CHAPTER 2

Comparing Virtualization Technologies

With this chapter, we begin our exploration of several popular virtualization strategies and explain how each works. The aim is to bring you the operational information you need to make informed choices for your strategy. Each vendor's software has its own interface (console), its own methods of building, importing, and altering virtual machines (VM), and its own idiosyncrasies, tweaks, and tools.

This chapter gives you a vendor-neutral but technical overview of the types of virtualization available. We approach the various types of virtualization from an application and performance perspective—in other words, a practical look at each technology and its implication for you. Each section also includes at least two representative examples of that technology.

Guest OS/Host OS

Virtualization aficionados perhaps know Guest OS/Host OS as *classic* or hosted virtualization. This type of virtualization relies on an existing operating system (the host operating system), a third-party virtualization software solution, and creation of various guest operating systems. Each guest runs on the host using shared resources donated to it by the host.

Guests usually consist of one or more virtual disk files and a VM definition file. VMs are centrally managed by a host application that sees and manages each VM as a separate application.

Guest systems are fully virtualized in this scenario and have no knowledge of their virtual status. Guests assume they are standalone systems with their own hardware. They are also not aware of other guests on the system unless it's via another guest's network services.

The greatest advantage of this kind of virtualization is that there are a limited number of devices and drivers to contend with. Each VM (guest) possesses a consistent set of hardware. The major disadvantage is that disk I/O suffers greatly in this particular technology. Nondisk operation speed, however, is near native. Therefore, we tell those who use hosted virtualization to interact with their VMs over the network using Windows Terminal Services (RDP) for Windows VMs or SSH for UNIX and Linux systems.

VMware Server

VMware Server is used throughout this book to illustrate virtualization techniques and technologies. It is a free offering from VMware and is considered an introductory package for use in small environments, testing, or for individuals. It has limited usefulness in large environments because of its memory limitations for VMs and sluggish disk performance. VMware Server supports 64-bit machines as hosts and guests.

Sun xVM (VirtualBox)

VirtualBox, which is now Sun xVM VirtualBox, is one of my favorite virtualization packages. Like VMware Server, it is free and cross-platform, but unlike VMware Server, it is open source. With adjustable video memory, remote device connectivity, RDP connectivity, and snappy performance, it may well be the best hosted virtualization package in your arsenal.

VirtualBox is best suited for small networks and individuals for the same reasons as VMware Server.

Hypervisor

A hypervisor is a bare metal approach to virtualization. Bare metal refers to the server system hardware without any OS or other software installed on it. The best way to describe hypervisor technology is to draw a comparison between it and hosted virtualization. At first glance, the hypervisor seems similar to hosted virtualization, but it is significantly different.

A hypervisor is virtualization software that runs an operating system. Conversely, hosted virtualization utilizes an operating system and runs virtualization software as an application. The hypervisor software is installed to the bare metal; then the operating system is installed, which is itself, a paravirtualized VM. The host operating system, if you can call it that, is designated as VM zero.

A new product, VMware ESXi, implements a bare metal hypervisor without a traditional operating system interface. It installs directly to the hardware in an almost impossibly small 32MB footprint. ESXi must be installed onto hardware that is virtualization optimized. VM management is performed via Direct Console User Interface (DCUI), which is the low-level configuration and management interface performed at the physical console of the server system. The VMkernel allows for remote management via a set of APIs and agents.

Citrix Xen

Xen versions 3.0 and earlier weren't particularly interesting to me because they were somewhat difficult to use and didn't seem to perform all that well for my specific applications. Xen 4.x products, however, have converted me heart and soul. The graphical interface is intuitive, fast, and extremely well thought out. The template engine in the new product is a pleasure to use, and provisioning a new VM with it is fast, fast, fast. If you have a need for high-end virtualization, you must check it out.

VMware ESX/VMware ESXi

Enterprise virtualization at its finest is brought to you by the people who breathed life into PC-based virtualization. ESX is a mature product that is rivaled only by Xen at this level of virtualization. Both products require 64-bit architecture, but ESXi has very special hardware requirements beyond those of ESX. ESXi is now a free product.

Microsoft Hyper-V

Microsoft steps up to the plate with its Windows 2008 Server family and Hyper-V virtualization solution where Citrix and VMware fall short: a Windows-based Enterprise virtualization product. Both Citrix Xen and VMware are Linux-based, which means that if you aren't familiar with Linux or UNIX commands, you may be better off using the Microsoft product.

This product, when more mature, promises to be a formidable challenge to VMware's and Xen's dominance in the Enterprise virtualization world

Emulation

Emulation refers to the capability to mimic a particular type of hardware for an operating system regardless of the underlying host operating system. For example, using an emulation virtualization solution, you can install a Sparc version of the Solaris operating system on a non-Sparc host computer. The emulation software runs as an application on the host system, but emulates an entire computer of

another platform. The guest operating system has no awareness of its status as a guest operating system or that it is running in a foreign environment.

In some cases, hardware emulation can be painfully slow, but newer technology, updated emulation software and drivers, and faster 64-bit host processors make emulation a viable virtualization option—especially for those who need to develop drivers or technologies for other platforms without a large investment in support staff or hardware for them.

The best examples of hardware emulation software are Bochs (http://bochs.sourceforge.net) and QEMU (http://bellard.org/qemu).

Bochs

Bochs is a free, open source, Intel architecture x86 (32-bit) emulator that runs on UNIX and Linux, Windows, and Mac OS X, but only supports x86-based operating systems. Bochs is a very sophisticated piece of software and supports a wide range of hardware for emulating all x86 processors and x86_64 processor architecture. It also supports multiple processors but doesn't take full advantage of SMP at this time.

QEMU

QEMU is another free, open source emulation program that runs on a limited number of host architectures (x86, x86_64, and PowerPC) but offers emulation for x86, x86_64, ARM, Sparc, PowerPC, MIPS, and m68k guest operating systems.

Microsoft Virtual PC and Virtual Server

Virtual PC is a free virtualization software package from Microsoft. Virtual PC uses emulation to provide its VM environment. These are good solutions for hosting a few VMs on a Windows XP Workstation or Windows 2003 Server. It isn't a large environment solution by any stretch of the imagination, but it can get some VMs up and running cheaply and in very short order.

VM performance on these products is surprisingly good for Windows VMs. It is difficult, if not impossible, to tell that you are using a VM when connecting over the network. Console performance tends to be a little sluggish at times—so whenever possible, minimize the console and use RDP to connect to your virtualized Windows systems.

KERNEL-LEVEL

Kernel-level virtualization is kind of an oddball in the virtualization world in that each VM uses its own unique kernel to boot the guest VM (called a root file system) regardless of the host's running kernel.

KVM

Linux KVM (Kernel Virtual Machine) is a modified QEMU, but unlike QEMU, KVM uses virtualization processor extensions (Intel-VT and AMD-V). KVM supports a large number of x86 and x86_64 architecture guest operating systems, including Windows, Linux, and FreeBSD. It uses the Linux kernel as a hypervisor and runs as a kernel loadable module.

User-Mode Linux

User-mode Linux (UML) uses an executable kernel and a root file system to create a VM. To create a VM, you need a user-space executable kernel (guest kernel) and a UML-created root file system. These two components together make up a UML VM. The command-line terminal session you use to connect to the remote host system becomes your VM console. UML is included with all 2.6.x kernels.

SHARED KERNEL

Shared kernel virtualization, also called operating system virtualization or system level virtualization, takes advantage of the unique ability of UNIX and Linux to share their kernels with other processes on the system. This shared kernel virtualization is achieved by using a feature called change root (chroot). The chroot feature changes the root file system of a process to isolate it in such a way as to provide some security. It (chroot) is often called a chroot jail or container-based virtualization. A chrooted program, set of programs, or entire system in the case of shared kernel virtualization is protected by setting up the chrooted system to *believe* that it is a standalone machine with its own root file system.

The chroot mechanism has been enhanced to mimic an entire file system so that an entire system can be chrooted, hence creating a VM. The technical advantages and disadvantages of shared kernel virtualization are listed next:

- Advantages
  Enhanced Security and Isolation
  Native Performance

Higher Density of Virtualized Systems

- Disadvantages

  Host Kernel and Guest Compatibility

The chroot system offers much in the way of enhanced security features and isolation; however, the greatest advantages of shared kernel virtualization are not in its security, although that's certainly important to consider, but in its performance. With this kind of virtualization, you'll get native performance for each individual system. Not only does each system perform at native speeds, but you can also have more than the standard number of VMs on a host system. By standard number, we mean the number that you could logically have on a host system if you used memory as the limiting factor—leaving 1GB for the host and taking the rest of the RAM for VMs.

The limit of the number of chrooted systems you can have on a host system more closely resembles a standalone system supporting multiple applications. If you think of each chroot system as an application instead of a VM, you'll more accurately allocate resources and enjoy performance that surpasses many other types of virtualization.

The disadvantage of shared kernel virtualization is a big one: All VMs have to be compatible with your running kernel. In other words, you can't run Windows operating systems, Solaris, Mac OS X, or any other operating system that couldn't run your system's kernel on its own. Major web hosting providers have run this scenario for years so that customers get their own virtual server for their hosting needs. They don't know that the system is virtual, nor can they contact the host system through their VM.

Solaris Containers (Zones)

Solaris 10 comes with built-in virtualization. The Solaris 10 operating system, itself, is known as the Global Zone. Solaris Zones are actually BSD jails, each with its own virtual root that mimics a complete operating system and file system. When you create a new zone, a full file system is copied to the new zone directory. Each zone sees only its own processes and file systems. The zone believes that it is a full, independent operating system; only the Global Zone has any knowledge of virtualization.

Each zone essentially creates a clean sandbox in which you may install applications, provide services, or test patches. Solaris zones are a scalable, enterprise-level virtualization solution providing ease of use and native performance.

OpenVZ

We use the OpenVZ kernel on my personal Linux server system. The OpenVZ kernel is optimized for virtualization and proves to be extremely efficient at handling VM performance for other virtualization products as well.

On my personal Linux server system, we run VMware Server, Sun's xVM, and QEMU. Before we installed the OpenVZ kernel, we had many CPU-related performance problems with some of my VMs. OpenVZ is similar to Solaris Zones except that you can run different Linux distributions under the same kernel. Various distribution templates are available on the OpenVZ website at www.openvz.org.

In the Virtual Trenches

As someone who works with virtualization software on a daily basis, we can give you some pointers, opinions, and suggestions for your environment. These are from my experiences; they may be biased, and, as always, your mileage may vary.

For true Enterprise-ready virtualization, you can't beat Xen or VMware ESX. They are robust, easy to use, well supported, well documented, and ready to go to work for you. Hypervisor technology is absolutely the right decision if you need to virtualize multiple operating systems on one host system. They are both costly solutions but well worth the price you pay for the performance you receive. You should use this technology in situations where disk I/O is of major concern.

As to which one of the hypervisor technologies we prefer, we're afraid that we can't answer that for you. Either one you choose will serve you well.

Solaris Zones (containers), and any jail-type virtualization, works extremely well for UNIX host systems where you want a consistent and secure environment with native performance. Kernel-level virtualization is extremely well suited for isolating applications from each other and the global zone (host operating system). This type of virtualization is an excellent choice for anyone who wants to get acquainted with virtualization for no money, little hassle, and ease of use. We highly recommend this virtualization method for your Solaris 10 systems.

Microsoft Virtual PC and VMware Server are great choices for testing new applications, services, patches, service packs, and much more. We use Virtual PC and VMware Server on a daily basis and can't live without them. We wouldn't recommend either for heavy production or Enterprise use, but for smaller environments, desktops, or IT laboratories, you can't go wrong with these. They're free, easy to use, durable, and can host a wide range of guest operating systems. In this same arena, Sun's xVM is also very good.

VMware Server and Sun xVM are both available on multiple platforms, whereas Virtual PC is available only for Windows.

We deliberately left out several other virtualization products from this dialog. Either we've had less experience with them or less good experience with them than the others mentioned previously, and we don't want to keep you from investigating them on your own. We are not diminishing their value or importance for viable virtualization solutions, but we just don't feel qualified to speak for or against them in this context.

Summary

This chapter was an overview of virtualization technology from a vendor-neutral perspective. There is always the question of which virtualization software is best. There is no single correct answer to this question unless it is either emotionally based or prejudicial in some way.

All virtualization software does the same thing: virtualize physical machines and the services that they provide. You'll have to decide what you need from virtualization and then choose the best technology that fits that need—and worry about vendor specifics later. You may also use more than one virtualization solution to solve the various needs within your network.

If you're going to invest thousands, perhaps hundreds of thousands, in virtualization, you need to experience the software for yourself. Vendors know this and are willing to work with you. Many offer full versions for a trial period. If a trial version won't work for you, get in touch with the vendor and get the actual licensed software for evaluation.