24 Virtualization

As enterprise data centers continue to rack up servers to slake the insatiable information appetite of the modern business, system administrators struggle with a technical conundrum: how can existing systems be managed more efficiently to save power, space, and cooling costs while continuing to meet the needs of users?

Software vendors have historically discouraged administrators from running their applications with other software, citing potential incompatibilities and in some cases even threatening to discontinue support in cases of noncompliance. The result has been a flood of single-purpose servers. Recent estimates have pegged the utilization of an average sever at somewhere between 5% and 15%, and this number continues to drop as server performance rises.

One answer to this predicament is virtualization: allowing multiple, independent operating systems to run concurrently on the same physical hardware. Administrators can treat each virtual machine as a unique server, satisfying picky vendors (in most cases) while simultaneously reducing data center costs. A wide variety of hardware platforms support virtualization, and the development of virtualization-specific CPU instructions and the increasing prevalence of multicore processors have vastly improved performance. Virtual servers are easy to install and require less maintenance (per server) than physical machines.

Implementations of virtualization have changed dramatically over the years, but the core concepts are not new to the industry. Big Blue used virtual machines in early mainframes while researching time-sharing concepts in the 1960s, allowing users to share processing and storage resources through an abstraction layer. The same techniques developed by IBM were used throughout the mainframe heyday of the 1970s until the client-server boom of the 1980s. The technology lay dormant during the 1980s and 1990s until the cost and manageability problems of enormous server farms rekindled interest in virtualization for modern systems. VMware is widely credited with having started the current virtualization craze by creating a virtualization platform for the Intel x86 architecture in 1999.

Today, virtualization technology is a flourishing business, with many vendors twisting knobs and pushing buttons to create unique entries into the market. VMware remains a clear leader and offers products targeted at business of all sizes, along with management software to support highly virtualized organizations. The open source community has responded with a project known as Xen, which is supported commercially by a company called XenSource, now owned by Citrix. With the release of Solaris 10, Sun introduced some powerful technology known collectively as zones and containers that can run more than 8,000 virtual systems on a single Solaris deployment. These are just a few of the players in the market. There are dozens of competing products, each with a slightly different niche.

See page 206 for more information about storage area networks.

Although server virtualization is our primary focus in this chapter, the same concepts apply to many other areas of the IT infrastructure, including networks, storage, applications, and even desktops. For example, when storage area networks or network-attached storage are used, pools of disk space can be provisioned as a service, creating additional space on demand. Applying virtualization to the desktop can be useful for system administrators and users alike, allowing for custom-tailored application environments for each user.

The many virtualization options have created a struggle for hapless UNIX and Linux administrators. With dozens of platforms and configurations to choose from, identifying the right long-term approach can be a daunting prospect. In this chapter, we start by defining the terms used for virtualization technologies, continue with a discussion of the benefits of virtualization, proceed with tips for selecting the best solution for your needs, and finally, work through some hands-on implementation activities for some of the most commonly used virtualization software on our example operating systems.

24.1 VIRTUAL VERNACULAR

The virtualization market has its own set of confusing terms and concepts. Mastering the lingo is the first step toward sorting out the various options.

Operating systems assume they are in control of the system's hardware, so running two systems simultaneously causes resource conflicts. Server virtualization is

an abstraction of computing resources that lets operating systems run without direct knowledge of the underlying physical hardware. The virtualization software parcels out the physical resources such as storage, memory, and CPU, dynamically allocating their use among several virtual machines.

UNIX administrators should understand three distinct paradigms: full virtualization, paravirtualization, and OS-level virtualization. Each model resolves the resource contention and hardware access issues in a slightly different manner, and each model has distinct benefits and drawbacks.

Full virtualization

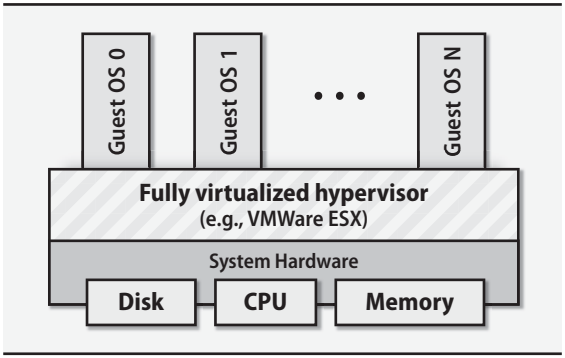
Full virtualization is currently the most accepted paradigm in production use today. Under this model, the operating system is unaware that it is running on a virtualized platform. A "hypervisor," also known as a virtual machine monitor, is installed between the virtual machines ("guests") and the hardware.

Such hypervisors are also known as bare-metal hypervisors since they control the physical hardware. The hypervisor provides an emulation layer for all of the host's hardware devices. The guest operating system is not modified. Guests make direct requests to the virtualized hardware, and any privileged instructions that guest kernels attempt to run are intercepted by the hypervisor for appropriate handling.

Bare-metal virtualization is the most secure type of virtualization because guest operating systems are isolated from the underlying hardware. In addition, no kernel modifications are required, and guests are portable among differing underlying architectures. As long as the virtualization software is present, the guest can run on any processor architecture. (Translation of CPU instructions does, however, incur a modest performance penalty.)

VMware ESX is an example of a popular full virtualization technology. The general structure of these systems is depicted in Exhibit A.

Exhibit A Full virtualization architecture



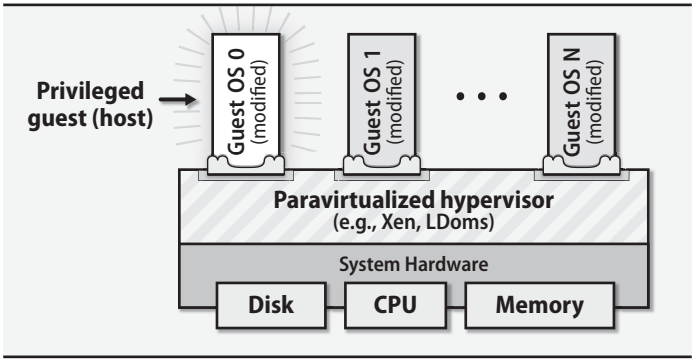
Paravirtualization

Paravirtualization is the technology used by Xen, the leading open source virtual platform. Like full virtualization, paravirtualization allows multiple operating systems to run in concert on one machine. However, each OS kernel must be modified to support "hypercalls," or translations of certain sensitive CPU instructions. User-space applications do not require modification and run natively on Xen machines. A hypervisor is used in paravirtualization just as in full virtualization.

The translation layer of a paravirtualized system has less overhead than that of a fully virtualized system, so paravirtualization does lead to nominal performance gains. However, the need to modify the guest operating system is a dramatic downside and is the primary reason why Xen paravirtualization has scant support outside of Linux and other open source kernels.

Exhibit B shows a paravirtualized environment. It looks similar to the fully virtualized system in Exhibit A, but the guest operating systems interface with the hypervisor through a defined interface, and the first guest is privileged.

Exhibit B Paravirtualization architecture



Operating system virtualization

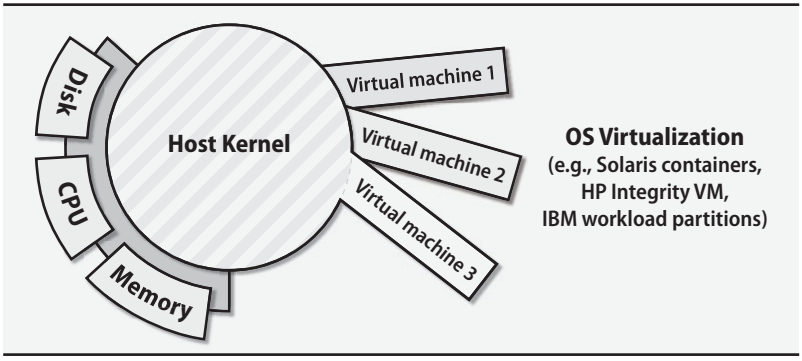
OS-level virtualization systems are very different from the previous two models. Instead of creating multiple virtual machine environments within a physical system, OS-level virtualization lets an operating system create multiple, isolated application environments that reference the same kernel. OS-level virtualization is properly thought of as a feature of the kernel rather than as a separate layer of software abstraction.

Because no true translation or virtualization layer exists, the overhead of OS-level virtualization is very low. Most implementations offer near-native performance. Unfortunately, this type of virtualization precludes the use of multiple operating systems since a single kernel is shared by all guests (or "containers" as they are

commonly known in this context).[1] AIX workload partitions and Solaris containers and zones are examples of OS-level virtualization.

OS-level virtualization is illustrated in Exhibit C.

Exhibit C OS-level virtualization architecture



Native virtualization

In an attempt to distinguish their hardware offerings, the silicon heavyweights AMD and Intel are competing head to head to best support virtualization through hardware-assisted ("native") virtualization. Both companies offer CPUs that include virtualization instructions, eliminating the need for the translation layer used in full and paravirtualization. Today, all major virtualization players can take advantage of these processors' features.

Cloud computing

In addition to traditional virtualization, a relatively recent offering in the industry known informally (and, to some, begrudgingly) as cloud computing is an alternative to locally run server farms. Cloud computing offers computing power as a service, typically attractively priced on an hourly basis. The most obvious benefit is the conversion of server resources into a form of infrastructure analogous to power or plumbing. Administrators and developers never see the actual hardware they are using and need have no knowledge of its structure. The name comes from the traditional use of a cloud outline to denote the Internet in network diagrams.

As a system administration book, this one focuses on cloud computing at the server level, but applications are also being moved to the cloud (commonly known as software-as-a-service, or SAAS). Everything from email to business productivity suites to entire desktop environments can be outsourced and managed independently.

1. This is not entirely true. Solaris containers have a feature called "branded zones" that allows Linux binaries to run on a Solaris kernel.

Cloud services are commonly bundled with a control interface that adjusts capacity on demand and allows one-click provisioning of new systems. Amazon's Elastic Compute Cloud (EC2) is the most mature of the first-generation services of this type. It has been widely adopted by companies that offer next-generation web platforms. Love it or hate it, utility computing is gaining traction with bean counters as a cheaper alternative to data centers and localized server infrastructure. Talking heads in the IT industry believe that cloud technologies in their myriad forms are the future of computing.

Cloud computing relies on some of the same ideas as virtualization, but it should be considered a distinct set of technologies in its own right.

Live migration

A final concept to consider is the possibility of migrating virtual machines from one physical machine to another. Most virtualization software lets you move virtual machines in real time between running systems, in some cases without interruptions in service or loss of connectivity. This feature is called live migration. It's helpful for load balancing, disaster recovery, server maintenance, and general system flexibility.

Comparison of virtualization technologies

Although the various virtualization options are conceptually different, each technique offers similar results in the end. Administrators access virtual systems in the same way as they access any normal node on the network. The primary differences are that hardware problems may affect multiple systems at once (since they share hardware) and that resource contention issues must be debugged at the same level at which virtualization is implemented (e.g., in the hypervisor).

24.2 Benefits of virtualization

Given the many blessings of virtual computing, it's surprising that it took so many years to be developed and commercially accepted. Cost savings, reduced energy use, simplified business continuity, and greater technical agility are some of the main drivers of the adoption of virtual technologies.

Cost is a major factor in all new IT projects, and with virtualization, businesses realize immediate short-term cost savings because they purchase fewer servers. Instead of acquiring new servers for a new production application, administrators can spin up new virtual machines and save in up-front purchasing costs as well as ongoing support and maintenance fees. Cooling requirements are cut dramatically since virtual servers do not generate heat, resulting in additional savings. Data centers also become easier to support and less expensive to maintain. With some organizations consolidating up to 30 physical servers onto a single virtual host, a quick glance at the savings in rack space alone is sure to set data center managers blushing with pride.

A reduced ecological impact is an easy marketing win for businesses as well. Some estimates suggest that nearly one percent of the world's electricity is consumed by power-hungry data centers.[2] Modern multicore CPUs are used more efficiently when several virtual machines are running simultaneously.

Business continuity—that is, the ability of a company to survive physical and logical crises with minimal impact on business operations—is a vexing and expensive problem for system administrators. Complex approaches to disaster recovery are simplified when virtual servers can be migrated from one physical location to another with a single command. The migration technologies supported by most virtualization platforms allow applications to be location independent.

Because hypervisors can be accessed independently of the virtual servers they support, server management ceases to be grounded in physical reality and becomes fully scriptable. System administrators can respond quickly to customer requests for new systems and applications by making use of template-driven server provisioning. Scripts can automate and simplify common virtual system administration tasks. A virtual server's boot, shutdown, and migration chores can be automated by shell scripts and even scheduled through **cron**. Discontinued operating systems and applications can be moved off unsupported legacy hardware onto modern architectures.

Virtualization increases availability. Live migration allows physical servers to be taken down for maintenance without downtime or interruptions in service. Hardware upgrades do not impact the business, either. When it's time to replace an aging machine, the virtual system is immediately portable without a painful upgrade, installation, test, and cutover cycle.

Virtualization makes the rigorous separation of development, test, staging, and production environments a realistic prospect, even for smaller businesses. Historically, maintaining these separate environments has been too expensive for many businesses to bear, even though regulations and standards may have demanded it. The individual environments may also benefit; for example, quality assurance testers can easily restore a test environment to its baseline configuration.

In terms of immediate gratification, few technologies seem to offer as many possibilities as server virtualization. As we'll see in the next section, however, virtualization is not a panacea.

24.3 A PRACTICAL APPROACH

The transition to a virtualized environment must be carefully planned, managed, and implemented. An uncoordinated approach will lead to a motley assortment of unstable, unmanageable implementations that do more harm than good. Furthermore, the confidence of stakeholders is easily lost: early missteps can complicate

2. Estimated by Jonathan Koomey in his excellent study "Estimating total power consumption by servers in the U.S. and the world."

future attempts to move reluctant users to new platforms. Slow and steady wins the race.

It's important to choose the right systems to migrate since some applications are better suited to virtualization than others. Services that already have high utilization might be better left on a physical system, at least at the outset. Other services that are best left alone include these:

- Resource intensive backup servers or log hosts
- High-bandwidth applications, such as intrusion detection systems
- Busy I/O-bound database servers
- Proprietary applications with hardware-based copy protection
- Applications with specialized hardware needs, such as medical systems or certain scientific data gathering applications

Good candidates for virtualization include these:

- Internet-facing web servers that query middleware systems or databases
- Underused stand-alone application servers
- Developer systems, such as build or version control servers
- Quality assurance test hosts and staging environments
- Core infrastructure systems, such as LDAP directories, DHCP and DNS servers, time servers, and SSH gateways

Starting with a small number of less critical systems will help establish the organization's confidence and develop the expertise of administrators. New applications are obvious targets since they can be built for virtualization from the ground up. As the environment stabilizes, you can continue to migrate systems at regular intervals. Large organizations might find that 25 to 50 servers per year is a sustainable pace.

Plan for appropriate infrastructure support in the new environment. Storage and network resources should support the migrations plans. If several systems on the same physical host will reside on separate physical networks, plan to trunk the network interfaces. Include appropriate attachments for systems that will use space on a SAN. Make smart decisions about locating similar systems on the same physical hardware to simplify the infrastructure. Finally, make sure that every virtual machine has a secondary home to which it can migrate in the event of maintenance or hardware problems on the primary system.

Don't run all your mission-critical services on the same physical hardware, and don't overload systems with too many virtual machines.

Thanks to rapid improvements in server hardware, administrators have lots of good options for virtualization. Multicore, multiprocessor architectures are an obvious choice for virtual machines since they reduce the need for context switches and facilitate the allocation of CPU resources. New blade server products from major manufacturers are designed for virtual environments and offer high I/O

and memory capacity. Solid state disk drives have inherent synergy with virtualization because of their fast access times and low power consumption.

24.4 VIRTUALIZATION WITH LINUX

Two major projects are vying for the title of Linux virtualization champion: Xen and KVM. In one corner, Xen is an established, well-documented platform with wide support from the distribution heavyweights. In the other corner, KVM has been accepted by Linus Torvalds into the mainstream Linux kernel. It enjoys a growing fan base, and both Ubuntu and Red Hat are supporting it.

In this section we'll stay out of the ring and stay focused on the pertinent system administration details for each technology.

Introduction to Xen

Initially developed by Ian Pratt as a research project at the University of Cambridge, the Linux-friendly Xen has grown to become a formidable virtualization platform, challenging even the commercial giants in terms of performance, security, and especially cost. As a paravirtual hypervisor, the Xen virtual machine monitor claims a mere 0.1%–3.5% overhead, far less than fully virtualized solutions. Because the Xen hypervisor is open source, a number of management tools exist with varying levels of feature support. The Xen source is available from xen.org, but many distributions already include native support.

Xen is a bare-metal hypervisor that runs directly on the physical hardware. A running virtual machine is called a domain. There is always at least one domain, referred to as domain zero (or dom0). Domain zero has full hardware access, manages the other domains, and runs all device drivers. Unprivileged domains are referred to as domU. All domains, including dom0, are controlled by the Xen hypervisor, which is responsible for CPU scheduling and memory management. A suite of daemons, tools, and libraries completes the Xen architecture and enables communication between domU, dom0, and the hypervisor.

Several management tools simplify common Xen administration tasks such as booting and shutting down, configuring, and creating guests. Xen Tools is a collection of Perl scripts that simplify domU creation. MLN, or Manage Large Networks, is another Perl script that creates complex virtual networks out of clean, easily understood configuration files. ConVirt is a shockingly advanced GUI tool for managing guests. It includes drag-and-drop live migration, agentless multiserver support, availability and configuration dashboards, and template-driven provisioning for new virtual machines. For hardened command-line junkies, the unapologetic built-in tool **xm** fits the bill.

Linux distributions vary in their support of Xen. Red Hat originally expended significant resources on including Xen in its distributions before ditching it for the competing KVM software. Xen is well supported in SUSE Linux, particularly in the Enterprise 11 release. Canonical, the company behind Ubuntu Linux, has

taken an odd approach with Xen, wavering on support in most releases before finally dropping it in version 8.10 in favor of KVM (although Xen is still mentioned in documentation). Once installed, basic Xen usage differs little among distributions. In general, we recommend Red Hat or SUSE for a large Xen-based virtualization deployment.

Xen essentials

A Linux Xen server requires a number of daemons, scripts, configuration files, and tools. Table 24.1 lists the most interesting puzzle pieces.

Table 24.1 Xen components

Path	Purpose
/etc/xen	Primary configuration directory
xend-config.sxp	Top-level **xend** configuration file
auto	Guest OS config files to autostart at boot time
scripts	Utility scripts that create network interfaces, etc.
/var/log/xen	Xen log files
/usr/sbin/xend	Master Xen controller daemon
/usr/sbin/xm	Xen guest domain management tool

Each Xen guest domain configuration file in **/etc/xen** specifies the virtual resources available to a domU, such as disk devices, CPU, memory, and network interfaces. There is one configuration file per domU. The format is extremely flexible and gives administrators granular control over the constraints that will be applied to each guest. If a symbolic link to a domU configuration file is added to the **auto** subdirectory, that guest OS will be automatically started at boot time.

The **xend** daemon handles domU creation, migration, and other management tasks. It must always remain running and typically starts at boot time. Its configuration file, **/etc/xen/xend-config.sxp**, specifies the communication settings for the hypervisor and the resource constraints for dom0. It also configures facilities for live migration.

See the footnote on page 308 for more info about sparse files.

Guest domains' disks are normally stored in virtual block devices (VBDs) in dom0. The VBD can be connected to a dedicated resource such as a physical disk drive or logical volume. Or it can be a loopback file, also known as a file-backed VBD, created with **dd**. Performance is better with a dedicated disk or volume, but files are more flexible and can be managed with normal Linux commands (such as **mv** and **cp**) in domain zero. Backing files are sparse files that grow as needed. Unless the system is experiencing performance bottlenecks, a file-backed VBD is usually the better choice. It's a simple process to transfer a VBD onto a dedicated disk if you change your mind.

Similarly, virtual network interfaces (aka VIFs) can be set up in multiple ways. The default is to use bridged mode, in which each guest domain is a node on the same network as the host. Routed and NAT modes configure guest domains to be on a private network, accessible to each other and domain 0 but hidden from the rest of the network. Advanced configurations include bonded network interfaces and VLANs for guests on different networks. If none of these options fit the bill, Xen network scripts are customizable to meet almost any unique need.

Xen guest installation with virt-install

One tool for simple guest installation is **virt-install**, bundled as part of Red Hat's **virt-manager** application.[3] **virt-install** is a command-line OS provisioning tool. It accepts installation media from a variety of sources, such as an NFS mount, a physical CD or DVD, or an HTTP location.

For example, the installation of a guest domain might look like this:

```
redhat$ sudo virt-install -n chef -f /vm/chef.img -l http://example.com/myos
    -r 512 --nographics
```

This is a typical Xen guest domain with the name "chef," a disk VBD location of **/vm/chef.img**, and installation media obtained through HTTP. The instance has 512MiB of RAM and uses no X Windows-based graphics support during installation. **virt-install** downloads the files needed to start the installation and then kicks off the installer process.

You'll see the screen clear, and you'll go through a standard text-based Linux installation, including network configuration and package selection. After the installation completes, the guest domain reboots and is ready for use. To disconnect from the guest console and return to dom0, type <Control-]>.

See page 1138 for more details on VNC.

It's worth noting that although this incantation of **virt-install** provides a text-based installation, graphical support through Virtual Network Computing (VNC) is also available.

The domain's configuration is stored in **/etc/xen/chef**. Here's what it looks like:

```
name = "chef"
uuid = "a85e20f4-d11b-d4f7-1429-7339b1d0d051"
maxmem = 512
memory = 512
vcpus = 1
bootloader = "/usr/bin/pygrub"
on_poweroff = "destroy"
on_reboot = "restart"
on_crash = "restart"
vfb = [  ]
disk = [ "tap:aio:/vm/chef.dsk,xvda,w" ]
vif = [ "mac=00:16:3e:1e:57:79,bridge=xenbr0" ]
```

3. Install the **python-virtinst** package for **virt-install** support on Ubuntu.

You can see that the NIC defaults to bridged mode. In this case, the VBD is a "block tap" file that provides better performance than does a standard loopback file. The writable disk image file is presented to the guest as **/dev/xvda**. This particular disk device definition, tap:aio, is recommended by the Xen team for performance reasons.

The **xm** tool is convenient for day-to-day management of virtual machines, such as starting and stopping VMs, connecting to their consoles, and investigating current state. Below, we show the running guest domains, then connect to the console for chef. IDs are assigned in increasing order as guest domains are created, and they are reset when the host reboots.

```
redhat$ sudo xm list
Name                        ID  Mem(MiB)   VCPUs  State  Time(s)
Domain-0                     0      2502       2  r-----    397.2
chef                        19       512       1  -b----     12.8
redhat$ sudo xm console 19
```

To effect any customization of a guest domain, such as attaching another disk or changing the network to NAT mode instead of bridged, you should edit the guest's configuration file in **/etc/xen** and reboot the guest. The **xmdomain.cfg** man page contains excellent detail on additional options for guest domains.

Xen live migration

A domain migration is the process of moving a domU from one physical host to another, and a live migration does so without any loss of service. Practically speaking, this is one of the handiest and most magical of virtualization tricks for system administrators. Open network connections are maintained, so any SSH sessions or active HTTP connections will not be lost. Hardware maintenance, operating system upgrades, and physical server reboots are all good opportunities to use migration magic.

One important requirement for implementing migrations is that storage must be shared. Any storage needed by the domU, such as the disk image files on which the virtual machine is kept, must be accessible to both host servers. File-backed virtual machines are simplest for live migration since they're usually contained in a single portable file. But a SAN, NAS, NFS share, or iSCSI unit are all acceptable methods of sharing files among systems. However the VBD is shared, be sure to run the domU on only one physical server at a time. Linux filesystems do not support direct, concurrent access by multiple hosts.

Additionally, because the IP and MAC addresses of a virtual machine follow it from one host to another, each server must be on the same layer 2 and IP subnets. Network hardware learns the new location of the MAC address once the virtual machine begins sending traffic over the network.

Once these basic requirements are met, all you need are a few configuration changes to the hypervisor configuration file, **/etc/xen/xend-config.sxp**, to enable

migrations. Table 24.2 describes the pertinent options; they are all commented out in a default Xen installation. After making changes, restart **xend** by running **/etc/init.d/xend restart**.

Table 24.2 Live migration options in the xend configuration file

Option	Description
xend-relocation-server	Enables migration; set to yes
xend-relocation-port	Network port used for migration activities
xend-relocation-address	Interface to listen on for migration connections. If unspecified, Xen listens on all interfaces in dom0.
xend-relocation-hosts-allow	Hosts from which to allow connections[a]

a. This should never be blank; otherwise, connections will be allowed from all hosts.

In the process of migrating a virtual machine between hosts, the domU's memory image traverses the network in an unencrypted format. Administrators should keep security in mind if the guest has sensitive data in memory.

Before attempting a migration, the guest's configuration file must be in place on both the source and destination servers. If the location of the disk image files differs between hosts (e.g., if one server mounts the shared storage in **/xen** and the other in **/vm**), this difference should be reflected in the disk = parameter of the domain's configuration file.

The migration itself is simple:

```
redhat$ sudo xm migrate --live chef server2.example.com
```

Assuming that our guest domain chef is running, the command migrates it to another Xen host, **server2.example.com**. Omitting the **--live** flag pauses the domain prior to migration. We find it entertaining to run a **ping** against chef's IP address during the migration to watch for dropped packets.

KVM

KVM, the Kernel-based Virtual Machine, is a full virtualization tool that has been included in the mainline Linux kernel since version 2.6.20. It depends on the Intel VT and AMD-V virtualization extensions found on current CPUs.[4] It is the default virtualization technology in Ubuntu, and Red Hat has also changed gears from Xen to KVM after acquiring KVM's parent company, Qumranet.

Since KVM virtualization is supported by the CPU hardware, many guest operating systems are supported, including Windows. The software also depends on a modified version of the QEMU processor emulator.

4. Does your CPU have them? Try **egrep '(vmx|svm)' /proc/cpuinfo** to find out. If the command displays no output, the extensions are not present. On some systems, the extensions must be enabled in the system BIOS before they become visible.

Under KVM, the Linux kernel itself serves as the hypervisor; memory management and scheduling are handled through the host's kernel, and guest machines are normal Linux processes. Enormous benefits accompany this unique approach to virtualization. For example, the complexity introduced by multicore processors is handled by the kernel, and no hypervisor changes are required to support them. Linux commands such as **top**, **ps**, and **kill** show and control virtual machines, just as they would for other processes. The integration with Linux is seamless.

Administrators should be cautioned that KVM is a relatively young technology, and it should be heavily tested before being promoted to production use. The KVM site itself documents numerous incompatibilities when running guests of differing operating system flavors. Reports of live migrations breaking between different versions of KVM are common. Consider yourself forewarned.

KVM installation and usage

Although the technologies behind Xen and KVM are fundamentally different, the tools that install and manage guests operating systems are similar. As under Xen, you can use **virt-install** to create new KVM guests. Use the **virsh** command to manage them.[5] These utilities depend on Red Hat's **libvirt** library.

Before the installation is started, the host must be configured to support networking in the guests.[6] In most configurations, one physical interface is used to bridge network connectivity to each of the guests. Under Red Hat, the network device configuration files are in **/etc/sysconfig/network-scripts**. Two device files are required: one each for the bridge and the physical device.

In the examples below, **peth0** is the physical device and **eth0** is the bridge:

/etc/sysconfig/network-scripts/peth0

```
DEVICE=peth0
ONBOOT=yes
BRIDGE=eth0
HWADDR=XX:XX:XX:XX:XX:XX
```

/etc/sysconfig/network-scripts/eth0

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
TYPE=Bridge
```

Here, the eth0 device receives an IP address through DHCP.

The flags passed to **virt-install** vary slightly from those used for a Xen installation. To begin with, the **--hvm** flag indicates that the guest should be hardware virtualized, as opposed to paravirtualized. In addition, the **--connect** argument guarantees that the correct default hypervisor is chosen, since **virt-install** sup-

5. You can use **virsh** to manage Xen domUs as well, if you wish.
6. This is equally true with Xen, but **xend** does the heavy lifting, creating interfaces in the background.