1

Overview of Virtualization

Virtualization is one of the hottest topics in information technology today. The increasing speed and capabilities of today's x86 processors have made virtualization possible on commodity hardware, and virtualization provides an attractive way of making the most of that hardware.

As happens with most popular IT buzzwords in their heyday, virtualization is frequently touted as *the* solution to all of your IT woes. I'm sure we'd all like to find some one true technology that enables us to set up a powerful, reliable, and highly available IT infrastructure once and for all, so that we can all go home or at least throttle down our hours so that we can work 9-to-5. I have both good news and bad news. The good news is that virtualization is a cool and powerful technology that can indeed simplify your computing infrastructure and help you get the most bang for your buck out of the latest, fastest processors, networking, and storage technologies. The bad news is that, like anything in the real world, successfully implementing, deploying, and supporting a new IT infrastructure based on virtualization requires the same level of planning and system design that any basic shift in infrastructure always has. That's why we're all here and why this book was written in the first place — to define basic concepts, explain basic issues, and explore how to successfully make the most of the best of today's virtualization technologies while providing the capability and availability guarantees that today's often-complex IT infrastructure requires.

This chapter provides an overview of the history of virtualization and the different types of virtualization that are in use today. Following this foundation, I'll discuss the pros and cons of virtualization — regardless of what hottechnologybuzzword.com may say, there are downsides, or at least issues that you have to be aware of when thinking about introducing virtualization into your computing environment. Although this book is about Xen, this chapter primarily focuses on providing a thorough background on virtualization concepts and theory rather than any specific technology; it remains relatively agnostic to any specific approach to virtualization or its implementation. Don't worry — as you might expect (and hope), the rest of this book focuses on Xen, why it is the most attractive of today's virtualization solutions, and how to successfully make the most of Xen in your computing environment.

What Is Virtualization?

Virtualization is simply the logical separation of the request for some service from the physical resources that actually provide that service. In practical terms, virtualization provides the ability to run applications, operating systems, or system services in a logically distinct system environment that is independent of a specific physical computer system. Obviously, all of these have to be running on a certain computer system at any given time, but virtualization provides a level of logical abstraction that liberates applications, system services, and even the operating system that supports them from being tied to a specific piece of hardware. Virtualization's focus on logical operating environments rather than physical ones makes applications, services, and instances of an operating system portable across different physical computer systems.

The classic example of virtualization that most people are already familiar with is virtual memory, which enables a computer system to appear to have more memory than is physically installed on that system. Virtual memory is a memory-management technique that enables an operating system to see and use noncontiguous segments of memory as a single, contiguous memory space. Virtual memory is traditionally implemented in an operating system by paging, which enables the operating system to use a file or dedicated portion of some storage device to save pages of memory that are not actively in use. Known as a "paging file" or "swap space," the system can quickly transfer pages of memory to and from this area as the operating system or running applications require access to the contents of those pages. Modern operating systems such as UNIX-like operating systems (including Linux, the *BSD operating systems, and Mac OS X) and Microsoft Windows all use some form of virtual memory to enable the operating system and applications to access more data than would fit into physical memory.

As I'll discuss in the next few sections, there are many different types of virtualization, all rooted around the core idea of providing logical access to physical resources. Today, virtualization is commonly encountered in networking, storage systems, and server processes, at the operating system level and at the machine level. Xen, the subject of this book, supports machine-level virtualization using a variety of clever and powerful techniques.

As a hot buzzword, it's tempting for corporate marketing groups to abuse the term "virtualization" in order to get a bit more traction for their particular products or technologies. The use of the term "virtualization" in today's marketing literature rivals the glory days of terms such as "Internet" and "network-enabled" in the 1990s. To try to cut through the haze surrounding what is and what is not virtualization, the next few sections discuss the most common classes of virtualization and virtualization technology today.

> *Whenever possible, references in this section to virtualization technologies refer to centralized resources for that term or technology. I've tried to use the Wikipedia entries as a primary reference for most terms and technologies because in most cases, the Wikipedia provides a great, product-agnostic resource that doesn't promote any single technical solution for a given technology. When looking things up on Wikipedia, be aware that terms there are case-sensitive — and that Wikipedia is only a starting point for good information.*

Application Virtualization

The term "application virtualization" describes the process of compiling applications into machine-independent byte code that can subsequently be executed on any system that provides the appropriate virtual machine as an execution environment. The best known example of this approach to virtualization

is the byte code produced by the compilers for the Java programming language (`http://java.sun.com/`), although this concept was actually pioneered by the UCSD P-System in the late 1970s (`www.threedee.com/jcm/psystem`), for which the most popular compiler was the UCSD Pascal compiler. Microsoft has even adopted a similar approach in the Common Language Runtime (CLR) used by .NET applications, where code written in languages that support the CLR are transformed, at compile time, into CIL (Common Intermediate Language, formerly known as MSIL, Microsoft Intermediate Language). Like any byte code, CIL provides a platform-independent instruction set that can be executed in any environment supporting the .NET Framework.

Application virtualization is a valid use of the term "virtualization" because applications compiled into byte code become logical entities that can be executed on different physical systems with different characteristics, operating systems, and even processor architectures.

Desktop Virtualization

The term "desktop virtualization" describes the ability to display a graphical desktop from one computer system on another computer system or smart display device. This term is used to describe software such as Virtual Network Computing (VNC, `http://en.wikipedia.org/wiki/VNC`), thin clients such as Microsoft's Remote Desktop (`http://en.wikipedia.org/wiki/Remote_Desktop_Protocol`) and associated Terminal Server products, Linux terminal servers such as the Linux Terminal Server project (LTSP, `http://sourceforge.net/projects/ltsp/`), NoMachine's NX (`http://en.wikipedia.org/wiki/NX_technology`), and even the X Window System (`http://en.wikipedia.org/wiki/X_Window_System`) and its XDMCP display manager protocol. Many window managers, particularly those based on the X Window System, also provide internal support for multiple, virtual desktops that the user can switch between and use to display the output of specific applications. In the X Window System, virtual desktops were introduced in versions of Tom LeStrange's TWM window manager (`www.xwinman.org/vtwm.php`, with a nice family tree at `www.vtwm.org/vtwm-family.html`), but are now available in almost every other window manager. The X Window System also supports desktop virtualization at the screen or display level, enabling window managers to use a display region that is larger than the physical size of your monitor.

In my opinion, desktop virtualization is more of a bandwagon use of the term "virtualization" than an exciting example of virtualization concepts. It does indeed make the graphical console of any supported system into a logical entity that can be accessed and used on different physical computer systems, but it does so using standard client/server display software. The remote console, the operating system it is running, and the applications you execute are actually still running on a single, specific physical machine — you're just looking at them from somewhere else. Calling remote display software a virtualization technology seems to me to be equivalent to considering a telescope to be a set of virtual eyeballs because you can look at something far away using one. Your mileage may vary.

Network Virtualization

The term "network virtualization" describes the ability to refer to network resources logically rather than having to refer to specific physical network devices, configurations, or collections of related machines. There are many different levels of network virtualization, ranging from single-machine, network-device virtualization that enables multiple virtual machines to share a single physical-network resource, to enterprise-level concepts such as virtual private networks and enterprise-core and edge-routing techniques for creating subnetworks and segmenting existing networks.

Xen relies on network virtualization through the Linux bridge-utils package to enable your virtual machines to appear to have unique physical addresses (Media Access Control, or MAC, addresses) and unique IP addresses. Other server-virtualization solutions, such as UML, use the Linux virtual Point-to-Point (TUN) and Ethernet (TAP) network devices to provide user-space access to the host's network. Many advanced network switches and routers use techniques such as Virtual Routing and Forwarding (VRF), VRF-Lite, and Multi-VRF to segregate customer traffic into separately routed network segments and support multiple virtual-routing domains within a single piece of network hardware.

Discussing virtual private networks and other virtual LAN technologies is outside the scope of this book. Virtual networking as it applies to and is used by Xen is discussed in Chapter 8.

Server and Machine Virtualization

The terms "server virtualization" and "machine virtualization" describe the ability to run an entire virtual machine, including its own operating system, on another operating system. Each virtual machine that is running on the parent operating system is logically distinct, has access to some or all of the hardware on the host system, has its own logical assignments for the storage devices on which that operating system is installed, and can run its own applications within its own operating environment.

Server virtualization is the type of virtualization technology that most people think of when they hear the term "virtualization", and is the type of virtualization that is the focus of this book. Though not as common, I find the term "machine virtualization" useful to uniquely identify this type of virtualization, because it more clearly differentiates the level at which virtualization is taking place — the machine itself is being virtualized — regardless of the underlying technology used. Machine virtualization is therefore the technique used by virtualization technologies such as KVM, Microsoft Virtual Server and Virtual PC, Parallels Workstation, User Mode Linux, Virtual Iron, VMware, and (of course) Xen. See the section "Other Popular Virtualization Software" in Chapter 2 for an overview of each of these virtualization technologies, except for Xen, which (as you might hope) is discussed throughout this book.

> *In the maddening whirlwind of terms that include the word "virtual," server virtualization is usually different from the term "virtual server," which is often used to describe both the capability of operating system servers such as e-mail and Web servers to service multiple Internet domains, and system-level virtualization techniques that are used to provide Internet service provider (ISP) users with their own virtual server machine.*

The key aspect of server or machine virtualization is that different virtual machines do not share the same kernel and can therefore be running different operating systems. This differs from system-level virtualization, where virtual servers share a single underlying kernel (discussed in more detail later in this chapter) and provide a number of unique infrastructure, customer, and business opportunities. Some of these are:

- Running legacy software, where you depend on a software product that runs only on a specific version of a specific operating system. Being able to run legacy software and the legacy operating system that it requires is only possible on virtual systems that can run multiple operating systems.
- Software system-test and quality-assurance environments, where you need to be able to test a specific software product on many different operating systems or versions of an operating system. Server virtualization makes it easy to install and test against many different operating systems or versions of operating systems without requiring dedicated hardware for each.

- Low-level development environments, where developers may want or need to work with specific versions of tools, an operating system kernel, and a specific operating system distribution. Server virtualization makes it easy to be able to run many different operating systems and environments without requiring dedicated hardware for each.

For more information about specific uses for server virtualization and its possible organizational advantages, see the section "Advantages of Virtualization," later in this chapter.

Server and machine virtualization technologies work in several different ways. The differences between the various approaches to server or machine virtualization can be subtle, but are always significant in terms of the capabilities that they provide and the hardware and software requirements for the underlying system. The most common approaches to server and machine virtualization today are the following:

- **Guest OS:** Each virtual server runs as a separate operating system instance within a virtualization application that itself runs on an instance of a specific operating system. Parallels Workstation, VMWare Workstation, and VMWare GSX Server are the most common examples of this approach to virtualization. The operating system on which the virtualization application is running is often referred to as the "Host OS" because it is supplying the execution environment for the virtualization application.
- **Parallel Virtual Machine:** Some number of physical or virtual systems are organized into a single virtual machine using clustering software such as a Parallel Virtual Machine (PVM) (`www.csm.ornl.gov/pvm/pvm_home.html`). The resulting cluster is capable of performing complex CPU and data-intensive calculations in a cooperative fashion. This is more of a clustering concept than an alternative virtualization solution, and thus is not discussed elsewhere in this book. See the PVM home page (`www.csm.ornl.gov/pvm/`) for detailed information about PVM and related software.
- **Hypervisor-based:** A small virtual machine monitor (known as a hypervisor) runs on top of your machine's hardware and provides two basic functions. First, it identifies, traps, and responds to protected or privileged CPU operations made by each virtual machine. Second, it handles queuing, dispatching, and returning the results of hardware requests from your virtual machines. An administrative operating system then runs on top of the hypervisor, as do the virtual machines themselves. This administrative operating system can communicate with the hypervisor and is used to manage the virtual machine instances.

  The most common approach to hypervisor-based virtualization is known as paravirtualization, which requires changes to an operating system so that it can communicate with the hypervisor. Paravirtualization can provide performance enhancements over other approaches to server and machine virtualization, because the operating system modifications enable the operating system to communicate directly with the hypervisor, and thus does not incur some of the overhead associated with the emulation required for the other hypervisor-based machine and server technologies discussed in this section. Paravirtualization is the primary model used by Xen, which uses a customized Linux kernel to support its administrative environment, known as domain0. As discussed later in this section, Xen can also take advantage of hardware virtualization to run unmodified versions of operating systems on top of its hypervisor.
- **Full virtualization:** Very similar to paravirtualization, full virtualization also uses a hypervisor, but incorporates code into the hypervisor that emulates the underlying hardware when necessary, enabling *unmodified* operating systems to run on top of the hypervisor. Full virtualization is the model used by VMWare ESX server, which uses a customized version of Linux (known as the Service Console) as its administrative operating system.

- **Kernel-level virtualization:** This type of virtualization does not require a hypervisor, but instead runs a separate version of the Linux kernel and an associated virtual machine as a user-space process on the physical host. This provides an easy way to run multiple virtual machines on a single host. Examples of this are User-Mode Linux (UML), which has been supported in the mainline Linux kernel for quite a while but requires a special build of the Linux kernel for guest operating systems, and Kernel Virtual Machine (KVM), which was introduced in the 2.6.20 mainline Linux kernel. UML does not require any separate administrative software in order to execute or manage its virtual machines, which can be executed from the Linux command line. KVM uses a device driver in the host's kernel for communication between the main Linux kernel and the virtual machines, requires processor support for virtualization (Intel VT or AMD-v Pacifica), and uses a slightly modified QEMU process as the display and execution container for its virtual machines. In many ways, KVM's kernel-level virtualization is a specialized version of full virtualization, where the Linux kernel serves as the hypervisor, but I think that UML and KVM are unique enough to merit their own class of server virtualization. For more information about the Intel and AMD hardware that supports hardware virtualization, see the section "Hardware Requirements for Xen" in Chapter 3.
- **Hardware virtualization:** Very similar to both paravirtualization and full virtualization, hardware virtualization uses a hypervisor, but it is only available on systems that provide hardware support for virtualization. Hypervisor-based systems such as Xen and VMWare ESX Server, and kernel-level virtualization technologies such as KVM, can take advantage of the hardware support for virtualization that is provided on the latest generation of Intel (Intel VT, aka Vanderpool) and AMD (AMD-V, aka Pacifica) processors. Virtual machines in a hardware virtualization environment can run unmodified operating systems because the hypervisor can use the hardware's support for virtualization to handle privileged and protected operations and hardware access requests, and to communicate with and manage the virtual machines. For more information about the Intel and AMD hardware that supports hardware virtualization, see the section "Hardware Requirements for Xen" in Chapter 3.

As you can see from the previous list, hypervisor-based virtualization is the most popular virtualization technique in use today, spanning the best-known server and machine virtualization technologies, including IBM's VM operating system, VMWare's ESX Server, Parallels Workstation, Virtual Iron products, and Xen. The use of a hypervisor was pioneered by the original commercial virtual-machine environment, IBM's CP/CMS operating system (`http://en.wikipedia.org/wiki/CP/CMS`), introduced in 1966, was popularized by IBM's VM/370 operating system (`http://en.wikipedia.org/wiki/VM_%28operating_system%29`), introduced in 1970, and remains a great idea today.

Standardizing Linux Server Virtualization

Although Xen is an open source project, keeping up with both the Xen patches and the latest revision of the Linux kernel is tough. The increasing popularity of Xen has made many people hope for the direct inclusion of the Xen patches into the mainline kernel sources. However, the folks from VMware aren't among them because the inclusion of Xen-specific patches could conceivably give Xen (and thus XenSource) a commercial edge over the VMware technologies. As you might hope, the Linux kernel is a truly open effort whose goal is open APIs and general, vendor-agnostic functionality, and

should therefore be capable of supporting more than one hypervisor-based virtualization solution.

In 2006, VMware proposed a generic Virtual Machine Interface (VMI) that would enable multiple hypervisor-based virtualization technologies to use a common kernel-level interface. This didn't quite suit the Xen folks, so much wailing, gnashing of teeth, and rattling of swords ensued. Finally, at the 2006 USENIX meeting, VMware and Xen agreed to work together (with others) to develop a more generic interface, known as paravirt_ops, which is being developed by IBM, VMware, Red Hat, and XenSource and is being coordinated by Rusty Russell, a well-known Linux kernel hacker. For a detailed discussion of paravirt_ops, see the section "Other Popular Virtualization and Emulation Software" in Chapter 2.

The upshot of all of this is that the eventual inclusion of the paravirt_ops patches into the mainline kernel will enable any compliant hypervisor-based virtualization technology to work with a vanilla Linux kernel, while kernel projects such as KVM will enable users to run virtual machines, themselves running any operating system, on hardware that supports them, without requiring a hypervisor. UML will continue to enable users to run additional Linux virtual machines on a single Linux system. Though this may appear confusing, increasing richness in mainline Linux support for virtualization simply falls in the "more is better" category, and enables hypervisor-based virtualization technologies to compete on their technical and administrative merits.

Storage Virtualization

Storage virtualization is the logical abstraction of physical storage. In conjunction with different types of filesystems, storage virtualization is the key to making flexible, expandable amounts of storage available to today's computer systems.

Storage virtualization has been around for many years, and should be familiar to anyone who has worked with RAID storage (`http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks`), logical volumes (`http://en.wikipedia.org/wiki/Logical_volume`) on systems such as Linux or AIX, or with networked filesystems such as AFS (`http://en.wikipedia.org/wiki/Andrew_file_system`) and GFS (`http://en.wikipedia.org/wiki/Global_File_System`). All of these technologies combine available physical disk drives into pools of available storage that can be divided into logical sections known as volumes on which a filesystem can be created and mounted for use on a computer system. A volume is the logical equivalent of a disk partition.

The core features that make storage virtualization so attractive in today's enterprise environments is that they provide effectively infinite storage that is limited only by the number and size of drives that can be physically supported by the host system or host storage system. The reporting and discovery requirements imposed by standards such as Sarbanes-Oxley, the Department of Homeland Security, or basic corporate accountability make it important to be able to store more and more information forever. Storage virtualization enables greater amounts of physical storage to be available to individual systems, and enables existing filesystems to grow to hold that information without resorting to an administrative shotgun blast of symbolic links and interdependent mount points for networked storage.

Technologies such as RAID (Redundant Array of Inexpensive Disks) and logical volumes as provided by the Linux LVM, LVM2, and EVMS packages are usually limited to use on the system to which the actual storage devices are physically attached. Some RAID controllers are dual-ported, allowing multiple computers access to the same volumes and associated filesystems through that RAID controller, although how well this works depends on the type of filesystem in use on the shared volume and how that filesystem is mounted on both systems.

To use a logical volume manager, you must define the disk partitions that you want to use for logical volumes, create logical volumes on that physical storage, and then create a filesystem on the logical volumes. You can then mount and use these filesystems just as you would mount and use filesystems that were created on physical disk partitions.

Like standard disk controllers, RAID controllers provide block-level access to the storage devices that are attached to them, although the size of the storage available from any set of disks depends on the RAID level that is being used. The devices attached to the RAID controller are then made available to the system as though they were a single disk, which you can then partition as you wish, create filesystems on those partitions, and mount and use them just as you would use single physical partitions.

Operating systems such as Linux also support software RAID, where no physical RAID controller need be present. The software RAID system functions exactly as a hardware RAID controller would, providing block-level access to available storage, but it enforces the characteristics of different RAID levels in software rather than in hardware. Software RAID is very efficient and has only slightly lower performance than many hardware RAID controllers. Many system administrators actually prefer software RAID over hardware RAID because hardware RAID controllers are very different from manufacturer to manufacturer and even controller to controller. The failure of a RAID controller typically requires a replacement controller of the same type from the same manufacturer in order to access the data on the storage device that was attached to the failed controller. On the other hand, software RAID is completely portable across all Linux systems on which the software is installed as long as they support the same physical disk drive interfaces (IDE, EIDE, SATA, and so on).

Distributed filesystem technologies such as AFS and GFS have their own internal logical-volume creation and management mechanisms, and also make it possible to share the filesystems on these logical volumes between multiple computer systems because AFS and GFS provide locking mechanisms to synchronize simultaneous writes to shared filesystems over the network. NFS, the default Network File System for most UNIX-like operating systems, also makes it possible to share logical storage across multiple computer systems, although it does this by exporting a directory from a filesystem on the logical storage rather than by directly mounting a system-specific volume or networked filesystem. Distributed filesystems such as AFS and GFS provide filesystem-level access to logical volumes. In this, they are conceptually similar to Network Attached Storage (NAS, `http://en.wikipedia.org/wiki/Network-attached_storage`) devices, which provide filesystem-level access over a network to the filesystems that they contain.

Storage virtualization has become much more accessible across multiple computer systems with the advent of Storage Area Networks (SAN, `http://en.wikipedia.org/wiki/Storage_area_network`), which support block-level I/O and therefore enable multiple systems to share low-level access to various types of storage devices over the network. Most SANs use expensive, high-power network technologies

such as Fibre Channel (`http://en.wikipedia.org/wiki/Fibre_Channel`) and InfiniBand (`http://en.wikipedia.org/wiki/InfiniBand`) to provide the high levels of throughput and general performance that are most desirable when many systems share access to block- or protocol-level networked storage.

Newer technologies such as iSCSI (Internet Small Computer Systems Interface, `http://en.wikipedia.org/wiki/ISCSI`) and AoE (ATA over Ethernet, `http://en.wikipedia.org/wiki/ATA_over_Ethernet`) provide less expensive mechanisms for getting block-level access to networked storage devices. As the name suggests, iSCSI supports the use of the SCSI protocol over TCP/IP networks, and requires a special type of network controller. AoE provides block-level access to suitable ATA devices using only a standard Ethernet connection. As you'd expect, both of these perform better on higher-bandwidth networks such as Gigabit Ethernet networks, although they are certainly usable on 100-megabit networks. iSCSI and AoE are making networked storage a very real possibility for most of today's data centers and IT infrastructure of any size, and are discussed in more detail in the section "Using Xen and Networked Storage Devices" in Chapter 10.

System-Level or Operating System Virtualization

The system-level virtualization, often referred to as, operating system virtualization, describes various implementations of running multiple, logically distinct system environments on a single instance of an operating system kernel. System-level virtualization is based on the change root (chroot) concept that is available on all modern UNIX-like systems. During the system boot process, the kernel can use root filesystems such as those provided by initial RAM disks or initial RAM filesystems to load drivers and perform other early-stage system initialization tasks. The kernel can then switch to another root filesystem using the chroot command in order to mount an on-disk filesystem as its final root filesystem, and continue system initialization and configuration from within that filesystem. The chroot mechanism as used by system-level virtualization is an extension of this concept, enabling the system to start virtual servers with their own sets of processes that execute relative to their own filesystem root directories. Operating within the confines of their own root directories and associated filesystem prevents virtual servers from being able to access files in each others' filesystems, and thereby provides basic protection from exploits of various server processes or the virtual server itself. Even if a chroot'ed server is compromised, it has access only to files that are located within its own root filesystem.

The core differentiator between system-level and server virtualization is whether you can be running different operating systems on different virtual systems. If all of your virtual servers must share a single copy of an operating system kernel, as shown in Figure 1-1, this is system-level virtualization. If different virtual servers can be running different operating systems, including different versions of a single operating system, this is server virtualization, sometimes also referred to as machine virtualization. Virtualization solutions such as FreeBSD's chroot jails, FreeVPS, Linux VServer, OpenVZ, Solaris Zones and Containers, and Virtuozzo are all examples of system-level virtualization. FreeBSD jails can run logically distinct versions of FreeBSD user-space on top of a single FreeBSD kernel, and can therefore use different instances or versions of libraries, server processes, and applications. Solaris containers and zones all share the same underlying version of Solaris, and can either use completely distinct root filesystems or share portions of a filesystem. Linux-VServer, FreeVPS, and OpenVZ can run different Linux distributions in their virtual servers, but all share the same underlying kernel. All of these are discussed in more detail in the section "Other Popular Virtualization Software" in Chapter 2.

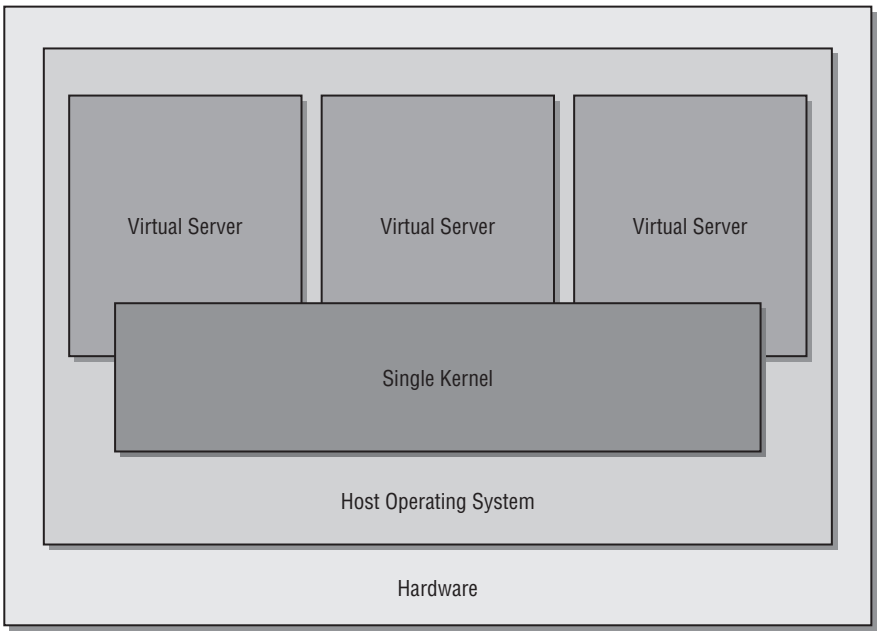

Figure 1-1

System-level virtualization provides some significant advantages over server or machine virtualization. The key to all of these is that, because they share a single instance of an operating system kernel, system-level virtualization solutions are significantly lighter weight than the complete machines (including a kernel) required by server virtualization technologies. This enables a single physical host to support many more "virtual servers" than the number of complete virtual machines that it could support. System-level virtualization solutions such as FreeBSD's chroot jails, Linux-VServer, and FreeVPS have been used for years by businesses such as Internet Service Providers (ISPs) to provide each user with their own virtual server, in which they can have relatively complete control (and, in some cases, administrative privileges) without any chance of compromising the system's primary security configuration, system configuration files, and filesystem. System-level virtualization is therefore most commonly used for server consolidation. The primary disadvantage of system-level virtualization is that a kernel or driver problem can take down all of the system-level virtual servers supported on that system.

Why Virtualization Today?

This section (and the rest of this book) focuses on server and machine virtualization, where a single host system supports multiple, independent instances of virtual machines running various operating systems. Unless otherwise identified, subsequent references to virtualization refer to machine virtualization.

Virtualization is not a new concept, and has been in use for decades in the different ways highlighted in the previous section. However, virtualization is more popular now than ever because it is now an option for a larger group of users and system administrators than ever before. There are several general reasons for the increasing popularity of virtualization:

- The power and performance of commodity x86 hardware continues to increase. Processors are faster than ever, support more memory than ever, and the latest multi-core processors literally enable single systems to perform multiple tasks simultaneously. These factors combine to increase the chance that your hardware may be underutilized. As discussed later in this chapter, virtualization provides an excellent way of getting the most out of existing hardware while reducing many other IT costs.
- The integration of direct support for hardware-level virtualization in the latest generations of Intel and AMD processors, motherboards, and related firmware has made virtualization on commodity hardware more powerful than ever before. See the section "Hardware Requirements for Xen" in Chapter 3 for an overview of virtualization support in commodity hardware.
- A wide variety of virtualization products for both desktop and server systems running on commodity x86 hardware have emerged, are still emerging, and have become extremely popular. Many of these (like Xen) are open source software and are attractive from both a capability and cost perspective. The section "Other Popular Virtualization Software" in Chapter 2 provides an overview of well-known virtualization products (other than Xen) that support commodity hardware.

More accessible, powerful, and flexible than ever before, virtualization is continuing to prove its worth in business and academic environments all over the world. The next two sections explore some of the specific reasons why virtualization can benefit your computing infrastructure and also discuss some of the issues that you must consider before selecting virtualization as a solution to your infrastructure requirements.

Basic Approaches to Virtual Systems

The section "What Is Virtualization?" highlighted the different ways in which the term "virtualization" is popularly used today and discussed different approaches to virtualization in each domain. This section provides a slightly different view of these same concepts, focusing on the type of virtualization that is the topic of this book, where a single physical machine can host multiple virtual machines. This section makes it easier to compare different approaches to running virtual machines on physical hardware by focusing on the underlying technology rather than on terminology and by providing a cheat sheet for general approaches to these types of virtual machines.

The most common approaches to virtual computer systems used today are the following:

- **Shared kernel:** A single operating system kernel supports multiple virtual systems. Each virtual system has its own root filesystem. Because all virtual machines share the same operating system kernel, the libraries and utilities executed by these virtual machines must also have been compiled for the same hardware and instruction set as the physical machine on which the virtual systems are running. For more details on this approach to virtualization and some examples of virtualization software that use this approach, see Figure 1-1 and the section earlier in this chapter entitled "System-Level or Operating System Virtualization." For details on any of

these software packages, see the section "Other Popular Virtualization Software" in the next chapter.

- **Guest OS:** Virtual machines run within an application that is running as a standard application under the operating system that executes on the physical host system. This application manages the virtual machines, mediates access to the hardware resources on the physical host system, and intercepts and handles any privileged or protected instructions issued by the virtual machines. Figure 1-2 illustrates this approach to virtualization. This type of virtualization typically runs virtual machines whose operating system, libraries, and utilities have been compiled for the same type of processor and instruction set as the physical machine on which the virtual systems are running. However, it can also run virtual machines, libraries, and utilities that have been compiled for other processors if the virtualization application can perform instruction-set translation or emulation, as is the case with products such as Microsoft's Virtual PC product. For more information about this approach to virtualization and some examples of virtualization software that uses this approach, see the section "Server or Machine Virtualization" earlier in this chapter. For details on any of these software packages, see the section "Other Popular Virtualization Software" in the next chapter.

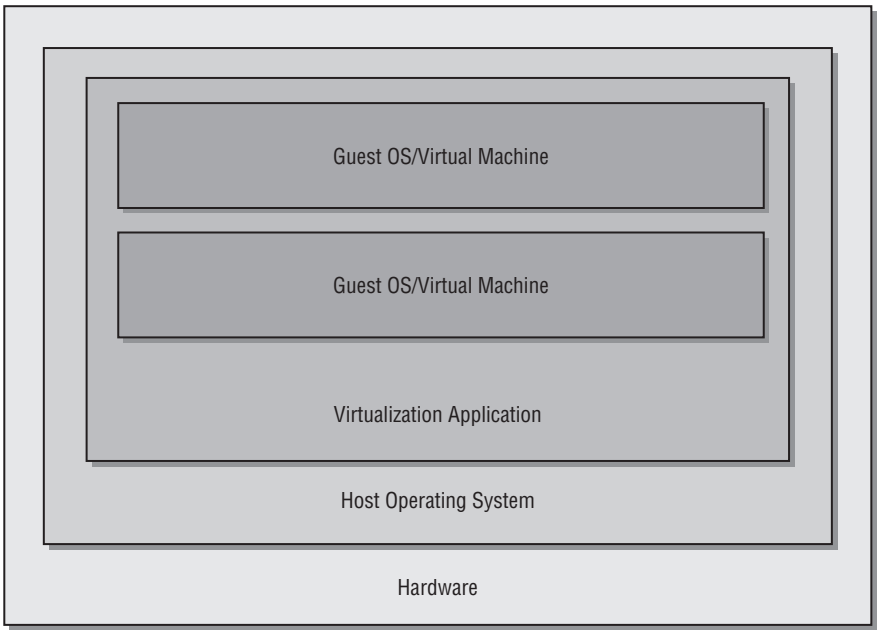

Figure 1-2

- **Hypervisor:** A hypervisor is a low-level virtual machine monitor that loads during the boot process, before the virtual machines, and runs directly on the physical hardware, as shown in Figure 1-3. The hypervisor handles requests for access to hardware resources on the physical host system, traps and handles protected or privileged instructions, and so on. Hypervisor-based virtualization runs virtual machines whose operating system, libraries, and utilities have been compiled for the same hardware and instruction set as the physical machine on which the virtual systems are running.

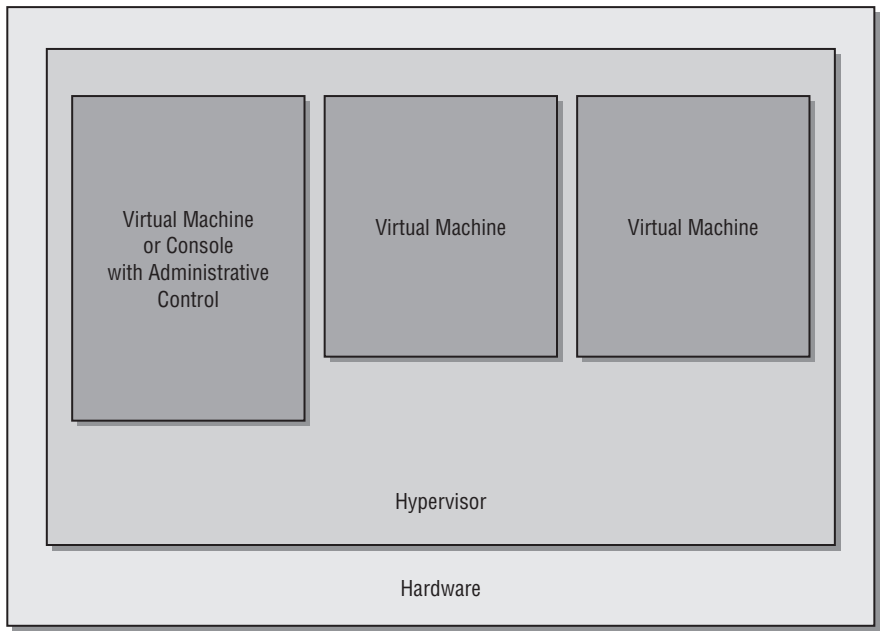

Figure 1-3

Hypervisors are used to support virtual machines in "paravirtualization," "full virtualization," and "hardware virtualization" environments. Depending on the type of hypervisor used and the specific approach to virtualization that it takes, the source code of the operating system running in a virtual machine may need to be modified to communicate with the hypervisor. Figure 1-4 shows hypervisor-based virtual machines that leverage hardware support for virtualization, but also require a hypervisor for some types of administrative interaction with the virtual machines. For more information about hypervisor-based approaches to virtualization and some examples of virtualization software that uses this approach, see the section "Server or Machine Virtualization" earlier in this chapter. For details on any of these software packages, see the section "Other Popular Virtualization and Emulation Software" in Chapter 2.

- **Kernel-level:** The Linux kernel runs the virtual machines, just like any other user-space process, as shown in Figure 1-5. This type of virtualization runs virtual machines whose operating system, libraries, and utilities have been compiled for the same hardware and instruction set as the Linux kernel that is running them, which was compiled for the physical machine on which the virtual systems are running. For more information about this approach to virtualization and some examples of virtualization software that uses this approach, see the section "Server or Machine Virtualization" earlier in this chapter. For details on any of these software packages, see the section "Other Popular Virtualization and Emulation Software" earlier in this chapter.

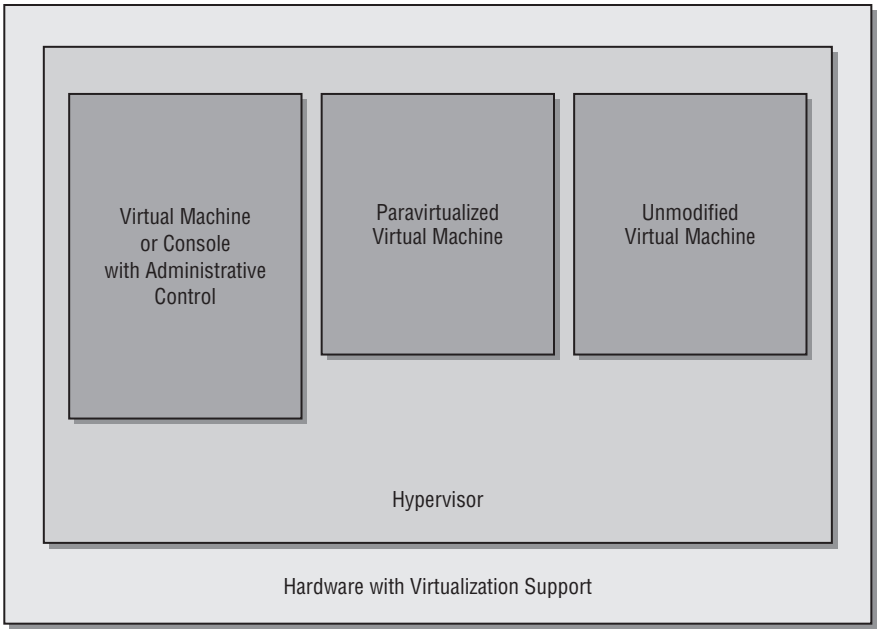

Figure 1-4

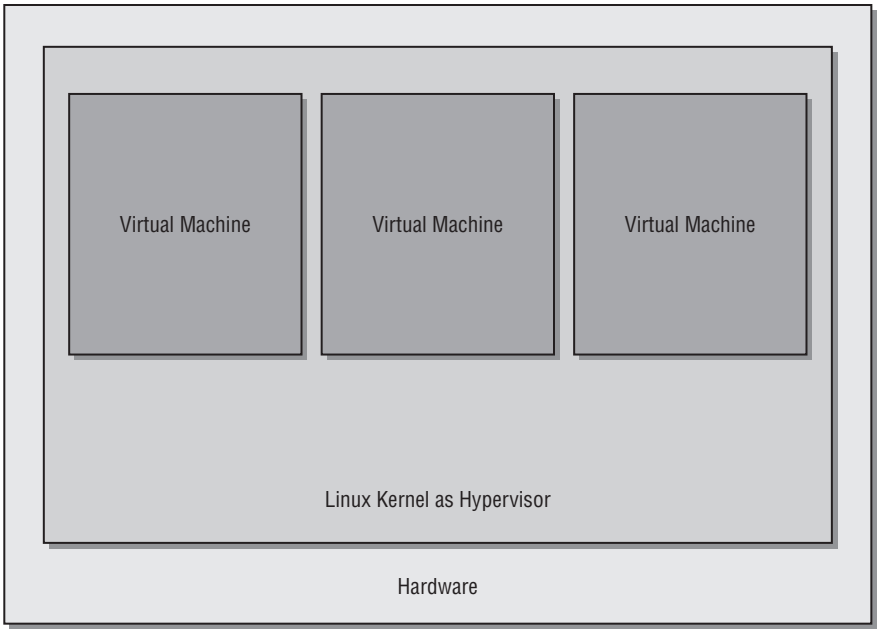

Figure 1-5

- **Emulation:** An emulator runs virtual machines by simulating a specific type of processor, its associated instruction set, and mandatory peripheral hardware, and can therefore run operating systems and associated software that have been compiled for processors and instruction sets other than the one used by the physical hardware on which it is running. The terms "emulation" and "server/machine virtualization" are easily confused because both of these enable multiple instances of different operating systems to run on a single host system. The key difference between the two is whether they execute virtual machines that are compiled for the native instruction set of the physical hardware on which the virtual machines are running, or those that have been compiled for some other processor and instruction set. The best-known emulation technology today is QEMU, which can emulate 32- and 64-bit x86, 32- and 64-bit Power PC, Motorola 68000, 32 and 64-bit SPARC, SH, MIPS, and ARM processors and run associated operating systems in those emulated environments. Microsoft's Virtual PC is actually an emulation environment because it emulates the PC instruction set and hardware, enabling it to boot and run x86 operating systems such as Linux and Microsoft Windows on both x86 and PPC Macintosh platforms. For more information about popular emulation software such as QEMU and Virtual PC, see the overviews of various packages provided in the section "Other Popular Virtualization Software" in Chapter 2.

Now that you have virtualization terminology firmly in hand and have explored some of the general reasons why it is such a hot topic today, it's time to look at some of the specific ways in which virtualization can be used to save time and money, simplify infrastructure, and so on.

Advantages of Virtualization

Virtualization can provide many operational and financial advantages as a key technology for both enterprise-computing and software-development environments. The following sections highlight these core advantages and discuss how they can save you time and money, and can help avoid or minimize many types of infrastructure, usage, and availability problems.

Better Use of Existing Hardware

Over the past few decades, processors have gone from 8 bits to 16 bits to 32 bits and now to 64 bits. Each of these increases in processor size has come with an associated increase in the amount of memory and the size of the storage that these processors can address and access. Similarly, processor speed and processor density continue to increase, where today's processors easily exceed 2 GHz and feature multiple processor cores per chip.

Sorry for the buzz kill, but much of that speed and processing power simply goes to waste for most computer systems. Heavily used Web servers, rendering systems, game machines, and the mainframes that are still searching for extraterrestrial intelligence may actually be using all of their processing power, but for most machines, all of that power is like doing your daily half-mile commute in a Lamborghini.

Enter virtualization. Running multiple virtual machines on your existing servers enables you to make good use of your spare processing power. Multiprocessor or multi-core systems can even run different virtual machines on different processors or CPU cores, taking full advantage of each portion of each processor that is available on your system. You can even get more use out of the devices, such as network interfaces, that are present on your existing servers by sharing them across your virtual machines.

Running multiple virtual servers on a single physical hardware system is generally known as "server consolidation." Historically, this meant hosting multiple server processes and their associated services on a single, physical system, increasing the importance of that system but heightening its potential to be a single point of failure for multiple services. Today, server consolidation means running multiple virtual machines on a single physical system. As you'll see throughout this book, server virtualization software such as Xen can help eliminate single points of failure in your IT infrastructure by providing portable virtual servers that can easily be moved from one physical host to another in the event of emerging problems, or which can quickly be restarted on other physical systems in the event of sudden, catastrophic failures.

Reduction in New Hardware Costs

The flip side of getting more mileage out of your existing servers is that, in many cases, you will not have to buy new physical hardware in order to deploy additional servers or services. As your business grows, deploying additional servers to better support the online capabilities that your users and customers require is a cost of being successful. Additional Web servers, new file servers for different groups or to handle increased load, new content management or intranet systems, and other similar systems are frequently added to enterprise environments as both the loads on existing systems and the number of users in general expands.

Combining server consolidation with capacity planning can reduce the number of new machines that you have to buy to support new and existing services by making better use of existing systems. In some cases, server consolidation may not eliminate the cost of new hardware, but it can simply reduce that cost. For example, buying additional memory or additional network interface cards for existing systems can enable you to expand their capabilities so that they can support additional virtual machines, without having to buy complete, new systems.

Reduction in IT Infrastructure Costs

The previous sections discussed how server consolidation can help you make the most of your existing hardware investments and reduce new hardware costs by enabling you to run multiple virtual servers on single hardware platforms. However, saving the cost of purchasing and deploying new servers isn't the only IT cost reduction associated with virtualization.

Machine rooms have a variety of per-machine infrastructure costs that you can reduce (or at least avoid increasing) by getting more mileage out of your existing hardware rather than adding new systems. Each new physical server uses a certain amount of power and places additional load on your cooling system. Virtual machines added to existing computer systems do not add to either of these loads, enabling you to add more servers with no increase in power and cooling requirements. Similarly, if you are able to consolidate multiple existing servers onto a lesser number of server systems, you can actually reduce your immediate power and cooling costs.

> *During server consolidation, you can often combine hardware from physical servers to increase the capacity of the remaining machines. For example, you can add the memory from a decommissioned system to another server that now supports multiple virtual machines. Similarly, hard drives that formerly provided local storage in decommissioned machines can be reused as spares, for backups, in RAID systems, and so on.*

Depending on how many services and server processes you must support and how successful you are in terms of server consolidation, virtualization can reduce space requirements in any systems that you host in a cage at your ISP or that you collocate. It may even provide space savings if you need to move your machine room, although building or allocating space that is tied to your current space requirements is rarely a good idea in the IT space. You should always prepare for some amount of future growth, even if virtualization enables you to minimize or optimize that growth.

In addition to power, cooling, and space savings, reducing the number of physical machines that you manage can reduce remote-access and reliability costs by requiring fewer Keyboard-Video-Mouse (KVM) systems, fewer connections to uninterruptible power supplies, and so on. Depending on how you configure networking on the physical hardware that supports your virtual machines and the number of network interface cards installed in each system, you may even be able to simplify your network cabling and reduce the number of hubs and switches that are required in the machine room.

Simplified System Administration

Using virtualization to reduce the number of physical systems that you have to manage and maintain doesn't reduce the number of systems that you are responsible for. However, it does segment the systems that you are responsible for into two groups of systems: those that are associated with specific physical resources and those that are completely virtual. The physical systems that host your virtual machines are the primary example of the first group, but this group also includes virtual machines that make specific and unique use of physical resources such as additional network cards, specific local storage devices, and so on. Running multiple virtual machines on single physical systems makes the health of those systems more critical to your business functions and introduces some new software infrastructure for virtual machine migration or cloning in the event of emerging hardware problems.

Most enterprise IT groups run some sort of centralized system status or heartbeat software, enabling you to remotely monitor the status of all of your hardware without checking each console. Depending on the capabilities of the monitoring package that you are running, you may be able to create separate sections or alert levels for systems with an explicit physical dependency on local hardware, systems with a dependency on centralized storage systems, and systems that are purely virtual. In addition, many Linux sites use the network support that is built into syslog (the system message log daemon) to consolidate system logs on specific systems in order to simplify identification of emerging hardware or software problems. Because virtual machines believe that they are running on physical hardware, you will need to group hardware-related messages from virtual machines in order to be able to identify any problems in virtual/physical machine communication. Similarly, you will want to group software logs from your virtual machines, so that you can identify emerging or immediate problems in software services such as Web servers, which may be supported on multiple machines for load balancing and redundancy reasons. For more information about consolidating system and process logs, see the sections "Centralized Logging for Virtual Machines" and "Centralized Warning Systems for Virtual Machines" in Chapter 10.

Finally, virtualization may enable you to streamline or simplify other time-consuming but standard system administration tasks, such as backups. Many virtual machines use networked storage to make themselves as independent as possible of the physical system on which they are running, as well as to improve centralization in general. The use of centralized storage such as a SAN, iSCSI, ATA-over-Ethernet, or networked filesystems can reduce the number of machines and storage systems that require physical backups. Similarly, if you choose to use thin clients or "desktop virtualization" so that your users all actually log in and work on centralized servers, you will not need to back up desktop systems that only run remote desktop software and on which no local storage is used.

Increased Uptime and Faster Failure Recovery

As mentioned in the previous section, increasing the isolation of virtual machines from specific physical hardware increases system availability by increasing the portability of those virtual machines. The portability of virtual machines enables them to be migrated from one physical server to another if hardware problems arise on the first system. Xen virtual machines can be migrated from one physical host to another without any interruption in availability — the migration process is transparent to users as well as to any processes running on those virtual machines.

Adopting virtualization and a strategy for automated problem detection and virtual machine migration can lower the costs that are traditionally associated with redundancy and failover because much of the hardware that was formerly required to ensure availability by having redundant physical systems can now be provided by being able to migrate multiple virtual machines to other, suitable hardware platforms in the event of emerging problems. You can migrate virtual systems without interrupting service, and can physically increase availability during power failures by reducing the load on your uninterruptible power supplies because they are supporting fewer physical machines, enabling you to maintain the same level of system availability for a longer period.

When partitioning and deploying software and services for high availability, one key to high availability is to efficiently divide physical and virtual machines in terms of the services that they provide. For example, in a completely virtualized environment, the primary purpose of your physical machines should be to support your virtual machines; they should not provide any external software services themselves. This enables you to respond to emerging hardware problems on your physical hardware by migrating your virtual machines to other physical hardware without having to worry about any software services that are provided by the physical machines themselves (other than support for Xen, of course). For example, you do not want to both run a Web server and support virtual machines on a physical system if you can avoid it because the failure of that physical system will make the Web server unavailable even after you have successfully migrated your virtual machines to other physical hosts. In general, you want to keep your IT infrastructure as independent as possible of the physical systems on which any portion of it is currently executing.

Simplified Capacity Expansion

Virtualization solutions such as virtual machines and storage virtualization remove the hard limits that are often imposed by physical machines or local-storage solutions. Virtual machines can be moved from one physical piece of hardware to another to enable them to benefit from hardware improvements, such as more powerful CPUs, additional CPU cores, additional memory, additional or faster network cards, and so on. Similarly, storage virtualization makes it possible to transparently increase the amount of available storage and the size of existing partitions and filesystems.

Simpler Support for Legacy Systems and Applications

Virtualization is an excellent solution to the need to run legacy software. Many businesses have certain applications that they depend on, but which may no longer be available from a specific vendor or which may not yet have been upgraded so that they can run on newer operating systems or hardware. Although depending on old software that itself depends on a specific version of an operating system is problematic from a business standpoint, it still may be a business reality.

Support for legacy software and operating environments was one of the primary motivations for virtualization when it was first introduced in an operating system by IBM in the 1960s. By running operating systems within logical partitions (known as LPARs in mainframe-speak), customers could upgrade to a newer operating system and newer, more powerful hardware without losing the ability to run the existing software and associated operating system that their businesses depended on.

Using virtualization to solve legacy software problems is a simple process. It consists of installing the appropriate legacy operating system in a virtual machine, installing the legacy software, and ensuring that the legacy software functions correctly in the new environment. Installing and using legacy software that is keyed to a traditionally unique hardware platform identifier such as the MAC address of an Ethernet card is actually simplified by virtualization software such as Xen, which enables you to set the MAC address that is associated with any virtual machine. For example, the need for occasional access to software that only runs on older Microsoft Windows operating system releases can be met quite nicely by creating a virtual machine on which the old version of Windows and your target software package is installed, and using Xen's built-in VNC support to enable remote connections to the virtual machine's desktop.

Of course, addressing legacy software issues through virtualization is only possible for legacy software that runs on the same processor architecture as the virtualization software. For example, you can't support legacy software for SPARC platforms in virtualization software for x86 platforms. In this case, you may be able to use a multi-architecture emulator such as QEMU to support the legacy operating system. Similarly, you should make sure that your virtualization solution supports the older operating systems. Xen is quite flexible in this respect, but many other virtualization solutions are not.

Simplified System-Level Development

A traditional solution to kernel and driver development testing, which often requires frequent reboots to test new kernels, is to do such development in the context of traditional Linux virtual machine solutions such as User-Mode Linux (UML). Being able to restart a virtual machine to test new kernels and drivers is much faster and less of an interruption to the development process than rebooting a physical machine. This approach can also provide significant advantages for low-level debugging if you are working on a desktop system that supports virtual machines because your development environment, development system, and the virtual machine can all coexist on one desktop platform. Virtualization solutions such as Xen provide a similarly easy-to-use development environment.

> *Hypervisor-based virtualization solutions are rarely the right environment for final testing of hardware drivers because they introduce a level of indirection that affects performance to some extent, which also masks a level of access to the bare hardware that such drivers may require. However, virtualization is a great development environment for higher-level drivers and system software such as networked filesystems. Similarly, hardware drivers should be tested against hypervisor-based virtualization solutions whenever possible to verify compatibility.*

Development and testing in virtual machines is a common use of LPARs on IBM mainframes today, where developers can work with and develop for Linux distributions running in logical partitions that are physically hosted on a mainframe.

Simplified System Installation and Deployment

The previous section discussed using virtual machines as a mechanism for testing kernel or driver development efforts. Virtualization also provides a fast, flexible, and cost-effective solution for deploying new systems, depending on the speed and memory available on your server system. Using virtual machines can simplify deploying new systems by enabling you to use a single filesystem image as the basis for all new installations. To install a new system, you can simply create a new virtual machine by cloning that filesystem and starting a new instance of a virtual machine that uses the new filesystem.

The ability to host users and customers on private virtual machines can also be used to simplify infrastructure for businesses that require large numbers of systems that are often extensively customized by their users. For example, linode.com (`www.linode.com/`) uses User-Mode Linux to provide completely customizable servers to their customers. This type of virtualization enables each user to have root access to his or her machine and complete control over the machine's execution and software environments. This is a significant step up from hosting environments that simply provide operating system–level virtual hosts to their customers. The use of full virtual machines also makes it possible to offer any virtualizable operating system to such customers, rather than having to share a kernel and thus limiting customers to various flavors of Linux, BSD, and so on.

When using full virtual machines to deploy new systems, the ability to migrate virtual machines from one host to another can also prove an asset when you're using virtual machines as a system deployment mechanism. Having a development system that is independent from a specific physical hardware platform can make life simpler and more productive for developers by making it easy to migrate those systems to be hosted in faster, more powerful machines, systems with better peripherals, and so on. Of course, whether or not migration is possible depends on the configuration and specific hardware requirements of each virtual machine, but can easily be guaranteed through clever planning and good virtual system design.

Finally, desktop virtualization simplifies deploying new systems by reducing the amount of software that needs to be installed locally. Enabling users to share a common set of software that is installed on a central server system requires careful attention to licensing issues to ensure that you do not violate the terms of each software license. These types of issues can often be solved through the use of open source software, eliminating the licensing issue, or through the use of floating licenses, which are on-demand licenses that are stored in a central pool and are temporarily assigned to users as they actually use the software.

Increasing centralization of shared resources and the standardization of deployed systems can provide significant advantages for system administrators. Regardless of whether you feel that "desktop virtualization" is a bandwagon use of the term or a true example of virtualization, deploying light-weight desktop systems and using rdesktop, Microsoft's Terminal Server, or similar packages to connect to a central server simplifies per-system software installation, reduces downtime because all desktop systems are completely interchangeable, and simplifies system administration tasks such as backups by ensuring that no important, required, or personal files are stored on local disks.

Simplified System and Application Testing

Besides server consolidation and associated savings in hardware and infrastructure costs, software system test and quality assurance environments are the biggest beneficiaries of virtualization. System-test and quality-assurance groups typically need to be able to test a specific software product on many

different operating systems or versions of an operating system. Server virtualization is extremely time- and cost-effective in such situations, reducing the amount of hardware required, and also reducing or eliminating much of the time required for system installation, reinstallation, and subsequent configuration.

Server virtualization makes it easy to install software products and test against many different operating systems or versions of operating systems without requiring dedicated hardware for each. By combining virtual machines with storage virtualization solutions such as logical volume management, you can eliminate the need to reinstall these test versions of an operating system in most cases by creating snapshots of pristine operating system distributions and then falling back to the snapshot for your next test run. Using a saved snapshot of a virtual machine is not only faster than reinstalling an entire virtual or physical system, but it can make reinstallation unnecessary if you can roll a virtual machine back to a pristine state via snapshots or the use of the non-persistent disks that are supported by some virtualization solutions.

An interesting use of virtualization in system testing outside the quality assurance or system test groups is using virtualization to test new releases of an operating system and associated software. As a specific example, newer versions of Microsoft Windows often expose or introduce incompatibilities with existing disk layouts, boot loaders, and so on. Testing a new operating system inside a virtual machine, as many of my friends have done with various releases of Microsoft Vista, provides a great sandbox in which to experiment with the new operating system, test and verify application compatibility, and so on, all without permanently disturbing the disks, partitions, and applications on their existing systems.

Virtualization Caveats

This books primarily focuses on the advantages of virtualization. As the previous section discussed, there are many good reasons for integrating virtualization into a computing environment. At the same time, virtualization is not a panacea for all IT woes — it is not appropriate for all scenarios, and it introduces real costs and concerns all its own. When considering integrating virtualization into your computing environment, you should take issues such as the ones discussed in the next few sections into account. Sayings like "An ounce of prevention is worth a pound of cure" and "forewarned is forearmed" aren't just fortune cookies to look for, they are as appropriate in IT management circles as they were when planning next season's crop rotations.

Single Point of Failure Problems

As discussed in the previous section, server consolidation leads to better use of your existing hardware by enabling you to use spare processing power to run multiple virtual machines on a single host. In a typical IT organization, each of these virtual machines runs a single server or set of related services, such as mail servers and associated anti-SPAM software, DNS servers, print servers, file servers, and so on.

The downside of server consolidation is that it increases the potential for the failure of a single physical machine, which hosts multiple virtual servers, to significantly impact your organization. If multiple servers and associated services are running on individual machines, the failure of a single machine has an impact on only a single server. When multiple servers are running as virtual machines on a single piece of hardware, the failure of that hardware can take down all of those servers.

The solution to this sort of problem is detailed planning. When designing a virtual machine–based IT infrastructure, it is very important to make sure that you plan for availability and failover whenever possible. Some common approaches are the following:

- Set up redundant hardware such as network interface cards in the host system and bond them together so that the failure of a single card is transparent to the virtual machines.
- Purchase and maintain duplicate hardware for the physical systems that host important virtual machines. The cost of having a spare power supply, motherboards, network interface cards, and so on sitting on a shelf is insignificant compared to the costs to your organization of having to wait until a vendor ships you replacement parts.
- Replicate virtual machines that host critical services across multiple physical systems so that you can survive the failure of a single physical machine and associated downtime by failing over to alternate servers.
- Run centralized system-monitoring software to alert you to emerging hardware and software problems before they become critical. This provides a window of opportunity for you to migrate existing virtual machines to other physical hosts, bring up new physical hosts if needed, and so on.

Centralized failures, such as power or network outages in your cage or machine room, will always be potential problems, and are not significantly different in virtualization environments. Your disaster recovery plans should already include information about how to deal with these types of problems through redundancy and collocation.

Server Sharing and Performance Issues

Planning for growth and increased demands is easy to overlook when creating any IT infrastructure. "Cost-conscious corporate computing" is a phrase that easily rolls off the tongue of bean counters, and it is important to make the case for capacity planning when specifying hardware requirements and designing any associated IT infrastructure. Capacity planning can be even more important when designing a virtualization-based IT infrastructure because the extent to which your virtual servers can handle increased loads and are portable across multiple physical host systems is largely dependent on how they are configured.

While approaches to server virtualization such as paravirtualization provide significant abstraction of the underlying hardware, you must be careful to design your virtual servers so that they are as independent of specific physical constraints as possible. For example, using local physical storage will always be relatively inflexible. The storage requirements of virtual servers can always provide a potential problem unless they, too, are abstracted from physical systems, either through the use of networked filesystems or through some storage virtualization technique. Similarly, applications that depend on the latest, greatest processor extensions may run slowly in a virtualization environment because they cannot get direct access to the hardware that they require.

In terms of capacity planning, the type of work that your users do on virtual servers can change the requirements for those virtual servers. For example, adding more users to existing virtual machines or performing more processor-intensive tasks on those systems can significantly increase the amount of memory that a virtual machine requires, as well as the amount of memory that the physical host system can allocate to it. Similarly, performing more data-intensive tasks on your virtual servers can change the

storage requirements of those servers, as well as the way in which they use that storage, requiring additional storage space, changes to how and where swap and paging space is allocated, and so on.

You also need to consider software licensing issues when using virtualization to host multiple user accounts on a single server or when cloning virtual machines. If you cannot afford site-wide licenses for software that your organization depends on, the use of flexible licensing schemes such as floating licenses is critical for servers that host large numbers of users. Another possible licensing problem can be caused by software vendors who do not support their software in virtualization environments — you should check your software license to be sure.

Even areas of virtualization such as desktop virtualization are not appropriate for all users. For example, remote and laptop users will need significantly more local software installed on their systems to guarantee their ability to get work done without continuous dependencies on remote access to your centralized systems.

Per-Server Network Congestion

Most full virtual machines use virtual network interfaces, subnets, and bridging packages to map those virtual interfaces (and virtual networks) to the physical hardware. If your physical host system provides only a single network interface, running multiple virtual machines that are performing network-intensive tasks can make too many demands of your physical network hardware. This can result in network performance problems for a single host or for all hosts that are sharing the same physical network interface. One obvious solution is to install multiple network interfaces in your physical host system and assign these to specific virtual machines. Unfortunately, this type of configuration can make it more complex to migrate these virtual machines from one host to another in order to work around emerging hardware problems or general performance issues.

Increase in Networking Complexity and Debugging Time

Networking on full virtual machines uses virtual network interfaces, subnets, and bridging software to make each host appear to have a unique network interface. Using full virtual machines that each have their own network interface (virtual or physical), IP address, and so on can be more complex than managing multiple physical hosts with the same characteristics. This is not only because of the extra level of software that bridging or other approaches to virtual networking require, but also because firewalls and other network control mechanisms need to be configured to allow specific traffic in specific directions and between specific virtual hosts.

Many sites set up virtual machines on their own subnets, which require separate DHCP servers in order to manage the associated ranges of IP addresses. These subnets also simplify firewalling and routing issues because the firewall or router can easily be configured to handle the blocks of IP addresses associated with those subnets differently.

Other potential issues that you must take into consideration are possible problems such as hardware MAC (Media Access Control) address collisions. In the real world, each physical Ethernet device has a unique MAC address, but most full virtual machines can be assigned specific MAC addresses. Unless you are careful when cloning and customizing your virtual machines' configuration data, it is easy to accidentally bring up multiple hosts whose network interfaces have the same MAC address. This can

cause packet routing and forwarding problems, and it is a difficult problem to identify. (A specific block of MAC addresses that should be used with Xen is discussed in Chapter 8.)

In general, the use of virtual network interfaces and virtual machines can make many network management tasks more complex unless you have spent a significant amount of time considering and working around potential problems during the planning phase of your virtual machine infrastructure.

Increased Administrative Complexity

Simplified system administration was identified earlier as a potential benefit of virtualization, but this may not always be the case if you are already using distributed system-management utilities that don't understand virtual machines. If this is the case, or if you are using multiple virtualization solutions at the same time, you should make sure that any management utilities that you already depend on (or plan to purchase) can understand virtual machines, or you'll have to restrict their use to only those systems that they can communicate with. This may not be a problem, but it's certainly a point to consider.

Identifying Candidates for Virtualization

The goal of most virtualization efforts is to consolidate multiple servers on specific physical hardware platforms. Although it is tempting to approach the question of where to start by identifying specific physical machines that you would like to eliminate, a better starting point is to consider the software that you need to support. Identifying older hardware, machines with unique or aging peripherals and storage systems, and so on is an excellent second step in your planning process, providing an excellent mechanism for cross-checking the physical hosts whose software you plan to move to virtual machines, but it is generally most effective to begin the virtualization planning process by identifying the software that you need to continue to support and its requirements.

Identifying existing software that can be moved from physical hosts to virtual machines involves a number of different factors, including hardware requirements, operating system and execution environment software requirements, the usage pattern of the software, and the load that it will place on its host system.

Collecting hardware, operating system, and execution environment data is most easily done using a checklist of some sort to ensure that you collect the same sort of data about all applications and systems that you are considering for virtualization. A spreadsheet with appropriate columns is typically the most useful framework for collecting this information, because it enables you to sort by various columns and identify software with specific physical constraints or current physical host systems that you may be able to either combine in a single virtual machine or implement in separate virtual machines that are hosted on a single piece of physical hardware that satisfies the requirements for multiple pieces of software.

The following list shows the information that you should collect for each of the applications that you are considering moving to a virtual machine. You may actually want to collect this data for all of the applications that your systems support. This will not only provide a good reference for support and licensing information, but it may also help you spot applications that are candidates for moving to virtual machines.

- **Application and version** — The name and specific version number of the application. Each version of an application that you must support should be listed separately because it may have specific operating system requirements, use specific hardware, and so on.
- **Current operating system and version** — The operating system under which the application is currently running.
- **Operating system patches or service packs** — Any specific patches that have been applied to the running operating system, including service packs for Windows-based applications.
- **Other supported operating systems** — Other operating systems and associated version numbers that this version of this application is supposed to run on.
- **Software execution environment** — Any auxiliary libraries or software packages that the software requires, including the version numbers. This includes software such as the Java virtual machine or GNU Java runtime; scripting environments such as Perl, awk, sed, and so on; interpreters; and auxiliary programs that the application can start communication with.

Trying to decipher shared and dynamically linked library requirements can be quite complex. On Linux systems, you can use the `ldd` *(list dynamic dependencies) command to identify libraries that an application requires, including the version of the system's C library that it uses (glibc, uclibc, newlib, and so on). On Microsoft Windows systems, you can use an application such as Dependency Walker, which is included with the free Process Explorer (*`www.microsoft.com/technet/sysinternals/ProcessesAndThreads/ProcessExplorer.mspx`*).*

- **Required privileges and/or users** — Any privileges required for installation and execution, including any specific user and/or group that the software must run as.
- **Associated hardware and drivers** — Any hardware that is specifically associated with this version of this application, such as a specific video card, sound card, network interface card, or storage device. If this hardware required special drivers from the hardware manufacturer, you should note their source and version numbers. If the hardware uses standard system drivers, it is still a good idea to note these to ensure that they are available for and supported on any other operating system that you may want to move the application to. You should also note any video resolution requirements.
- **Current video resolution** — The video resolution at which the application is currently running, if appropriate. You can simply mark this field as N/A for server or command-line software.
- **Memory requirements** — Any specific memory requirements that are associated with the application. This should include any limits on the maximum amount of information that the server or application can use.
- **Memory in the current host system** — The amount of memory that is available in the system on which this version of this application is currently running. You can use this information for virtual machine memory sizing purposes if the application does not have explicit memory requirements.
- **Current application performance** — Your own or your users' perception of how well the application runs and performs on its current host. Although it is not an empirical measurement, you can use this information to help assess the memory and processor requirements of any virtual machine to which you consider moving this application. This can help determine if, for example, you may want to pin a virtual machine to a specific processor to minimize administrative overhead and help guarantee the responsiveness of any applications or servers that it supports.

- **Current licensing** — Whether the application requires a license and, if so, the type of license: per-copy, floating, node-locked (locked to a specific system based on a board identifier, network IP, or MAC address), and so on.
- **Virtualization licensing** — Whether the application can be used in a virtual machine environment, which may not be the case if the software license prohibits the use of the application or the operating system that it requires in a virtual machine. This should also include information about whether the use of the application in a virtual machine requires licensing changes, the purchase of additional licenses, and so on.

As you can see, it can take some time to collect and organize all of this information, but doing so will simplify the process of identifying applications that can (or cannot) be moved to a virtual machine, and any special characteristics that the virtual machine must have in order to provide a successful and robust execution environment for your applications and servers.

Summary

This chapter provides an introduction to virtualization, discussing the many ways in which the term is used, and the different approaches to each that are available. It explains why virtualization is an increasingly popular topic today, and provides an overview of the primary advantages of virtualization. It also discusses some circumstances in which virtualization may not be appropriate.

The chapter concludes with a checklist of information that you should collect for each application, server, or service that you are using, in order to help identify candidates for virtualization and to make sure that the migration process is as smooth as possible for any applications that you are moving to virtual machines.