applications on end-user computers. The first option is the most popular and allows installing a different operating systems and applications in a completely isolated environment from the hosting operating system. 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 graphical user interface (GUI) of the host operating system). [Figure 3.13](#) provides an overview of the architecture of these systems.

The virtualization environment is created by an application installed in guest operating systems, which provides those operating systems with full hardware virtualization of the underlying

**FIGURE 3.13**

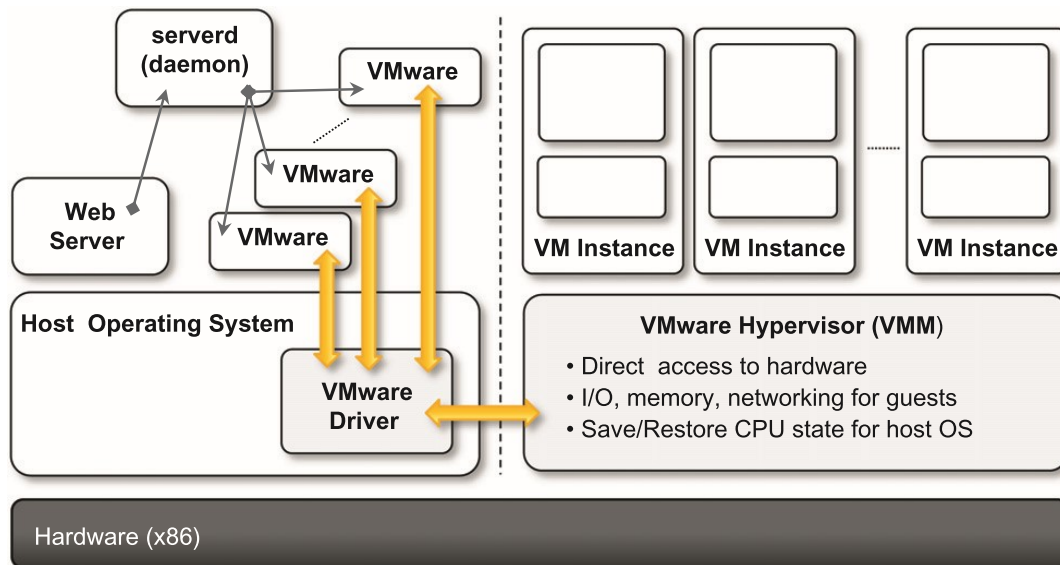
VMware workstation architecture.

hardware. 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 host OS. Virtual machine images are saved in a collection of files on the host file system, 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.

Other solutions related to the virtualization of end-user computing environments include VMware Player, VMware ACE, and VMware ThinApp. VMware Player is a reduced version of VMware Workstation that allows creating and playing virtual machines in a Windows or Linux operating environment. VMware ACE, a similar product to VMware Workstation, creates policy-wrapped virtual machines for deploying secure corporate virtual environments on end-user computers. VMware ThinApp is a solution for application virtualization. It provides an isolated environment for applications in order to avoid conflicts due to versioning and incompatible applications. It detects all the changes to the operating environment made by the installation of a specific application and stores them together with the application binary into a package that can be run with VMware ThinApp.

**FIGURE 3.14**

VMware GSX server architecture.

Server virtualization

VMware provided solutions for server virtualization with different approaches over time. 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 [Figure 3.14](#).

The architecture is mostly designed to serve the virtualization of Web servers. A daemon process, called *serverd*, controls and manages VMware application processes. These applications are then connected to the virtual machine 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 requests for virtual machine management and provisioning are routed from the Web server through the VMM by means of *serverd*.

VMware ESX Server and its enhanced version, 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. The two solutions provide the same services but differ in the internal architecture, more specifically in the organization of the hypervisor kernel. VMware ESX embeds a modified version of a Linux operating system, which provides access through a service console to hypervisor. VMware ESXi implements a very thin OS layer and replaces the service console with interfaces and services for remote management, thus considerably reducing the hypervisor code size and memory footprint.

The architecture of VMware ESXi is displayed in [Figure 3.15](#). The base of the infrastructure is the VMkernel, which is a thin Portable Operating System Interface (POSIX) compliant operating system that provides the minimal functionality for processes and thread management, file system, I/O stacks, and resource scheduling. The kernel is accessible through specific APIs called User world API. These