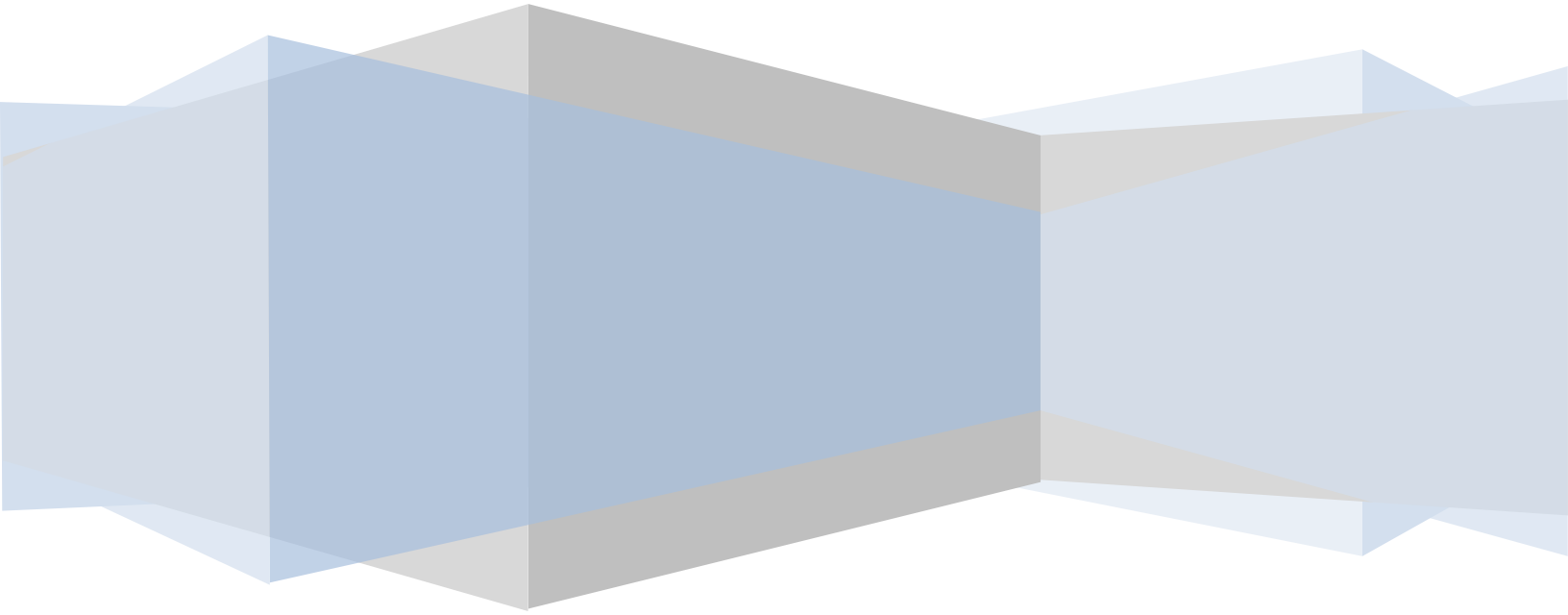
VMware Server 2.0 Essentials

Virtualization Deployment and Management



Contents

Chapter 1. What is VMware Server 2.0?

VMware Server 2.0 is the second release of a virtualization solution provided by VMware, Inc., a division of EMC Corporation. VMware Server is supplied free of charge and is the entry level product of a range of virtualization solutions provided by VMware.

1.1 What is Virtualization?

In a traditional computing model, a computer system typically runs a single operating system. For example, a desktop computer might run a copy of Windows XP or Windows Vista, while a server might run Linux or Windows Server 2008.

The concept of virtualization, as it pertains to this book, involves the use of a variety of different technologies to allow multiple and potentially varied operating system instances to run concurrently on a single physical computer system, each sharing the physical resources of the host computer system (such as memory, network connectivity, CPU and storage). Within a virtualized infrastructure, a single physical computer server might, for example, run two instances of Windows Server 2008 and one instance of Linux. This, in effect, allows a single computer to provide an IT infrastructure that would ordinarily required three computer systems.

1.2 Why is Virtualization Important?

Virtualization has gained a considerable amount of coverage in the trade media in recent years. Given this sudden surge of attention it would be easy to make the assumption that the concept of virtualization is new. In fact, virtualization has been around in one form or another since it was first introduced on IBM mainframe operating systems in the 1960s.

The reason for the sudden popularity of virtualization can be attributed to a number of largely unconnected trends:

Green computing - So called *green computing* refers to the recent trend to reduce the power consumption of computer systems. Whilst not a primary concern for individual users or small businesses, companies with significant server operations can save considerable power usage levels by reducing the number of physical servers required using virtualization. An additional advantage involves the reduction in power used for cooling purposes, since fewer servers generate less heat.

Increased computing power - The overall power of computer systems has increased exponentially in recent decades to the extent that many computers, by running a single operating system instance, are using a fraction of the available memory and CPU power.

Virtualization allows companies to maximize utilization of hardware by running multiple operating systems concurrently on single physical systems.

Financial constraints - Large enterprises are under increasing pressure to reduce overheads and maximize shareholder returns. A key technique for reducing IT overheads is to use virtualization to gain maximum return on investment of computer hardware.

Web 2.0 - The term *Web 2.0* has primarily come to represent the gradual shift away from hosting applications and data on local computer systems to a web based approach. For example, many users and companies now use Google Apps for spreadsheet and word processing instead of installing office suite software on local desktop computers. Web services such as these require the creation of vast *server farms* running hundreds or even thousands of servers, consuming vast amounts of power and generating significant amounts of heat. Virtualization allows web services providers to consolidate physical server hardware, thereby cutting costs and reducing power usage.

Operating system fragmentation - In recent years the operating system market has increasingly fragmented with Microsoft ceding territory to offerings such as Linux and Apple's Mac OS. Enterprises are now finding themselves managing heterogeneous environments where, for example, Linux is used for hosting web sites whilst Windows Server is used to email and file serving functions. In such environments, virtualization allows different operating systems to run side by side on the same computer systems. A similar trend is developing on the desktop, with many users considering Linux as an alternative Microsoft Windows. Desktop based virtualization allows users to run both Linux and Windows in parallel, a key requirement given that many users looking at Linux still need access to applications that are currently only available on Windows.

1.3 How Does Virtualization Work?

A number of different approaches to virtualization have been developed over the years, each with inherent advantages and disadvantages. VMware Server 2.0 uses a concept known as *software virtualization*. This, and the other virtualization methodologies, will be covered in detail in the remaining sections of this chapter.

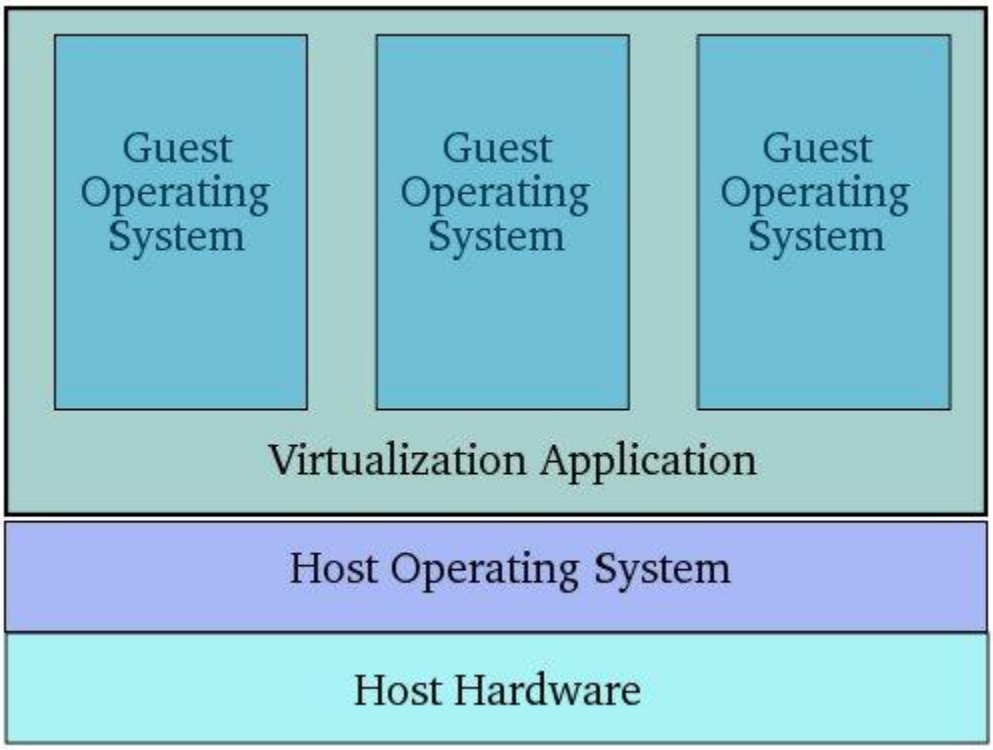
1.4 Software Virtualization

Software virtualization is perhaps the easiest concept to understand. In this scenario the physical host computer system runs a standard unmodified operating system such as Windows, Linux, UNIX or Mac OS X. Running on this operating system is a virtualization application which executes in much the same way as any other application such as a word processor or spreadsheet would run on the system. It is within this virtualization application that one or

more virtual machines are created to run the guest operating systems on the host computer. The virtualization application is responsible for starting, stopping and managing each virtual machine and essentially controlling access to physical hardware resources on behalf of the individual virtual machines. The virtualization application also engages in a process known as *binary rewriting* which involves scanning the instruction stream of the executing guest system and replacing any privileged instructions with safe emulations. This has the effect of making the guest system think it is running directly on the system hardware, rather than in a virtual machine within an application.

Some examples of software virtualization technologies include VMware Server, VirtualPC and VirtualBox.

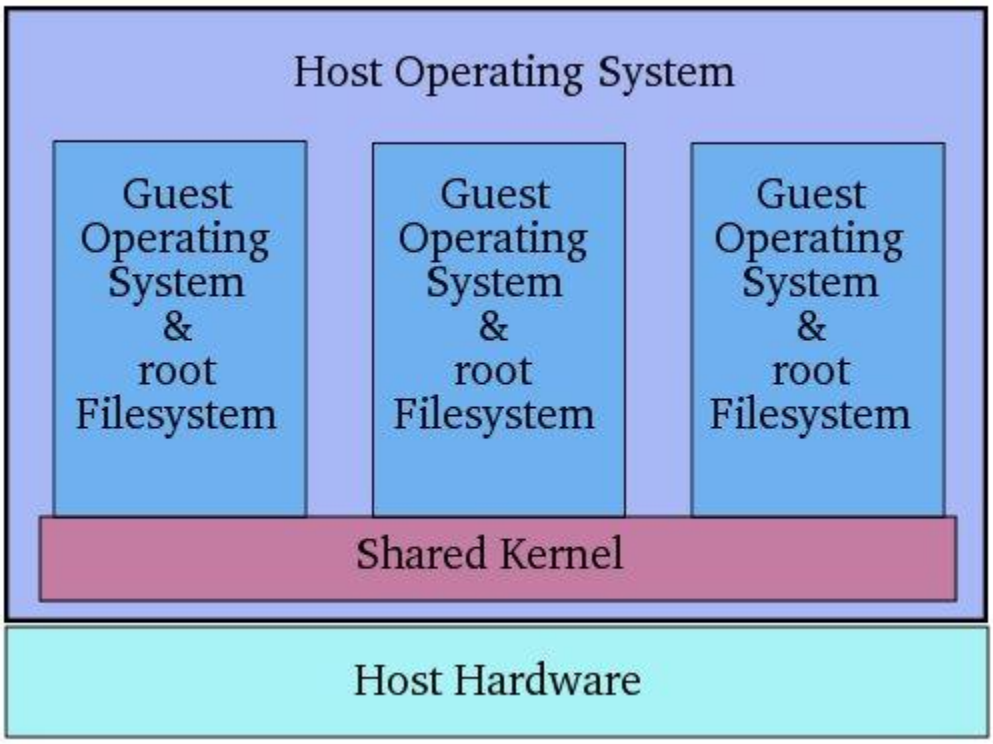
The following figure provides an illustration of software based virtualization:



As outlined in the above diagram, the guest operating systems operate in virtual machines within the virtualization application which, in turn, runs on top of the host operating system. Clearly, the multiple layers of abstraction between the guest operating systems and the underlying host hardware are not conducive to high levels of virtual machine performance. This technique does, however, have the advantage that no changes are necessary to either host or guest operating systems and no special CPU hardware virtualization support is required.

1.5 Shared Kernel Virtualization

Shared kernel virtualization (also known as system level or operating system virtualization) takes advantage of the architectural design of Linux and UNIX based operating systems. In order to understand how shared kernel virtualization works it helps to first understand the two main components of Linux or UNIX operating systems. At the core of the operating system is the *kernel*. The kernel, in simple terms, handles all the interactions between the operating system and the physical hardware. The second key component is the *root filesystem* which contains all the libraries, files and utilities necessary for the operating system to function. Under shared kernel virtualization the virtual guest systems each have their own *root filesystem* but share the kernel of the host operating system. This structure is illustrated in the following architectural diagram:



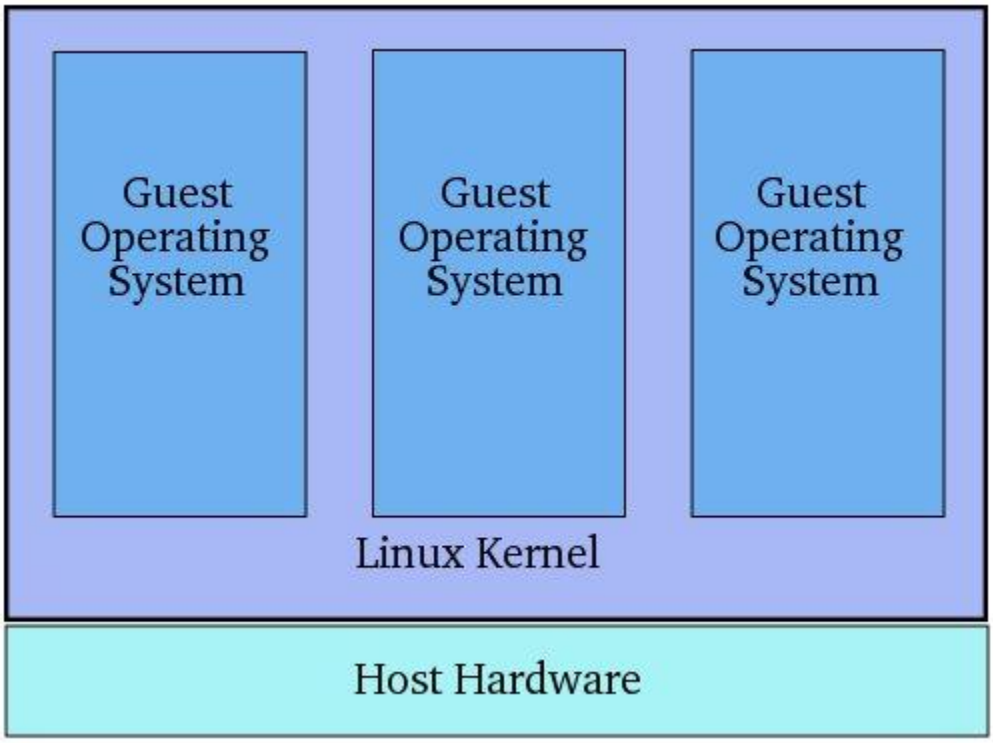
This type of virtualization is made possible by the ability of the kernel to dynamically change the current root filesystem (a concept known as *chroot*) to a different root filesystem without having to reboot the entire system. Essentially, shared kernel virtualization is an extension of this capability. Perhaps the biggest single drawback of this form of virtualization is the fact that the guest operating systems must be compatible with the version of the kernel which is being shared. It is not, for example, possible to run Microsoft Windows as a guest on a Linux system using the shared kernel approach. Nor is it possible for a Linux guest system designed for the 2.6 version of the kernel to share a 2.4 version kernel.

Linux VServer, Solaris Zones and Containers, FreeVPS and OpenVZ are all examples shared kernel virtualization solutions.

1.6 Kernel Level Virtualization

Under kernel level virtualization the host operating system runs on a specially modified kernel which contains extensions designed to manage and control multiple virtual machines each containing a guest operating system. Unlike shared kernel virtualization each guest runs its own kernel, although similar restrictions apply in that the guest operating systems must have been compiled for the same hardware as the kernel in which they are running. Examples of kernel level virtualization technologies include User Mode Linux (UML) and Kernel-based Virtual Machine (KVM).

The following diagram provides an overview of the kernel level virtualization architecture:



1.7 Hypervisor Virtualization

The x86 family of CPUs provide a range of *protection levels* also known as *rings* in which code can execute. Ring 0 has the highest level privilege and it is in this ring that the operating system kernel normally runs. Code executing in ring 0 is said to be running in *system space*, *kernel mode* or *supervisor mode*. All other code such as applications running on the operating system operate in less privileged rings, typically ring 3.

Under hypervisor virtualization a program known as a *hypervisor* (also known as a type 1 Virtual Machine Monitor or VMM) runs directly on the hardware of the host system in ring 0. The task of this hypervisor is to handle resource and memory allocation for the virtual machines in addition to providing interfaces for higher level administration and monitoring tools.

Clearly, with the hypervisor occupying ring 0 of the CPU, the kernels for any guest operating systems running on the system must run in less privileged CPU rings. Unfortunately, most operating system kernels are written explicitly to run in ring 0 for the simple reason that they need to perform tasks that are only available in that ring, such as the ability to execute privileged CPU instructions and directly manipulate memory. A number of different solutions to this problem have been devised in recent years, each of which is described below:

1.7.1 Paravirtualization

Under paravirtualization the kernel of the guest operating system is modified specifically to run on the hypervisor. This typically involves replacing any privileged operations that will only run in ring 0 of the CPU with calls to the hypervisor (known as *hypercalls*). The hypervisor in turn performs the task on behalf of the guest kernel. This typically limits support to open source operating systems such as Linux which may be freely altered and proprietary operating systems where the owners have agreed to make the necessary code modifications to target a specific hypervisor. These issues notwithstanding, the ability of the guest kernel to communicate directly with the hypervisor results in greater performance levels than other virtualization approaches.

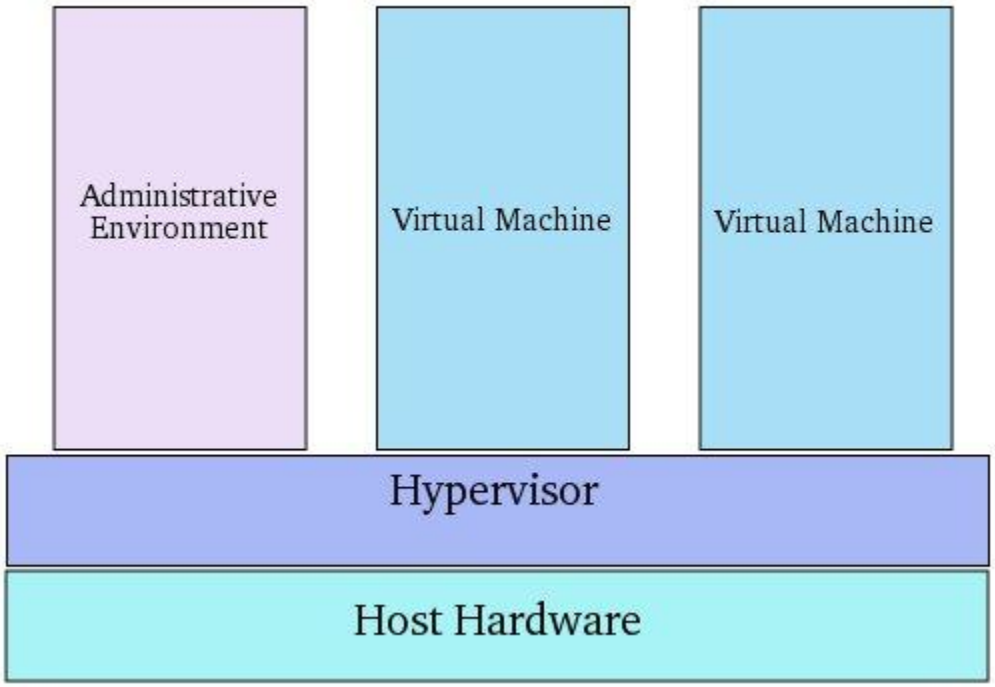
1.7.2 Full Virtualization

Full virtualization provides support for unmodified guest operating systems. The term *unmodified* refers to operating system kernels which have not been altered to run on a hypervisor and therefore still execute privileged operations as though running in ring 0 of the CPU. In this scenario, the hypervisor provides CPU emulation to handle and modify privileged and protected CPU operations made by unmodified guest operating system kernels. Unfortunately this emulation process requires both time and system resources to operate resulting in inferior performance levels when compared to those provided by paravirtualization.

1.7.3 Hardware Virtualization

Hardware virtualization leverages virtualization features built into the latest generations of CPUs from both Intel and AMD. These technologies, known as Intel VT and AMD-V respectively, provide extensions necessary to run unmodified guest virtual machines without the overheads inherent in full virtualization CPU emulation. In very simplistic terms these new processors provide an additional privilege mode above ring 0 in which the hypervisor can operate essentially leaving ring 0 available for unmodified guest operating systems.

The following figure illustrates the hypervisor approach to virtualization:



As outlined in the above illustration, in addition to the virtual machines, an administrative operating system and/or management console also runs on top of the hypervisor allowing the virtual machines to be managed by a system administrator. Hypervisor based virtualization solutions include Xen, VMware ESX Server and Microsoft's Hyper-V technology.