2

THE VMWARE VIRTUAL MACHINE

VMware Workstation provides virtual hardware on top of a host computer's real processor and memory. That hardware is the focus of this chapter. First, we'll take a short look at the hardware layout on a PC and show where the line between virtual and real is on VMware. An overview of each virtual device follows. To provide a base reference, a description of the real hardware accompanies each explanation of a virtual device.

The last two sections in this chapter are about the PC BIOS and boot process. You'll need to know the information there if you plan to use your disks in raw (direct) mode.

2.1 Processor, Bus, Memory, and Interrupts

A PC has three essential components: the CPU, memory, and I/O devices (which we'll just call devices from now on). CPU stands for *central processing unit* (or *processor*), and it is the heart of a computer. It not only runs the instructions that comprise a program, but it also supervises the rest of the system.

Because a CPU can't store all of the essential system data in its limited amount of storage space, the processor needs external memory, in the form of *random-access memory* (RAM). Because this storage is very important to the CPU's operation, the CPU has nearly direct access to it. Between the CPU and its RAM is an important layer called the *memory management unit* (MMU). This piece splits the RAM into many small pieces known as *pages*. The MMU has a map (called a *page table*) of the CPU-addressed memory to pages in real memory.

The memory that user programs see is called *virtual memory* (not a VMware term). When a CPU instruction asks for a certain address in memory, the MMU redirects the access through its page table to a place in real memory. (Note that this system is not hardware only; the operating system works with the MMU to get the job done. Some older systems, such as DOS, do not use the MMU.)

On the other hand, it's rare for a device to have a close connection to the CPU for two reasons. One is that devices vary so much that it would be almost impossible to create an interface inside the CPU supporting each device. Furthermore, devices often transmit data at irregular intervals, and several devices may try to talk to the CPU at once. It doesn't make sense for the CPU to bother with a device until all of its data is in order and in a format that the CPU understands, so the CPU has a common *bus* that connects the devices to the CPU. In addition, there are often extra controllers and interfaces between the devices and the bus; an example of this is the SCSI controller.

VMware has two virtual buses, resembling those in most modern PCs: PCI and an extension off the PCI bus called the *PCI-to-ISA bridge*. ISA is an old standard dating back to the first IBM PC. Though it is remarkably inferior to current technology, backward compatibility keeps ISA alive.

Although certain devices, such as serial ports, still use the ISA interface, hardware developers avoid it whenever possible because of its limitations. Some of those limitations have to do with *interrupts* and *I/O ports*. Devices send interrupts over the bus to the CPU when they're ready to transmit data or when they have something otherwise significant to say. The CPU notices this, stops its current activity, and processes the interrupt. ISA numbers its interrupts into *interrupt request* (IRQ) *levels* to differentiate them. Unfortunately, there aren't many IRQs, and if two devices try to share one number, serious conflicts result. Similarly, devices use I/O ports to communicate through the bus to the processor.

While ISA had fixed port numbers for every device, and you had to be very careful to avoid conflicts, PCI has neither of these problems. PCI interrupts have much more information, so several PCI devices can share an IRQ, and the chipset also automatically sets up the I/O ports when you turn the computer on.

Figure 2.1 shows an abstract map of the VMware Workstation PC. Notice the curious arrangement of the IDE disk interface (you can access it through the PCI or ISA bus).

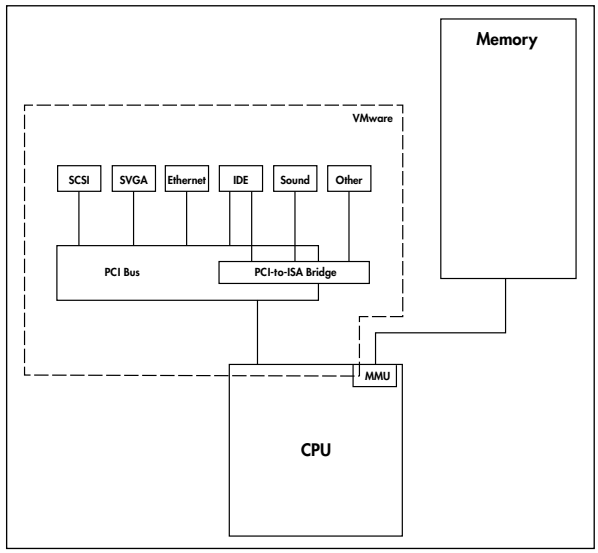

Figure 2.1: VMware Workstation virtual and real hardware—the virtual hardware is within the dashed box

2.2 The VMware Devices

Let's have a look at VMware's virtual devices. Each device discussed here corresponds to a real device on a PC. To ensure that you're up to speed, we'll first take a brief look at each real device before talking about VMware's virtual device.

2.2.1 IDE Disks and CD-ROM Drives

The most common type of disk in PCs is the Integrated Device Electronics, or IDE, disk. Traditional disk architectures had a disk *controller* somewhere on the main CPU board, with a dedicated data channel between the controller and any disks. However, these controllers tended to add cost to the machine. Also, they often did far more than a PC needed—a SCSI-I or II controller, for example, can handle up to seven disks or other devices. Because a PC rarely has more than one or two hard drives, and other components are usually in the form of Industry-Standard Architecture, or ISA, cards, PC designers decided that they might be able to reduce costs by removing this flexibility. They came up with IDE disks, where the controller is actually integrated into the disk itself, leaving only an ISA *interface* card on the PC's motherboard.

As PCs became popular, IDE disks quickly became the most common commodity disks, and prices dropped far below those of other disk types. However, the shortcuts taken in the design of IDE caused problems later, and every time a new difficulty arose, someone added a new "standard" to the specification. For example, the IDE interface didn't support CD-ROM drives, so hardware engineers squeezed ATAPI into the design. (ATAPI, or AT Attached Packet Interface, connects CD-ROM and tape drives to the IDE interface.) In modern PCs, it isn't even strictly true that the disks sit behind an interface anymore—it's hard to draw a clear line between the controller and the interface.

Like most PC motherboards, VMware Workstation has two IDE controllers, called the *primary* and *secondary* interfaces. The virtual hardware is the Intel 82371AB PIIX4 chipset. The IRQ (or interrupt request) and port assignments are as follows:

Interface	IRQ	I/O Ports
Primary	14	0x01f0 to 0x01f7, 0x03f6
Secondary	15	0x0170 to 0x0177, 0x0376

Like all IDE interfaces, each of these ports can support a maximum of two devices (sometimes called *master* and *slave* devices). Under VMware, they are limited to disk, CD-ROM/DVD-ROM, CD-R, and CD-RW drives that you can configure in a few different ways.

VMware stores *virtual* and *plain disks* as files on your host system. These files are images of a complete disk and are usually quite large. A *raw disk* is a mapping from one of your real disks to a disk under VMware.

VMware disks provide some added flexibility that you don't have with disks on a real machine, such as the ability to undo or make a disk appear as read-write without actually altering the data underneath. You may also configure VMware ATAPI CD-ROM drives as mappings to a real disk, or if you have a CD-ROM image on your host system (such as an ISO9660 file), you can point VMware to that. (See page 58 for more information on all of these modes and section 11.4 for a discussion of CD-ROM images.)

2.2.2 SCSI Disks

The small computer system interface, or SCSI, is a peripheral bus standard for small computers, meant for attaching various devices to a computer. The bus is independent of the computer's architecture, so you can, for example, exchange SCSI devices between PC and Sun systems, assuming that you have the drivers for those particular kinds of devices on those systems. The gateway between the SCSI bus and a computer is called a *SCSI host controller*, though it's often known as a SCSI adapter or SCSI card.

There are several types of SCSI buses: SCSI-I, SCSI-II, Fast SCSI-II, Ultra Wide SCSI, and so on. SCSI-I and II can have up to eight devices on a bus (this number includes the host controller). Wide SCSI can handle up to 16 devices. If you're interested in SCSI, a good place to look is *The Book of SCSI, 2nd Edition* (another No Starch Press publication; ISBN 1-886411-10-7).

VMware Workstation has virtual SCSI hardware. Its host controller is a PCI-based BusLogic BT-958 Ultra Wide SCSI adapter. You can configure up to seven disks or CD-ROM drives from VMware's controller, even though it shows up as wide (on real wide SCSI buses, you can have up to 15 devices). The special VMware SCSI disk modes and features are the same as for the IDE disks mentioned in the previous section. Because the virtual SCSI controller is a PCI device, the system dynamically configures the IRQ number and I/O ports when it starts.

In addition to SCSI disks, VMware Workstation 3 supports arbitrary virtual SCSI devices such as CD-R writers and SCSI scanners.

2.2.3 Floppy Drives

Probably the most primitive device that VMware supports is the PC floppy disk drive, and there isn't much hardware less intelligent than it. While floppy drives on other platforms have features such as automatic media detection and eject notification, everything is manual on a PC floppy drive. Like the actual floppy disks themselves, the hardware specification hasn't changed in years. The floppy controller on a PC can't even figure out whether a drive is present or what size drive it is. It just tries to access the media based on a predetermined configuration, and if it fails, it simply tells the operating system that it can't cope.

However, this design is something of a blessing for VMware Workstation, because you don't have to worry about guest system support. You can configure one or two virtual floppy drives as mappings to real floppy drives on your host system or as floppy image files on the host operating system, and as you work with floppy disks under VMware, you'll probably realize that the latter method is much more convenient. Floppy disks not only tend to be unreliable, but they're also slow. If you can just yank a floppy image off the disk at once and point VMware at the file you created, you'll get your work done quickly. You can also interact with floppy images on the host system. (You'll see how to do that in sections 11.2 and 11.3.)

The floppy controller resides at IRQ 6 on a PC, with I/O ports at 0x3f0 to 0x3f5 and 0x3f7.

To get around the deficiencies of the old floppy devices, vendors have started to offer USB and even IDE floppy disk drives. In addition to VMware Workstation's USB capability, a USB floppy drive on a Linux host system shows up as a SCSI drive, and you can map it to a virtual SCSI drive. The benefits are somewhat unclear though: If you're interested in getting something off a floppy disk, it's usually much faster to pull the entire image off at once from your host system and throw the floppy disk away afterward.

2.2.4 Ethernet Interfaces

A *network interface card* (NIC) is an adapter that intercepts the signals on a network, filters them, and sends them to the CPU for processing.

Ethernet is the most common type of local area network. Its low cost has led many PC manufacturers to include onboard Ethernet interfaces on their machines. VMware's virtual Ethernet interface is an AMD PCnet II based on the AMD 79C970A chip.

You can add up to three interfaces to a virtual machine, in three kinds of configurations. A *host-only network* exists only in the host operating system and is used primarily for communication between the host and guest. In contrast, *bridged networking* multiplexes the host system's real Ethernet interface in much the same way that VMware multiplexes the host's CPU, so a guest system can talk on the host's network. *NAT networking* is a host-only network that uses network address translation to communicate with outside networks.

Like the VMware SCSI controller, the virtual Ethernet interfaces are PCI devices, so the interrupts and I/O ports are automatic at system boot.

Networking support is one of VMware's gems. After enabling it on your guest systems, you open up many new possibilities for interaction between your

host and guest systems. You aren't limited to just sharing files; you can use *any* feature that real networks offer, from SSH tunneling to remote print servers. Chapters 9 and 10 are dedicated to VMware networking and what you can achieve with it.

2.2.5 Serial Ports

An older type of peripheral interface, serial ports and lines transfer data between two devices one bit at a time. The most common type of serial interaction is between a computer and a modem; internal modems also show up as serial ports. The serial chip is called a UART (or universal asynchronous receiver-transmitter), and these are now integrated into motherboard chipsets.

PC UARTs have a long history of not being up to scratch. The original 8250 and 16450 chips didn't have adequate buffering to store incoming characters. This wasn't an issue with DOS, which didn't have multiple processes to worry about. But for real operating systems that were more than just program loaders, this became a problem at speeds over 2400 baud, because the UART often overwrote incoming data before the operating system kernel had a chance to pull the data out of the UART's buffer. To fix this problem, the 16550 UART came around with a first-in, first-out (FIFO) queue buffer. Unfortunately, this chip had a bug that rendered the FIFO buffer useless, and a replacement, the 16550A, was issued.

VMware Workstation has four serial ports with virtual 16550A UARTs. These ports are fixed at the PC default hardware configuration, shown here:

DOS Name	Linux Name	IRQ	I/O Ports
COM1:	/dev/ttyS0	4	0x3f8 to 0x3ff
COM2:	/dev/ttyS1	3	0x2f8 to 0x2ff
COM3:	/dev/ttyS2	4	0x3e8 to 0x3ef
COM4:	/dev/ttyS3	3	0x2e8 to 0x2ef

Under VMware, you can connect serial ports to actual devices, redirect output to files on the host system, or even attach serial ports to pipes and pseudo-terminals on the host system.

NOTE *Because the four ports share IRQs 3 and 4, you may run into trouble if you try to use the third or fourth port because some operating systems (notably, older versions of Linux) don't like to share serial port IRQs.*

2.2.6 Parallel Ports

Unlike serial ports, which send data one bit at a time to a device, parallel ports send eight bits at a time. The design was originally intended as largely unidirectional, which made it convenient for connecting a computer to moderate-bandwidth devices such as printers. As time went on, though, parallel ports gained greater bidirectional capability so that a device could talk back to the computer. Eventually, parallel ports became a sort of cheap alternative to SCSI. Though their bandwidth isn't very high and you can't really effectively use more than

one device on a parallel port at once, they come on practically every PC, so parallel port Zip drives and the like popped up.

VMware Workstation supports two PC parallel ports in unidirectional and bidirectional modes:

DOS Name	Linux Name	IRQ	I/O Ports
LPT1:	/dev/lp0, /dev/parport0	7	0x3bc to 0x3be
LPT2:	/dev/lp1, /dev/parport1	5	0x378 to 0x37f

Similar to serial ports, you can redirect the output of a parallel port to a file instead of attaching it to a real device on the host system with VMware Workstation.

2.2.7 USB Interface

VMware Workstation 3 includes support for the universal serial bus (USB). A relatively new development, USB is a moderate-speed interface intended for small peripheral devices such as keyboards, mice, printers, scanners, and removable media readers. It is *hot-pluggable*, meaning that you can connect and disconnect devices without powering off the computer. One other advantage is that most devices also receive power from the bus, reducing cable clutter and power outlet overload. USB devices transfer data like a serial port, but USB also includes multiple device support, like SCSI. Unlike the SCSI bus, where the host controller and devices reside on a chain as equal peers, a USB host controller is the root of a device tree. This tree branches at hubs and has leaves at the devices. Many devices include built-in hubs.

VMware Workstation's virtual USB hardware emulates the universal host controller interface (UHCI), a specific hardware specification. Any USB-enabled operating system should support it; not only is it the most common kind of host controller on current motherboards, but there is only one other interface type in common use, so operating system developers generally choose to support all USB interfaces. VMware maps your host system's USB host controller to a virtual controller. Like other PCI devices, the virtual host controller's interrupt and I/O ports depend on the virtual machine's particular configuration.

2.2.8 Graphics

On a PC, the video card does more than connect your computer to the monitor; it holds the memory for the color value of each dot on the screen and has instructions for drawing complex structures. After going through a number of incompatible video adapter types, the PC industry settled on VGA (Video Graphics Adapter) as a base standard; all subsequent adapter types implement VGA and work around it. The VGA standard has *text modes*—for example, an 80-column by 25-row setting that displays only text—and a 16-color *graphics mode* that can display 640 *pixels* (dots) across and 480 pixels down (also called *VGA16* mode). Almost immediately, vendors started to extend the graphics mode because it doesn't offer enough colors or resolution. These extensions tend to be called SVGA adapters, although standards are hard to come by. Conse-

quently, to use the "special" modes (which any sane person would use; VGA16 output is simply painful to look at), you must load your vendor's special driver into your operating system.

VMware's virtual graphics adapter works in the same way. The adapter has the text and VGA16 modes, but it also has its own special SVGA extensions that enable any resolution at the exact same number of colors as your host system. To enable these extensions, you must install the *VMware Tools* for your guest operating system. Aside from adding graphics capabilities, VMware Tools also dramatically improve graphics performance because they talk directly to VMware on the host system, sidestepping many layers of virtual devices. We'll discuss how to install VMware Tools for each guest system in Chapters 5, 6, and 7.

Another graphics enhancement that comes with the VMware Tools is *full-screen mode*, which eliminates the VMware window around the guest operating system and maps it to the entire screen of your host system. It's easy to switch back and forth between your host and guest environments.

2.2.9 Mouse

The IBM PS/2 platform had a new port for the PS/2 Auxiliary Device. This eventually became the standard for PC mouse ports, and most new mice have PS/2 mouse capability of some sort. This device is at IRQ 12 and shares I/O ports with the PC keyboard (0x060 to 0x06f).

VMware takes mouse events from the host operating system's windowing environment and funnels them to its PS/2 mouse. Using the VMware Tools, you can play a few extra tricks with the mouse to make it operate seamlessly with your host system. We'll look at this in section 4.11.

Early mice normally connected to a computer's serial ports. Naturally, VMware Workstation supports this configuration, since it supports serial devices. Of course, you'll likely need to attach an extra mouse to your computer if you want to use this configuration.

2.2.10 Sound Cards

The state of PC sound cards is somewhat chaotic. There are many different varieties, all with their own proprietary hardware schemes. Nevertheless, they all have a few things in common, like the fact that they all have digital signal processors and mixers. VMware Workstation translates the host's sound device to a Creative Technology SoundBlaster 16 card, at the factory settings: IRQ 5, I/O ports 0x220 to 0x22f, DMA channel 1 (16-bit DMA channel: 5). Because these are the defaults for most drivers, guest system support shouldn't be a problem, though VMware may not support all SB16 features (such as the MP401 MIDI UART).

NOTE *Because this sound card uses IRQ 5, you may have problems in VMware Workstation if you try to configure a bidirectional parallel port on the second parallel port when sound is present, because that port also uses IRQ 5. Many guest systems don't bother with IRQs on unidirectional parallel ports and instead use a polling interface, so you may have some degree of success with this configuration. The only way to find out is to try it.*

2.3 PC BIOS

All PCs have a *BIOS* (or basic input/output system), which is a small piece of firmware on the motherboard. The BIOS knows how to talk to a number of devices on a PC in a very limited capacity.

Older operating systems such as DOS relied on the BIOS for communication with the hardware, but newer systems use it only for basic configuration and booting. The BIOS performs the memory and peripheral tests when you turn on a computer and is responsible for the beep you hear after the tests. The BIOS may also display all sorts of information about your computer.

Some vendors like to hide their BIOS screens from you, though there's always a key sequence you can use to display a setup utility. This setup tool is essential to any BIOS.

Each vendor's BIOS has its own degree of complexity. The basics include your floppy disk configuration, a few hard drive parameters, the boot sequence, and power management options. Some BIOS types offer a staggering degree of customization beyond this. But regardless of the number of bells and whistles, the BIOS must store these parameters somewhere so that the user doesn't have to enter them every time the computer is turned on. The BIOS normally stores its configuration in *nonvolatile memory*, or *NVRAM*, that stays in place either in flash memory or battery-operated memory.

Like a real PC, VMware also has a BIOS based on the Phoenix BIOS. You'll find a description of its configuration utility in section 4.14. VMware stores its BIOS NVRAM in a file on the host system.

NOTE *You can find way too much information about the PC BIOS at http://www.wimsbios.com/.*

Another component on a PC's motherboard closely related to the BIOS is the *real-time clock* (*RTC*). This is a small, battery-powered digital clock that operates when the computer is off, and it normally is used for setting the initial time and date when the system boots. VMware maps the host system's true RTC to a virtual device. If something else is using the RTC, VMware emulates it with date-stamps from the host operating system—but this is not the same as the RTC.

2.4 How a PC Boots

Before you begin working with an operating system on a VMware virtual machine, you should know how a real PC boots. The PC's BIOS oversees the first stages of the process. Since there are several ways that you can boot a PC (from the floppy drive, hard drive, CD-ROM, or even a network), the BIOS looks at the devices in a particular order. First it may decide to look at the floppy drive, and if there's nothing in the floppy drive, it sees whether the hard disk has a boot block, and so on.

The BIOS approaches each kind of device differently. Floppy disks are very simple and have a fixed boot sector that loads a program on the floppy. Since floppy disks don't normally have partition tables, this scheme works fine. CD-ROM drives are a little more complicated; you normally put an image of a bootable floppy disk somewhere in the CD-ROM filesystem and activate a special ISO9660 extension to point to this image.

Because they have partition tables, hard drives are more complex. You normally put a boot loader on the partition of the operating system. Then, with a program such as `fdisk`, you mark that partition as active in the partition table. However, at boot time, the BIOS loads sector 1 on the disk, which is called the *master boot record*, or *MBR*. The BIOS runs whatever it finds on the MBR. Normally, this is a tiny program that tries to find an active partition in the partition table, and if it is found, loads yet another boot sector from that partition (which is your operating system's boot loader, from above).

This can be a problem if you have multiple operating systems on the same disk, as you may with dual-boot configurations with VMware. Some boot loaders, such as Linux Loader (LILO) and the FreeBSD boot loader, are capable of loading the boot sectors on other partitions. This is fairly typical with Linux and Windows dual-boot systems. The BIOS first loads LILO from the active boot sector (normally on a Linux partition), and you can jump from there to the boot sector on the Windows partition.

You can circumvent the active partition by replacing the default MBR with your own boot loader. You'll often find this scheme in place when the disk is too large for the BIOS to handle. LILO makes circumventing the active partition easy; for example, just use `/dev/hda` instead of `/dev/hda1` for the boot device. What is not so easy is to remove this custom boot sector once it is in place. You can use the DOS `fdisk /MBR` command or try to find the LILO backup of the original boot sector.