Chapter 2

ESX Architectural Overview

Now that you have a basic understanding of the VMware products and a basic knowledge of ESX and how it works, its time to go into detail as to what makes it all possible. As a warning, this chapter is not intended for those with a weak heart or who are prone to falling asleep. If you want to truly understand the magic behind ESX server then this chapter is for you! Also, make sure you have an empty stomach, because there's a lot of information to digest here.

The Console Operating System versus VMkernel

One of the most difficult concepts for new VMware admins to understand is the difference between the Console Operating System (called "COS") and the VMkernel. Both play extremely important roles in your ESX environment and it's important to fully understand what each does and is capable of doing. A common misconception of ESX is that it "runs on Linux." Let's set the record straight once and for all: ESX is not Linux. It's not derived from Linux, and does not run on Linux. Now that that's out of the way, let's get back to the COS and the VMkernel.

The easiest way to distinguish the differences between these two components is to think of the server's console as being the "physical world" and the VMkernel as the "virtual world." The console lets you touch and interact with it directly, and it allows access to modify configurations and manage the environment. The VMkernel manages everything that relates to the "virtual world" and the guests that run within the host.

Console Operating System

The COS is used to boot your system and prepare your hardware for the VMkernel. As the COS loads, it acts as the bootstrap for the VMkernel which means it prepares all the necessary resources for turnover to the VMkernel. Once the COS has loaded ESX, the VMkernel will warm boot the system, assuming the role of primary operating system. The VMkernel will then load the COS and several other "helper worlds" as privileged VMs. The COS is responsible for quite a few things that are vital to the proper operation of ESX, including:

- *User interaction with ESX.* The COS is responsible for presenting the various methods to communicate with the ESX host system. It runs services that allow user interaction with the host using various methods such as:
  - Direct console access
  - Telnet/ssh access to the console
  - Web interface
  - FTP
- *Proc file system.* The proc file system can be utilized by both the COS and the VMkernel to provide real time statistics and to change configuration options on the fly. The proc file system will be discussed in greater detail in Chapter 6.
- *Authentication.* There are several processes that run within the COS that provide authentication. These mechanisms determine what rights a specific user ID has over the COS itself and the various guests running on the host. Chapter 7 is dedicated to these processes and how they interact with each other, the VMkernel, and the COS.
- *Running Support Applications*. There are some applications that may be run within the COS to provide extended support of the host environment. Every major hardware vendor has some form of agent that will run within the COS that can detect hardware issues as they arise (voltage problems, drive failures, fans quitting, etc.) In some scenarios it may also be necessary to install a local backup client to the COS to backup critical system files. The number of applications that are installed and run on the COS should be limited though, as the COS is really designed to support ESX and nothing else.

VMkernel

Once the OS has loaded, the VMkernel is started. At this point the VMkernel will warm boot the system and assume responsibility for all hardware management and resource scheduling is turned over to the VMkernel for management. Even the COS gets reloaded by the VMkernel as a VM and is restricted by the VMkernel and its configurations. (Okay, technically, the COS still manages it's own memory and NIC, but that's it.) The COS must follow the same rules for resource allocations and sharing as every virtual guest running on the host.

The VMkernel performs a number of functions, but one of the main jobs it has is to manage the interaction of virtual machine hardware and the physical hardware of the server. It acts as the "go-between" for scheduling resources for VMs on an as needed and as configured basis. While this may seem like a brief and simplistic description of the VMkernel, the remainder of this chapter focuses heavily on how the VMkernel works its magic and makes ESX what it is.

The ESX Boot Process

By taking a look at the boot process of an ESX host we can see how the COS and the VMkernel interact and at what point the VMkernel takes control of the system resources. There are several steps to the boot process. While we won't cover all of them, we will highlight the ones that perform important system tasks as they relate to ESX.

LILO

LILO (or the "Linux Loader") is a boot loader application (similar to ntloader for Windows) that the system reads when it boots from the hard drive. Based on the information contained in /etc/lilo.conf file, the system begins its boot process. The default boot option for LILO within ESX is to boot and load the VMkernel. The /etc/lilo.conf file also contains information on how the COS should be configured as it is booting. This information contains the amount of memory allocated to it and the devices that are configured for COS use. Many of the LILO configuration items are controlled by the vmkpcidivy command, discussed later in this chapter

Console Operating System

After LILO properly initializes the boot instructions the COS begins to load. The majority of the boot process is contained in the COS.

Most of these steps are used to prepare the VMkernel to take control of the hardware resources.

init

The first process that the COS executes is init. This process reads the /etc/inittab file and determines the system runlevel that should be executed. (A Linux runlevel determines what services are started on the server and the order in which they are started.) Varying runlevels on a Linux system is comparable to the various boot options available on a Windows server such as “Safe Mode” or “Command Prompt Only.” The default system runlevel for ESX is 3, which means the system will boot and present a console for system login. Based on this value the COS will run the scripts contained in the /etc/rc.d/rc3.d directory during the boot process.

/etc/rc.d/rc3.d

The /etc/rc.d/rc3.d directory actually contains symbolic links to start scripts in the /etc/init.d directory. By running an “ls” command in the /etc/rc.d/rc3.d directory you will see some scripts that begin with a K and some that begin with an S. Scripts that begin with a K are used to stop (or “kill”) a service during the boot process (or ensure it is not running) and scripts beginning with an S are used to start a service. You will also notice a number after the S or the K in the script’s file name. These determine the order the script is run, starting at 0 and going up to 99. The S scripts are executed in ascending number order, whereas the K scripts are executed in descending order. The order of the K or S values in the file have no meaning when it comes to what order a script runs in.

S00vmkstart

If you run an “ls –l” command in the scripts directory you’ll notice that the S00vmkstart command actually links to a script called vmkhalt. By running this script first, VMware ensures that there are no VMkernel processes running on the system during the boot process.

S10network

The network script (S10network) starts the TCP/IP services on the COS and assigns the IP address and hostname of the system.

S12syslog

The syslog script starts the daemon that processes system logs. Starting this script allows the remainder of the boot process to be logged. After the VMkernel begins to load it also provides a mechanism to capture the log files generated by the VMkernel for review when errors occur.

S56xinetd

The xinetd script starts the services required for the COS to handle incoming requests for access. Each application that can be started by xinetd has a configuration file in /etc/xinetd.d. If the "disable = no" flag is set in the configuration file of a particular application then xinetd starts the application. (Yeah it's a double-negative.) The most important application that is started here is the vmware-authd application which provides a way to connect and authenticate to ESX to perform VMkernel modifications.

S90vmware

This is where the VMkernel finally begins to load. The first thing that the VMkernel does when it starts is load the proper device drivers to interact with the physical hardware of the host. You can view all the drivers that the VMkernel may utilize by looking in the /usr/lib/vmware/vmkmod directory.

Once the VMkernel has successfully loaded the proper hardware drivers it starts to run its various support scripts:

- The vmklogger sends messages to the syslog daemon and generates logs the entire time the VMkernel is running.
- The vmkdump script saves any existing VMkernel dump files from the VMcore dump partition and prepares the partition in the event that the VMkernel generates unrecoverable errors.

Next the VMFS partitions (the partitions used to store all of your VM disk files) are mounted. The VMkernel simply scans the SCSI devices of the system and then automatically mounts any partition that is configured as VMFS. Once the VMFS partitions are mounted the

VMkernel is completely loaded and ready to start managing virtual machines.

S91httpd.vmware

One of the last steps of the boot process for the COS is to start the VMware MUI (the web interface for VMware management). At this point the VMkernel has been loaded and is running. Starting the MUI provides us with an interface used to graphically interact with ESX. Once the MUI is loaded a display plugged into the host's local console will display a message stating everything is properly loaded and you can now access your ESX host from a web browser.

So why do I need to know the boot process?

You need to understand the basic boot process to understand that the VMkernel is a separate entity from the COS. Also if your server fails to boot or certain services or processes fail to start you'll have a good idea of where to start looking for problems. If you're not familiar with Linux then this is all probably very new to you. If you have some experience with Linux then this section just helped you understand how the VMkernel fits into the picture.

Now let's take a look at how the VMkernel does its magic and "virtualizes" the hardware.

Hardware Virtualization

The whole idea behind VMware is to present a standard hardware layer as a virtual machine to the guest operating systems. These virtual hardware resources remain constant regardless of what physical hardware makes up the host.

The VMkernel is responsible for providing the virtual hardware layer to virtual machines. When a guest OS accesses a resource the VMkernel is then responsible for mapping the virtual request through to the physical hardware for processing. Since VMware presents standard hardware to virtual guests, you need to be familiar with this hardware. Some resources such as SCSI and the network have several options, so we need to understand when each option is used and what it changes in the environment with each.

System Devices

When ESX presents a hardware layer to a guest operating system it presents a system based on Intel's 440BX chipset. This is a highly supported chipset and is compatible with every guest operating system that can be run within ESX. You may be wondering how we can run Pentium 4 XEON and AMD Opteron processors in a guest that has a 440BX chipset. While this sounds a little off, we'll describe that in detail a later in this chapter. For now you just need to understand that the 440BX is what is presented to the guest and it allows for a high degree of compatibility across numerous platforms for our guest operating systems.

Processors

Assuming you are utilizing a processor that meets the requirements of ESX server, your guest will see the same type of physical processor that's installed on the host. The VMkernel is capable of accepting processor calls and handing it straight to the physical processors of the host with limited virtualization overhead. By presenting the host processor type to the guest, the VMkernel does not need to perform any conversions to ensure compatibility between the physical and hardware layer. This simply means that the processor is NOT accessed through on emulation layer. And that if your host has an MP or DP type processor then that's what is presented to the guest.

It's important to note that not all registers of the physical CPU are presented by the VMkernel. While VMware is fairly tight-lipped about these registers, one that is known for sure is processor serial numbers. Applications that are licensed to a serial number of a processor or group of processors will not function within VMware.

Network

ESX provides us with two hardware options when presenting virtual network adapters to guest operating systems. Depending on the guest operating system, there may be a requirement for one over the other.

VLANCE

The vlance adapter is a virtualized AMD PCNET driver. This adapter has guaranteed compatibility across every guest operating system that can be run within ESX. Since it's based on legacy hardware it will also have some limitations when utilizing it within a guest. After installing the drivers you'll notice that the connection speed within the guest operating system shows 10Mb/sec. This is a limitation of the driver and in fact doesn't impact the transfer rate of the hardware. The vlance adapter will utilize as much bandwidth as is available to the physical connection. There is native support for this device in every operating system that ESX has been certified for. If you're configuring a DOS boot disk for a network based installation or to use a DOS based tool such as Ghost, this is the only driver that will properly function. Using this driver will require increased virtualization overhead over the other networking option available to us.

VMXNET

VMware has created a virtual network device that was designed from the ground up to interact with the VMkernel. This device is the vmxnet adapter. Because of its tight integration with the VMkernel you will receive enhanced performance when using it in a guest, especially with high speed connections. Since this device is a VMware creation there is no native support for it in any guest operating system. The only way to configure this device is to install the drivers provided by the VMware Tools installation package within the guest. Using this adapter minimizes the amount of virtualization overhead and increases the network performance for a guest OS. It's important to note that not all operating systems will have the capability to use this device. Use of this device is based strictly on the availability of a VMware Tools installation package and vmxnet driver for the target guest.

SCSI

Like the virtual network adapter, VMware provides two different SCSI adapters that may be presented to a guest operating system. The device that's used by your specific guest depends on the operating system that will be installed. The two options available to use are an LSI Logic adapter or a Bus Logic adapter. Each adapter has different levels of support in each of the supported operating systems. To eliminate any error when building a guest, ESX automatically assigns the proper controller during the virtual machine configuration wizard based on operating system choice. While the default controller may be changed in some cases, it typically requires additional drivers to first be installed in the guest. It may also impact the performance of the virtual machine (more on this in Chapter 4). As a general rule of thumb the choices that VMware makes for us guarantee compatibility for our guest servers.

As you can see the virtual hardware presented to the guests create a relatively flexible environment that can be used by almost any mainstream Intel OS. Now that you have a basic understanding of the virtual hardware as presented to the guests, let's look a little more at how hardware is divided between the physical and virtual worlds.

Hardware Allocation

When installing and configuring ESX, you'll see that both the COS and the VMkernel are responsible for controlling certain aspects of the hardware. There are three different settings for your hardware devices: Virtual, Console, or Shared. Devices that are allocated as "virtual" can only be accessed by the VMkernel (the virtual world). "Console" devices are those that are limited to functioning in the console operating system (the physical world). The third option is a mix of the two and allows for a device to be accessed in both the COS and the VMkernel (physical and virtual worlds). There are also several different ways in which the device allocations may be changed to accommodate changing needs of the environment.

Virtual

Virtual devices, as stated above, may only be accessed by the virtual guests running on your host. The first obvious device that would be configured for virtual machines is at least one network adapter. (Later in this book we'll discuss why you would want to strongly consider using multiple adapters). Configuring a network adapter for virtual machine use is the only way that your guests will be able to communicate with the network outside of your host server.

In addition to network connectivity you also need a place to store the data that makes up your guest. In order to do this a SCSI adapter should also be assigned for virtual machine (which means the VMkernel). To simplify things for now, ESX also considers fiber adapters to be SCSI adapters in terms of virtual, console, or shared configuration. Depending on the size of your environment and the type of data you'll be connecting to, you may or may not have a need for fiber HBAs or additional SCSI adapters.

Console

While "virtual" devices are only be seen by your virtual guests, "console" devices, as you may have guessed, are only seen by the COS. Every ESX host has at least one network adapter that is used by the service console. (This adapter is usually, although not always, dedicated to the COS.) When you communicate to the host with the MUI or connect via ssh, you're interacting with this network interface. When you install backup or management agents on the console operating system, this adapter is also used to communicate through the network.

In order for the console operating system to properly boot, it needs a disk controller to be allocated to Console use. Since the COS is a standalone operating system (just like Windows) it needs a hard drive configured so it can create and use the partitions and files required to boot. This can be a physically attached hard drive, or in the case of ESX 2.5 or newer, a remote boot off of a SAN. (More on this in Chapter 4.)

We should note that you don't need a disk controller that's dedicated to the COS only. You just need to have a controller (either "shared" or "console") that's available to the COS so it can boot.

Shared Resources

Shared resources are those that can be accessed by both the VMkernel and the COS at the same time. Consider the situation we just alluded to previously where you have a system with only one SCSI controller and no SAN in your environment. In order to hold large amounts of data you purchase a SCSI drive cage that externally attaches to your ESX host. Since you only have the one SCSI adapter, you need to make sure the console has access to the internal hard drives for the installation of the COS. We also need to make sure that once ESX is installed the VMkernel will have the appropriate access to the external drive cage.

Shared devices are not limited to SCSI controllers, but may also be fiber HBAs or network adapters. In Chapter 4 we'll introduce you to some of the advanced configurations available to you utilizing the shared bus mode.

Modifying These Configurations

During the installation process of ESX we'll show you how to initially allocate your devices in Chapter 3. As your needs change and your virtual environment grows it's essential to know that you can modify which devices are seen in the physical and virtual worlds. Fortunately VMware provides several tools to make this possible.

MUI

Modifying device allocations utilizing the MUI (the web interface) is a relatively simple process. In order to access the configuration screen you need to log into the MUI as the root user. This will enable the "Options" tab on the main page. The top link in the left column of the options tab will be "Startup Profile." This is the area where you can configure HyperThreading options and memory resources for the service console, and device allocations.

Device allocation configuration through the MUI is somewhat limited in that the only devices that can be configured as "Shared" are SCSI and Fiber Storage Adapters. In order to share a device you must choose to allocate it to "Virtual Machines" and then select the "Share with Service Console" checkbox. You'll notice that the network adapters installed in the system do not have this option. Configuring the network devices for shared use is an advanced configuration and not recommended unless certain conditions are met. This limitation should be sufficient for a vast majority of the implementations, but we feel it's important to note that you cannot do everything from the MUI. Details on this can be found in Chapter 4.

Console Operating System

Modifying device allocations through the service console can be done with the vmkpcidivy command. This command can be run in two different ways: interactive and batch mode.

Running vmkpcidivy in interactive mode is the easiest way to configure your devices outside of the MUI. You can run vmkpcidivy in interactive mode by accessing the service console (locally or via ssh) and using the following command:

```
# vmkpcidivy -i
```

After executing the interactive mode command you'll be presented with a list of configurable devices in the system and how they're currently allocated. (This is shown in the Example: 2.1). The devices are presented in a list categorized by their allocations. You'll see "Shared" devices listed twice—once under the Console section and once in the Virtual Machines section.

Example 2.1

```
[root@esx1 root]# vmkpcidivy -i
Checking for existing VMnix Boot Configurations.

The following VMnix kernel images are defined on your
system:
Boot image configuration: esx
Image file: /boot/vmlinuz-2.4.9-vmnix2
```

```
Memory: 192M
Service Console devices:
Ethernet controller: Intel Corporation 82557
[Ethernet Pro 100] (rev 08)
RAID storage controller: Symbios Logic Inc. (formerly
NCR) 53c895 (rev 02) (shared)
VM devices:
Ethernet controller: 3Com Corporation 3c905C-TX [Fast
Etherlink] (rev 78)
RAID storage controller: Symbios Logic Inc. (formerly
NCR) 53c895 (rev 02) (shared)
Type in the name of the boot image configuration you
wish to configure or type "new" to create a new image
[esx]:
```

Following the list of devices is a prompt asking if you would like to modify an existing configuration or create a new one. The default configuration name for ESX 2.1.1 and higher is ESX. Prior to 2.1.1 the default configuration name was VMNIX. You can tell what your default is by either paying attention to the LILO boot menu at start-up or by viewing the /etc/lilo.conf file with the following command:

```
      # grep default /etc/lilo.conf
```

By choosing your default configuration you will be presented with each device and its current setting. When presented with a list of devices, the current values will be set as the "default" values. By simply hitting enter the particular device that's listed will keep its current configuration.

There are three possible values to use for allocating your devices: c, v, or s. These represent Console, Virtual Machines, and Shared, respectively. When you get to the particular device(s) you wish to modify, enter the proper value and hit enter. Once you've gone through the entire list of devices you'll be prompted as to whether you want to apply the configuration changes or not. Once you've chosen to apply the changes you will need to reboot the system the changes to take effect. If you're using vmkpcidivy for information gathering you'll either want to break out of the application using CTRL+C or choose not to apply the changes to the configuration.

We strongly recommended that if you're unsure of the change you're attempting to make that you create a new configuration with a proper temporary name such as "esx-test." This will require that you type "new" at the first prompt followed by your temporary configuration name at the second. When you create a new profile, the settings from your original profile are not remembered. You'll have to pay close attention to each option presented to ensure your system comes up in a functional state after its next reboot.

The Core Four Resources

There are four resources that you need to strongly consider when you review and design your virtual environment. (These are what we've starting calling "Core Four.") Properly understanding and configuring these resources are essential to maintaining a stable virtual environment. This section focuses on these "Core Four" resources and how they pertain to VMware ESX Server and its guests.

Processor

As mentioned previously, the virtualization of the processor component that is presented to virtual machines is slightly different than other devices. As we discussed, the motherboard architecture presented to the guest operating system is based on the Intel 440BX chipset which is a Pentium III-based motherboard. So how does this impact the physical processors that are installed in the host server?

This simple answer is, "it doesn't." The best way to describe how VMware virtualizes the processor was described to us by one of the VMware sales engineers that we frequently have the opportunity to work with. Since the system architecture as presented to the guest operating system is 440BX-based, the device manager within Windows shows you all the typical components of that "virtual motherboard." The single exception in this virtual motherboard is that at the hardware level there's a hole cut out of the motherboard where the processor belongs.

The VMkernel, based on processor family, presents the specific capabilities of your host processors to the guest operating system which allows full utilization of the processors installed. While there are some registers that are not virtualized, the guest has the capability to benefit from the key registers of advanced processors such as Pentium 4 XEON and AMD Opteron. Simply put, the processors are not really virtualized the same way as the other “core four” (memory, disk and network) are. The processor utilization is scheduled, but what the guest sees is pretty much what it gets.

Hyper-threading

Hyper-threading is an Intel technology that allows a single processor to execute threads in parallel, which Intel claims can boost performance by up to 30%. If you think about it, what Intel is really doing is presenting two processors to the OS for each physical processor installed. This idea and technology really comes from trying to make up for the poor task scheduling that Microsoft does in its Windows operating systems (but that’s another book all together).

VMware ESX 2.1 introduced support for hyper-threaded processors. The additional logical processors are packaged with the physical and are numbered adjacently. For example, processors 0 and 1 would be physical CPU 1 and its logical counterpart (and 2 and 3 would be CPU 2 and its logical counterpart, etc.). This behavior is different than that displayed in a typical x86 operating system in that all physical CPUs are counted first, and then the logical CPU pairs are numbered. It will be important to remember this numbering system if you begin to use the more advanced concepts of “pinning” a Guest VM to a specific processor for performance reasons.

The increase that a system receives from hyper-threading is dependent on how well the applications running on the system utilize the system cache. While a typical operating system requires that hyper-threading be enabled or disabled for the entire system, VMware has provided several mechanisms for configuring hyper-threading sharing on the system that can be configured on a per virtual machine basis:

- *Any.* This is the default setting for virtual machines running on the system. This allows multiple virtual CPUs to share a single processor package at the ESX level. It allows you to get the most out of enabling hyper-threading on your system, but can introduce problems where an inefficient application may impact overall performance of the other virtual machines sharing a package with it.

- *Internal.* This option is only supported by SMP (multiprocessor) machines. It allows both virtual CPUs for a virtual machine to run in a single package and isolates it from any other virtual CPUs in the system. This prevents the configured guest form impacting other guests and protects it from other guests that may have inefficient applications. If overall system utilization allows, a guest configured with internal hyper-threading sharing can still utilize a package per virtual CPU to maximize performance.
- *None.* In cases where an application is known to perform poorly with hyper-threading, sharing can be disabled. This completely isolates each virtual CPU of the system to its own package. This option should only be used when suggested by VMware or an application vendor as it isolates a large amount of system resources.

Modifying the hyper-threading settings of the system may be done one of three ways. Your system must have hyper-threading enabled at the hardware level in order to view and modify these options. Also, the virtual machine must be powered off for these modifications to be possible.

- *MUI.* By using the MUI, the hyper-threading sharing option can be modified two ways. The first is by editing the CPU Resource Settings for the virtual machine. You'll be presented with a checkbox that's labeled "Isolate Virtual Machine from Hyper-Threading." The behavior of this setting depends on whether the system is a single processor system or if it has Virtual-SMP enabled. For a single processor machine this option will set the sharing value to "none." For SMP machines, the value will be changed to "internal. The other option is to use the verbose configuration option for the guest. When presented with a list of configuration options, add (or modify) a value titled "cpu.htsharing." Assign the value of "any, internal, or none" to this option.

- *COS.* You can easily set the hyper-threading sharing value by directly modifying the VMX file for the virtual machine in question. The easiest way to modify the file is to utilize the following command:

```
# echo cpu.htsharing = \"value\" >> /path/to/server-
name.vmx
```

Make sure you substitute the proper value inside the escaped quotation marks. The escape character of "\" is required for echo to properly insert the quotation marks into the configuration file. Another thing you MUST be careful of is that you use the double output symbol ">>". If you use a single output symbol the existing text of the file will be overwritten. It's always recommended that you make a backup copy of the VMX file before making any modification. If you're familiar with the vi editor then you may use that to modify the VMX file with the following line:

```
cpu.htsharing = "value"
```

There's one final note about using hyper-threading in your ESX environment. If you're using a system that utilizes a NUMA memory architecture, it's strongly recommended that you only use ESX versions 2.1.2 or higher. There have been specific issues tied to these systems that appear to have been fixed by the 2.1.2 updates. While there is no "official" announcement from VMware on this, internal employees have confirmed that update code was included to enhance this functionality. Non-NUMA systems seem to perform extremely well with hyper-threading enabled regardless of the ESX version.

Symmetrical Multi-Processing (also known as SMP or Virtual-SMP)

SMP is an add-on module for ESX that provides the capability to configure multi-processor guest operating systems. You enable Virtual SMP by plugging in a license key to your ESX host either during the install process or by modifying the licensing options afterwards. (Note that you do NOT need an SMP license to use ESX on a multiprocessor host. You only need it to create VMs that use multiple physical host processors.) While SMP can provide enhanced performance to your system, there are several guidelines that should be strictly followed as SMP can just as easily negatively impact an environment.

- Administrators should never start off by configuring a virtual machine as an SMP system.
- Once upgraded to SMP, it is extremely difficult (and sometimes impossible) to properly downgrade a Windows guest.
- Utilizing SMP slightly increases CPU and memory overhead of an ESX host.

While performing best practices analysis of environments we've notice there are quite a few people who start off deploying every virtual machine as an SMP system. The added virtualization overhead from this configuration can be the source of significant performance problems once the environment becomes fully utilized.

Additionally, "downgrading" the kernel of a Windows system is a sensitive process with Windows 2000 and is not possible with Windows 2003. It's recommended that all guests be created as single processor machines and then if performance dictates and the application is capable of fully utilizing SMP technology it's a simple process to upgrade the system to SMP. By only utilizing SMP on guests that are capable of taking advantage of it, the virtualization overhead of an ESX host is kept low allowing the system utilization to be maximized.

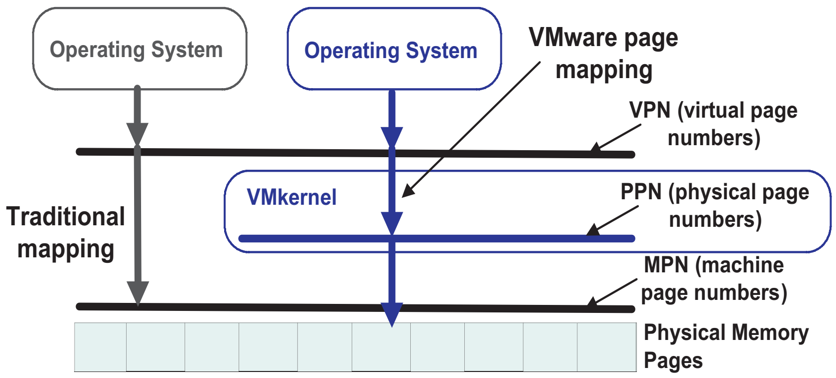
One final note, SMP is only supported on ESX Server—not GSX or Workstation.

Memory

Memory utilization within ESX is managed at many levels. To start, the COS is given dedicated memory resources as part of its boot process (based on the configuration choices you made during ESX installation). This memory is used to support the operation of the COS and virtualization overhead for the service console and each running virtual machine. The COS is allocated 24MB of memory from its available bank for its own virtualization. This is performed automatically and cannot be customized. Each virtual machine that's powered on requires memory in the COS space to support virtualization. (This is the "virtualization overhead" that you hear people talking about). We'll further discuss the details on virtualization requirements and console configuration in Chapter 3 when we install ESX server.

The remaining physical memory that's not assigned to the COS is presented to the VMkernel for virtual machine use. The way memory is controlled and accessed is complicated by the fact that the VMkernel intercepts memory pages and presents them to the guests as if it were continuous memory pages. This process is somewhat complex and described about a hundred different ways depending on who you talk to.

Figure 2.1



We've found that the process is best described in Carl Waldsurger's "Memory Resource Management in a VMware ESX Server" document. Summarizing this document (and following Figure 2.1):

- VMkernel takes machine page numbers (MPNs) and stores them as physical page numbers (PPNs).
- MPNs are memory pages that are located in the physical memory of the host server.
- PPNs exist only within the VMkernel and are used map MPNs to virtual page numbers (VPNs) that the VMkernel

presents to the guest operating systems. By presenting VPNs to a guest in a contiguous manner, the VMkernel gives the illusion that contiguous memory space is being used within the guest.

- "Shadow page tables" help to eliminate virtual machine overhead by allowing direct correlation of VPNs to MPNs. The VMkernel keeps these mappings up-to-date as the PPN to MPN mappings change.

Later in this chapter we'll describe something called "transparent page sharing." This process (or architecture) allows for a certain amount of secure memory sharing between VMs. The memory resource management used by VMware is what makes transparent page sharing and other memory saving features possible.

NUMA

With the increasing demand for high-end systems, today's hardware vendors needed an affordable and easily scalable architecture. To answer these needs the NUMA (Non-Uniform Memory Access) architecture was developed and adopted by several hardware vendors. NUMA functions by utilizing multiple system buses (nodes) in a single system connected by high speed interconnects. Systems that have NUMA architectures provide certain challenges for today's operating systems. As processor speeds increase, memory access bandwidth becomes increasingly more important. When processors must make a memory call to memory residing on a different bus it must pass through these interconnects—a process which is significantly slower than accessing memory that is located on the same bus as the processor.

NUMA optimizations that were included with version 2.0 of ESX have turned ESX into a fully NUMA-aware operating system. These optimizations are applied using several methods:

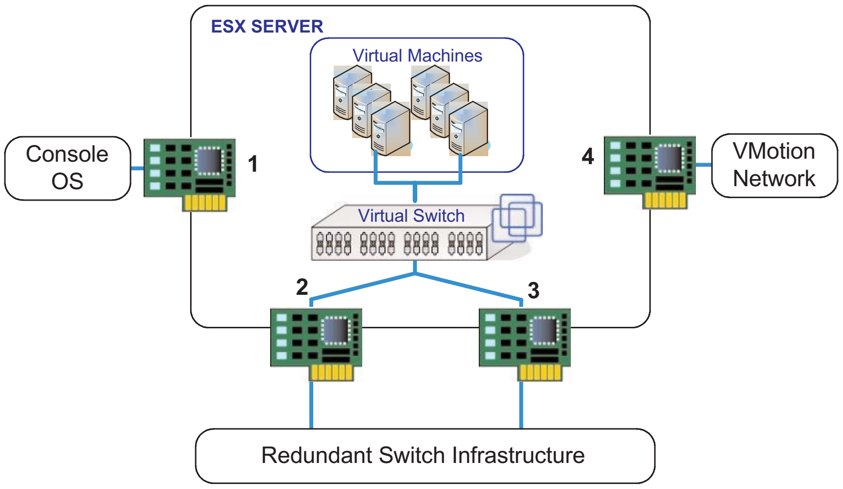
- *Home Nodes.* When a virtual machine initially powers on, it's assigned a home node. By default it attempts to access memory and processors that are located on its home node. This provides the highest speed access from processor to memory resources. Due to varying workloads, home nodes alone do not optimize a system's utilization. For this reason it's strongly recommended that NUMA nodes remain balanced in terms of memory configuration. Having unbalanced memory on your nodes will significantly negatively impact your system performance.

- *Dynamic Load Balancing.* At a default rate of every two seconds, ESX checks the workloads across the virtual machines and determines the best way to balance the load across the various NUMA zones in the system. If workloads are sufficiently unbalanced, ESX will migrate a VM from one node to another. The algorithm used to determine which VM to migrate takes into consideration the amount of memory the VM is accessing in its home node and the overall priority of the VM. Any new memory pages requested by the VM are taken from its new node while access to the old pages must traverse the NUMA bus. This minimizes the impact of a guest operating system from a migration across nodes.
- *Page Migration.* While dynamic migration of a virtual machine across nodes limits the impact on the guest, it does not completely eliminate it. Since memory pages now reside on two nodes, memory access speeds are limited by the fact that the processors do not have direct access to them. To counter this, ESX implements a page migration feature that copies data at a rate of 25 pages per second (100 kb/sec) from one node to the other. As it does this the VMkernel updates the PPN to MPN mapping of memory to eliminate virtualization overhead.

Network

Like everything else in the VMware environment, network resources exist in two worlds. The use and configuration of a network resource is quite different depending on whether it's assigned to the COS or to the VMkernel. While the COS configuration is fairly straightforward, virtual adapters have a wide variety of options. In fact, there is enough information around this technology that we've dedicated an entire chapter to it later in this book. Figure 2.2 represents what a typical VMware network configuration would look like on an ESX host.

Figure 2.2



Console NIC Configuration

This adapter is utilized by the COS (remember COS is the "console operating system") for management tasks. ESX management, backups of VMware configuration files, and file copies between ESX hosts are done over this interface.

While this interface is not typically used as much as the interface that the production VMs use, you still want it to be fast if you'll be using it to backup VM disk files. This interface is known to the COS as eth0, and it will commonly be referred to as the "management adapter" throughout this book. By default, the first NIC the COS detects is assigned eht0, and it requires a unique IP address. This does not mean the console NIC will always be an on-board Ethernet interface. Depending on how the system BUS numbering is laid out, it's entirely possible that a PCI slot will act as eth0 for the system.

VMNIC Configuration

ESX assigns adapters configured for virtual machine using sequential names starting with "Outbound Adapter 0" which maps to "Physical Adapter 1." These adapters are labeled as NICs 2 and 3 in Figure 2.2. Like the COS adapter, this numbering is determined by system BUS order. Please refer to Figure 2.3 for Physical Adapter to Outbound Adapter mapping.

Figure 2.3

Outbound Adapter	Physical Adapter
0	1
1	2
2	3
3	4
etc.	etc.

To prevent network bottlenecks on VMware, Gigabit Ethernet connections should be used whenever possible. There are ways to assist in limiting the amount of bandwidth that travels over the wire which will be described later, but for production use, gigabit connections allow you to provide adequate resources to each VM.

Virtual Switch

Virtual switches were introduced in ESX 2.1 as a new way to assign network connectivity to virtual machines. While virtual switches can be created with a single physical NIC (or in some cases, no physical NICs), the failure of that NIC would cause the virtual machines using it to lose their network connectivity. To prevent this downtime, ESX allows us to bond up to 8 gigabit Ethernet adapters (up to 10 10/100 adapters) together to present to virtual machines. A virtual switch is just what its name describes—it emulates a 32-port switch for the guests that are configured to utilize it. Each time a virtual machine references a virtual switch in its configuration it utilizes one port. Virtual switches also load-balance virtual machines across all physical NICs used to create the switch. If one network switch port or VMNIC were to fail, the remaining VMNICs in the bond that makes up the virtual switch would pick up the workload.

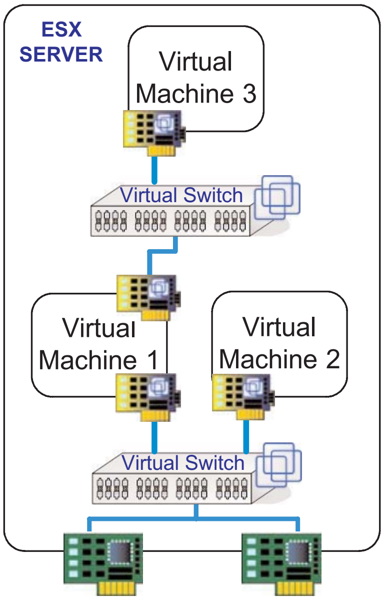
Another feature of the virtual switch is that any traffic between VMs on the same virtual switch is typically transferred locally across the system bus as opposed to the across network infrastructure. This helps to lessen the amount of traffic that must travel over the wire for the entire host. An example of where this may be used is a front end web server making database queries to another server configured on the virtual switch. The only traffic that traverses the network infrastructure is the request from the client to the web server.

During the creation process of a virtual switch in VMware you are prompted for a "Network Label." This label serves several purposes, the first being that it's a meaningful way to manage the switch. If you give it a name of "Trusted" or "VLAN17," you have a good idea as to the exact specifications of the switch. The other purpose is for utilizing VMware's VMotion technology. In order to move a virtual guest from one physical host to another, all settings on both hosts must be identical, including virtual switch labels.

Virtual Network (VMnet) Configuration

Feeding off the virtual switch methodology, VMware has also implemented what they call virtual networks, or "VMnets" in ESX. This feature provides the capability to create a private network that is visible only to other hosts configured on the same VMnet on the same physical host. VMnets are simply virtual switches that don't have any outbound adapters assigned to them. Using this feature it's entirely possible to create a multi-network environment on a single host. In the Figure 2.4, traffic coming in through the external virtual switch would have no way to directly communicate to the Virtual Machine 3 and would only be able to interact with Virtual Machine 1 or 2.

Figure 2.4



There are several instances in which this configuration may be beneficial:

- Testing in a secure isolated environment. Setting up a VMnet within VMware does not require any physical network connectivity to the box beyond the management NIC. As an example, to see how a group of schema changes impacts an Active Directory, a parallel directory could be created in the virtual environment and tested without ever tying into a production network.
- DMZ Architecture. Since this design creates an isolated environment, it could be created to secure critical systems using network address translation (NAT). This would entail a firewall or reverse proxy being connected to both the virtual switch and the VMnet. Requests from clients would come into this server over the virtual switch. The firewall would then initiate its own connection to the backend web server, which in turn receives data from a database. The information is then returned back to the client without ever having the capability to talk directly to the web or database server.

Storage

Storage for ESX comes in two flavors: local and SAN. With ESX 2.1.2 and lower, every system (at least every supported system) requires local storage for the installation of the COS. In the introduction of ESX 2.5, VMware provided limited support for a "boot from SAN" option which eliminates the need to have local storage connected to the system at all. Before we get into the differences of locally-attached and SAN-attached storage we should understand what it is that will be utilizing these resources.

Virtual Disk Files for VMs

Each VMware virtual disk file represents a physical hard disk that can be allocated to a virtual machine. VMDK files may be up to 9TB in size, which is a limitation that should never really be encountered. Any system that would require that much storage would most likely fall outside of the "VM Candidate" range as described in Chapter 5. ESX does not allow dynamically expanding disks (also known as "sparse" disks) like GSX does. Any VMDK file that is created has all space immediately allocated to it. This is done for a very good reason. It eliminates fragmentation of the hard drive that the VMDK resides on. If you've used VMware's GSX or Workstation products, you may have noticed that heavy fragmentation occurs when a VMDK file must expand itself as it grows. By immediately allocating all space when the vDisk file is defined ESX configures a VMDK file in a relatively contiguous layout on the physical disk.

Allocating the full disk space immediately also ensures that the full amount will be available to the guest OS. There's nothing worse in a datacenter than having a guest crash because it encounters a non-recoverable disk error because it can't write to a section of the disk that it thinks is there.

As a final note, VMDK files may only be presented to virtual machines as SCSI drives. The only device that has IDE support within a guest operating system—at least on ESX Server—is the CD-ROM drive.

VMFS

Any VMDK file that's created for a virtual machine must reside on a VMFS partition. The VMFS file system is optimized or high I/O. It's a flat file system (which means it may not contain subdirectories) that has a 1MB sector size (by default). Unlike the standard way a Windows Server locks a file system, an ESX Server does not lock the entire partition while communicating with it. The lock instead is placed at a file level which allows multiple ESX hosts to utilize the same SAN space and same VMFS partitions (In order to achieve this functionality a SAN is required.)

Standard files should not be stored on VMFS partitions.

There are two different VMFS configuration options you can configure when building a VMFS partition:

- *Public.* This is the default VMFS mode for VMware and fits most needs for configuring and running virtual machines.

When the file system is configured as “public,” multiple hosts may access files on the same partition. When a virtual machine is running, a lock is placed on the individual file so no other host can utilize it. If the virtual machine is powered off, power can be restored from any host that has access to the VMDK file.

- *Shared.* This mode is used when configuring clustering across two or more physical hosts as it allows more than one virtual machine to access the same VMDK file at the same time (although only one host can have write access at a time—the other(s) would have read-only access). With Shared mode, ESX allows the guest OS to manage the locking of the file system which is why a “Shared” VMFS volume always seems to be locked or “read only” from the COS. The VMDK files stored in a shared VMFS partition can be clustered with nodes on the same ESX host or nodes that expand across several ESX hosts. For the second option, a SAN LUN must be zoned to all ESX hosts that will host a node of the cluster.

Real World VMFS Mode strategy

At this point some of you may be asking, “why not used shared mode for all my LUNs?” The simple answer is overhead. Managing and monitoring which hosts have VMDKs files opened on a shared mode VMFS partition adds overhead to the system.

In the real world the default mode of public is used as a standard for VMFS partitions. Then, when shared mode is needed, a specific “clustering LUN” is created to host the shared mode VMFS partition and the shared VMDK files.

Local Storage

Local storage for ESX is defined as disk drives directly attached to the system by either a SCSI or IDE interface. (Yes, you can use an IDE interface for storage in an ESX system in a limited configuration.) For example, some blade systems do not have SCSI support for hard drives. In these cases it’s possible to install and configure the COS on a local IDE drive. However, it’s important to note that IDE cannot be used for a VMFS file system. So although you can install ESX to IDE, you will not be able to run virtual machines unless you have SCSI or fiber-attached storage.

When building an ESX server for a small environment that may only require one or two servers, local storage may fit the bill for VMFS partitions. (This is certainly much cheaper than a SAN infrastructure.) While a SAN does provide significant benefits over locally attached storage, sometimes these benefits do not outweigh the costs.

Nearly all hardware vendors provide external drive cages that can be loaded up with SCSI hard drives at fairly reasonable prices. This allows for very large amounts of storage in a small amount of rack space. There are several advantages and disadvantages to using local storage.

Advantages of Using Local Storage

- Extremely high speed access rates (dependent on the SCSI controller)
- The entire ESX system is contained in one or two rack devices (the server and disk enclosure), making it easy to move
- Easy to manage

Disadvantages of Using Local Storage

- Limited redundancy
- Disk resources are limited to one host at a time (since virtual machines cannot be clustered across physical hosts in this fashion)
- You cannot use VMware's VMotion product to migrate between hosts

Often smaller environments will find that local storage is the only cost-effective solution available. This is not to say that local storage is any better or worse than SAN storage, it just means that it fits them better. Saying outright that SAN Storage is better than local storage for every ESX environment is like saying a Porsche is better than a Ford truck. The truth is that a Porsche is not better than a Ford truck if you are trying to haul a cubic meter of rocks. If one accomplishes your goals at a price you can afford, then that is the right solution.

SAN Storage

ESX Server has a strong dependency on SAN storage when implemented in an enterprise environment. (Just make sure you check VMware's HCL to make sure your SAN is supported!) Since VMware is used to consolidate machines and all space for a VMDK file is allocated upon creation, you'll need quite a bit of storage. As stated previously, VMware configures partitions for VMDK files as VMFS partitions. Strategies on sizing these partitions (or "LUNs" in the SAN world) will be discussed in Chapter 4. While VMware has the capability to expand a VMFS volume by combining several LUNs together, we strongly recommend against this. Combining LUNs within VMware is equivalent to creating a Volume Set in windows 2000/2003. It does not provide a layer of redundancy or striping, and if there are issues with a path or particular LUN, data corruption is more likely to occur than if the VMFS volume were on a single LUN.

In addition to mapping LUNs and configuring them as VMFS partitions, you can map raw SCSI disks through to a virtual operating system. What this means is you can configure a LUN that has an existing file system, such as NTFS, as a disk on the virtual guest. This is useful in a situation where you are setting up clustering in an active/passive configuration with the active node being a physical server and the passive node being a virtual machine. Like using locally attached storage, there are several advantages and disadvantages to using SAN space in your environment:

Advantages of Using SAN Storage

- Multiple hosts may access the same storage space
- Provide large degree of system redundancy
- Ability to cluster virtual machines across physical hosts
- You can use VMware VMotion

Disadvantages of Using SAN Storage

- High level of management. (Many companies have separate departments responsible for managing enterprise storage)
- Expensive to set up storage infrastructure

- Slightly slower than locally attached storage (depending on SAN configuration)

Other Pluggable Devices

In addition to the core four resources, ESX requires special considerations for “pluggable” hardware components. These components are those that may be easily added to or removed from the physical hardware through either an internal or external connection. While the console operating system may properly detect and utilize many of these devices, a majority of them will not work for the guest operating systems running on the host. This is because special drivers would need to be written and compiled into the VMkernel. Adding additional drivers and support in the VMkernel for a wide variety of available hardware would defeat the purpose of a lightweight kernel with low virtualization overhead.

SCSI

SCSI is one of the resources that’s fairly compatible with both the console operating system and the guests running on the host. The VMkernel provides pass-through support for SCSI devices which can include tape backup devices and physical hard drives. The most common use for raw SCSI access is to attach a tape backup unit to a host to either backup the local files on the service console or to have one of the virtual machines directly accessing the backup device acting as a backup server for the other VMs. How ESX accesses any SCSI device is dependent on how the specific SCSI controller that it is connected to is configured.

Console Operating System SCSI Device Access

As you saw earlier in this chapter, there are three different ways to configure devices for host and guest access (dedicated to the COS, dedicated to the vmkernel, or shared between the two). By choosing to dedicate the SCSI controller to which a specific physical disk or tape device is connected to the service console, you only will be able to access the device through the console operating system. It’s important to remember that the setting chosen is for the entire SCSI controller and not individual devices attached to it. If the controller is dedicated to the service console only, all devices attached to it will only be accessible by the COS. In the event that a configured device is a physical disk, you could configure additional ext3 partitions for additional storage space for the host.

Virtual Guest SCSI Device Access

In addition to configuring SCSI devices for use by the COS, ESX also gives you the opportunity to pass SCSI devices directly through to a guest operating system. Due to limitations of the SCSI architecture, each SCSI device can only be configured on one guest at a time. In order to configure a guest to access a SCSI resource of the host server, the SCSI adapter that the device is connected to must be configured for either Virtual Machine access or shared access. As you will see when we get to installing ESX in Chapter 3, it's recommended that you configure the primary SCSI controller for shared access.

Configuring a guest to use an attached SCSI device can be performed by using the "Add Hardware" button for the target virtual machine. After choosing to attach a "Generic SCSI Device", a list of compatible devices will be listed. Once you select a device from the presented list, it will become available to the virtual machine after the VM is powered back on. Remember, hardware can only be configured on a virtual machine if the VM is powered off. With the current versions of VMware ESX Server, the only devices that are compatible for passthrough to a guest operating system are hard drives and tape devices. Other device types are not supported on virtual machines as "Generic SCSI Devices".

We feel it is quite important to note that up to ESX 2.1.2 there have been reports of SCSI errors and locking occurring when utilizing a tape drive on the primary SCSI device. It is strongly recommended that if a tape device is required that a second SCSI controller be utilized. This serves two purposes. First, this will enhance the overall throughput of the system. If the second controller is installed on a different system BUS than the original controller there will not be any I/O contention on the system. The second reason for this is that if a SCSI lock does occur it will not impact the system drive. If the controller were to "lock" on the first SCSI controller you would be looking at a purple screen issue and downtime to recover the host and all guests. The I/O compatibility guide should also be consulted prior to making any decisions on adding additional controllers.

iSCSI

One of the most requested features for ESX is the integration of iSCSI for connections to external storage arrays. VMware has been very tight lipped up to this point on anything relating to this functionality. We can verify that this functionality is on VMware's roadmap, but it is still not ready for implementation. ESX 2.5 does not contain any additional code to support this. Our understanding is that in order for ESX to support iSCSI a TCP/IP stack would need to be integrated into the VMkernel. This would more than double the size of the VMkernel in its current state. One of the main selling points to ESX is its lightweight kernel that does not utilize large amounts of CPU cycles.

Chapter 5 has a great description on kernel mode processor calls, we'll summarize the issue here. The amount of TCP/IP calls that would be required for this functionality would also increase the amount of kernel mode calls the VMkernel is required to handle. The VMkernel would have to perform significant amounts of conversion from the guest through to the physical hardware. The performance hit from this does not make iSCSI a viable option at this point in time.

PCI

PCI devices within ESX have extremely limited support. By reviewing the ESX I/O Compatibility Guide, you'll notice that the only devices listed are network, SCSI and Fiber adapters. One of the most commonly sought after PCI cards, Brooktrout Fax cards, is not supported. The main reason for this is driver support. Not only would a Linux level driver be required, but custom VMkernel drivers would also need to be written to properly allow guest access to the resource. This is not to say that other PCI cards will not work. PCI cards that provide additional parallel or serial ports have been reported to work, and work quite well at that. Before choosing a card it is extremely important that VMware support be contacted to verify supportability and functionality of the card in question.

USB/Firewire

While ESX does support USB devices, it only does so at the COS level. Currently, there is no functionality built in to provide USB pass-through to a guest operating system. For this functionality there are several USB network devices that have received excellent reviews in the VMware community and are compatible with guest operating systems. There is no firewire support at the COS or VMkernel level.

Parallel/Serial

Virtual machines running within ESX can be allowed to have direct access to parallel and serial ports of the host operating system. There are several limitations to be aware of before attempting to configure one of these mappings. First, only one guest is allowed to access a single port at any given time. Each guest may have access to one parallel port and two serial ports. Second, there is no way to configure these devices using the MUI. The only way to properly configure them is to manually modify configuration files with the COS. VMware provides this support for external serial modems and external parallel dongles that are required for certain applications.

Configuring Parallel Ports

Parallel ports, as you will see, are more difficult to configure than serial ports. This is because you must first configure the COS to properly see the parallel port as it is not enabled by default. Before attempting to configure parallel pass-through you must ensure that the port is properly configured in the system BIOS. Make sure the parallel port mode in the BIOS is set to either PS/2 or bi-directional. These are typically the default settings within a majority of the systems. Once complete you must add three lines to the end of your /etc/rc.d/rc.local file:

```
/sbin/insmod parport
/sbin/insmod parport_pc
```

```
    /sbin/insmod ppdev
```

These lines will properly load the drivers into the COS that are required to utilize the parallel port. These settings will take effect on the next reboot. You can force them to take effect immediately by issuing each of the three commands at the command line.
Once the drivers are properly loaded (you can verify with the "lsmod" command) you will be able to modify the VMX file of the guest that needs to access the parallel port. Shut down the guest and add the following lines to its VMX file:

```
    parallel0.present = "true"
    parallel0.fileName = "/dev/parport0"
    parallel0.bidirectional = "true"
    parallel0.startConnected = "true"
```

When the system is powered on, it will have proper access to the system's parallel port.

Configuring Serial Ports

Configuring the virtual machine to use a serial port is very similar to configuring access to a parallel port. The major difference is the COS is already properly configured to hand over access. With the proper virtual machine powered off, add the following lines to its VMX file.

```
    serial0.present = "true"
    serial0.fileType = "device"
    serial0.fileName = "/dev/ttyS0"
    serial0.startConnected = "true"
```

When the machine is powered on, the newly mapped serial port will be presented as COM1 to the guest operating system. In order to configure a second serial port to a guest, use the same lines as above while swapping out "serial0" with "serial1". The second port will be recognized as COM2 on the guest operating system.

Resource Sharing

The greatest benefit of using ESX over any other virtualization product is its capability to dynamically share system resources. Since the VMkernel runs on top of and controls the hardware, we're given high levels of flexibility to share system resources across multiple guests. VMware also provides us with several ways to configure sharing of resources giving us an extremely dynamic and flexible platform. Here we'll discuss the various methods that ESX has that allow us to maximize system utilization of the core four resources.

Processor

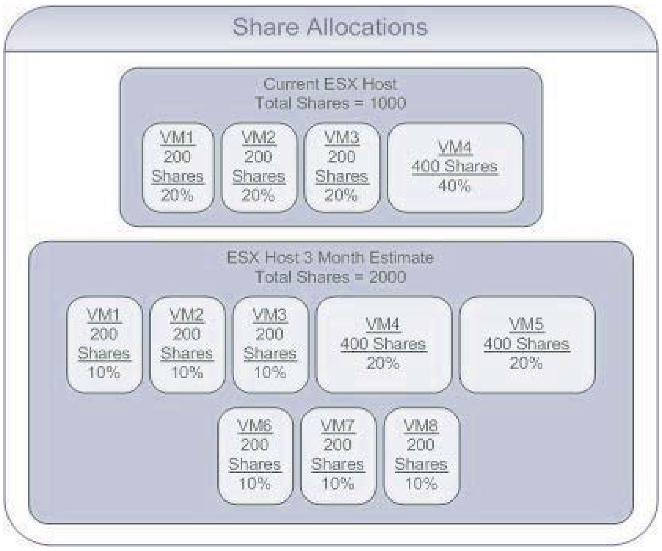
The VMkernel was designed to provide a high level of interaction with the processors, allowing ESX to dynamically shift resources on the fly (and invisibly) to running virtual machines. If you have three machines that are running in an idle state on the same host as a document processing server, ESX will temporarily shift resources to the highly utilized server to accommodate its immediate needs. If a process is spawned on a low utilization server, the necessary resources are returned to the original VM to effectively run the new application. Generally, a good rule of thumb is to allocate four virtual processors per physical processor, although we've worked in some environments where seeing 5-6 virtual processors per physical was not out of the question based on the processing needs of the virtualized machines. ESX does have a hard coded limit of 80 virtual processors that may be assigned within any single host. With larger systems such as 8 or 16-way hosts, this limit should be considered during the design process. This can be broken down any number of ways mixing single and dual processor virtual machines.

If further granularity is required, ESX provides several mechanisms for manually adjusting processor allocation. This may be useful when one system requires a higher priority than others, such as a database server that is handling transactions for several different applications on the same host.

Processor Share Allocation

One of the easiest ways to modify the default processor allocations of a virtual guest within ESX is to utilize shares. Shares are a mechanism to allocate resources relative to all virtual machines running within a specific host and are used in several instances. Using this method, you can assign priority to specific guests when the host becomes limited on processor cycles. As you add more virtual servers to a host, the total number of shares goes up, and the percentage of total shares to a particular guest goes down. A server that has 1000 shares will receive twice the priority when assigning CPU cycles as a host with 500 shares. The downside to this method is that with each new virtual guest created, the allocation to existing machines decreases, which will slightly decrease their performance when the host system is under a heavy load.

Figure 2.5



A very important fact to note is that share values do not come into play within ESX until there is sufficient load on the system. Until all available resources are utilized, the VMkernel is happy assigning as many cycles to running guests as they request. Since one of the benefits to ESX server is to maximize hardware, it's unlikely that this will ever be the case, but we've seen hosts that are never utilized to the point that share allocations are required. This does not mean there is no benefit to properly assigning priority, as a runaway guest still has the potential to devour an entire host processor. This could be devastating in a dual or even quad-processor configuration.

Specifying Min/Max Percentages

Within ESX you may assign a minimum and/or maximum percentage value for the processing resources of a virtual machine. By setting the minimum percentage, you're telling VMware to never allow that particular virtual machine to drop below the assigned percent of a single host processor. This is useful if a processor-intensive application requires a bare minimum amount of resources to effectively run. When assigning these values, you need to be extremely careful not to over allocate resources. If you assign eight virtual machines to run with a minimum of 30% in a dual processor server, you will only be allowed to turn on six machines, as three servers per processor will utilize 90% of the resources available on each.

The maximum value is the exact opposite as the minimum. It provides a hard value that a particular virtual machine may exceed. This would be used in a situation in which an application is known for taking as many resources as the server can throw at it. By setting this value, you tell ESX to allocate a maximum percentage of one CPU to the virtual machine which would provide protection from impacting other virtual machines running on the same host. If you're using the virtual SMP option you can assign a value of up to 200%—the combination of the max percentage per assigned CPU. For example, a value of 140% would mean that each CPU assigned to the virtual machine may be utilized up to 70%.

Assigning a maximum value to an unstable guest operating system would prevent it from consuming an entire host processor. For development guests that are being used to test new code, this configuration could prove to be vital in maintaining a stable environment, especially if multiple developers have guests on the same physical host. Work from one developer will not interfere with work from another.

Setting minimum or maximum processor values are independent of one another. You can choose to set only a minimum value, a maximum value, or both values. Not setting any values allows VMware to have full control over the processor allocation of your virtual machines. While this is sufficient for small deployments with a low number of guest operating systems, it may be crucial to manually change these values as the number of guests increase on a host.

Combination of Min/Max and Share Allocation

When brief periods of contention are expected, one of the best ways to configure the resource allocation of your guests is to use a combination of share values and Min/Max specifications. Before we dive into this theory, we must recommend that very careful planning and utilization analysis be performed before changing any values, as improper settings may impact the integrity of the system. As an example consider Figure 2.6:

Figure 2.6

Guest	Minimum %	Maximum %	CPU Shares
1	15	35	2000
2	40	60	500
3	30	40	1000

Even though Guest 1 has a lower Minimum percentage, he will receive priority of idle shares at a ratio of 4:1 over Guest 2 and 2:1 over Guest 3. This scenario is useful if Guest 1 has a greater benefit than Guest 2 or 3 when additional idle shares are available. With Guest 2 and 3 having a higher minimum CPU percentage, the available cycles will provide enhanced performance on Guest 1 during times of stress.

Until there is processor contention, the systems will run at their maximum allocations. Once resources become limited, ESX will begin to remove cycles from the guests with the lowest share values to best balance them based on the custom share configurations. Based on the chart above, Guest 1 would be the last guest impacted in this scenario due to it having the lowest maximum CPU percentage and highest amount of shares.

We cannot stress enough that modifying these settings should only be done after closely analyzing your system to determine the utilization trends of each virtual machine. While we show an extremely simplistic model with 3 virtual machines, this type of resource management is extremely difficult as the number of guests increase. You should have a solid idea of what impact modifying the settings for a single virtual machine will have on the host and all remaining virtual machines.

Affinity

In addition to setting processing thresholds using one of the previously described methods, you can also specify which physical processor(s) the virtual machine can use. This gives you complete control of the virtual machine processor usage. Not only can you specify that a virtual machine is guaranteed a minimum percentage of processor usage and has a high level of priority in receiving additional allocations, you can also specify the exact processor that provides the resources. Using this methodology one would have complete control over how a production infrastructure reacts on a box, ensuring that every virtual machine has the proper resources and does not step on the toes of other critical applications.

Processor affinity is not limited to specifying a single processor. By specifying a group of processors, you can tell ESX that it is allowed to allocate resources only from the selected processors, leaving the remaining processors inaccessible to the virtual machine. Application servers such as Citrix servers can be pinned to different processors to prevent kernel level calls from interfering with each other. Support servers may then be pinned to the remaining processors and be allowed to share those resources. This allows for a layer of isolation for resource allocations. Typically as shares are granted and removed from the entire pool of processors, every guest on the host is impacted. By utilizing processor affinity, the support servers may dynamically adjust their processing resources while the Citrix servers react as if business is normal.

Memory

Like processing resources, ESX has the capability to dynamically optimize utilization by several techniques: transparent page sharing, ballooning and swapping. These various techniques allow memory of a host to be over allocated, meaning that you can assign more memory to virtual machines than the physical host contains. While we recommend only using sharing and ballooning for production systems, swapping can be utilized to further maximize development hosts and allow more guests. Using the following techniques, it's not uncommon to achieve over allocation of 20-30% in a production environment and up to 100% in development.

Transparent Page Sharing

When a guest operating system is loaded, there are many pages in the memory space that are static and that contain common pages found on all similar operating systems. The same can be said about applications that run on the operating systems. The transparent page tables provide a mechanism to share this space among several virtual operating systems. By mapping identical virtual page numbers back to physical page numbers, guests that are using identical space in the machine page space can share these resources. This lets the system free up memory resources for over allocation without impacting any guests. This is the only memory allocation method that takes place without the host running at maximum memory use.

Idle Memory Tax

When memory share allocation takes effect, VMware provides a mechanism to prevent virtual machines from hoarding memory it may not be utilizing. Just because a particular server has four times the memory share priority than another does not mean it requires it at the time allocation takes place. VMware has a process that applies an idle memory tax. This associates a higher "cost value" to unused allocated shares than it does to memory that is actively used within a virtual machine. This allows the virtual machine to release it for use on other guests that may require it. If the virtual machine in question has a need for the memory, it still has the proper authority to reclaim it as it still has priority over the memory space. A default value of 75% of idle memory may be reclaimed by the tax. This rate may be adjusted by modifying the "MemIdleTax" advanced option. The polling rate of the memory tax may also be adjusted by adjusting the "MemSamplePeriod" advanced option which is set to 30 seconds by default. These values are configured in the advanced options of the ESX host.

Ballooning

VMware has provided functionality to let the guest operating system decide what memory it wants to give back when the system is plagued by high utilization. This is made possible by using a memory reclamation process called "ballooning." Ballooning consists of a vmmemctl driver on the virtual operating system that communicates with the vmkernel. What this driver does is emulate an increase or decrease in memory pressure on the guest operating system and forces it to place memory pages into its local swap file. This driver differs from the VMware swap file method as it forces the operating system to determine what memory it wishes to page. Once the memory is paged locally on the guest operating system, the free physical pages of memory may be reallocated to other guests. As the ESX hosts sees that memory demand has been reduced, it will instruct vmmemctl to "deflate" the balloon and reduce pressure on the guest OS to page memory.

If the vmmemctl driver is not installed or running on the guest, the standard VMware swap file method is utilized. The vmmemctl driver is the preferred method of memory collection as the guest operating system gets to call its own shots. The amount of memory reclaimed from a guest may be configured by modifying the "sched.mem.maxmemctl" advanced option.

Paging

ESX has its own application swap file. This file is independent of both the console operating system and page files setup within the virtual guest operating systems. VMware recommends that this swap file capacity be set to the total amount of memory that will be allocated to all virtual machines. This allows up to a 100% over allocation of memory resources using paging. This is not recommended though, as paging large amounts of data requires additional CPU resources and tends to have a negative impact on the host. When an ESX system becomes over allocated, it takes guest memory that is infrequently used and stores it in the swap file. If VMware requires the resources, it retrieves the memory from the swap space and brings it back into local memory. It may be useful in the development environment, where paging will have less of an impact on the solution, but should be avoided at all costs on a production host.

Network

Each virtual machine configured in an ESX environment shares the combined bandwidth of a virtual switch. In its default configuration, ESX provides limited load-balancing across each physical connection of a virtual switch. If there is only one physical adapter that makes up a virtual switch then you won't gain any benefit from load balancing. The reason we called VMware's default load balancing mechanism "limited" is because of the way it actually functions. We'll go into more detail in Chapter 4, but will briefly summarize here: Each virtual NIC that's assigned to a virtual machine is assigned a virtual MAC address. When a virtual NIC is activated on the guest, the VMkernel ARPs the virtual MAC address for that adapter down one of the physical paths of the virtual switch. The algorithm used for this does not take network load into consideration so it is possible that one path of a virtual switch may become heavily loaded. This behavior limits the sharing benefit gained from load balancing.

ESX provides a single method for throttling network bandwidth. While this may seem limited, the tool used is actually quite powerful. VMware provides the nfshaper module, which allows the VMkernel to control the outgoing bandwidth on a per guest basis. This module only limits outgoing traffic since there is no way (at the host level) to configure how much bandwidth is received by the system. This module is not enabled on a global basis, so must be configured for each virtual machine that you wish to throttle. It's extremely useful if there are a limited number of physical adapters in a system configured with low bandwidth uplinks. A server that's network intensive that will utilize all resources given to it can be limited to allow multiple guests to have access to the networking resources they need.

The nfshaper module can be configured by utilizing the MUI. After logging into the MUI and opening the properties for the target guest, navigate to the "Network" tab. The machine may be in any power state to perform this action. In the right column, the "Edit" button will be active. Clicking on it will present several options.

- *Enable Traffic Shaping.* This one is pretty much a no-brainer. This enables and disables the nfshaper module.
- *Average Bandwidth.* This setting is the sustained throughput that you would like the nfshaper module to maintain for the guest. Remember, this is only for outbound bandwidth. There are no controls to limit how much information a guest receives.
- *Peak Bandwidth.* This is the maximum amount of throughput allowed by the nfshaper module. A guest may hit its peak bandwidth to assist in properly processing all data that needs to be sent. Peak bandwidth is often configured to double the value of average bandwidth. This allows the system to properly send all packets when the system comes under load.
- *Burst Size.* The amount of data that the system may send while hitting its peak bandwidth. If the burst amount is hit, the nfshaper driver will drop below the peak bandwidth until the next burst cycle is available. The burst size should be configured as a byte value of 15% of the average bandwidth. This should be more than enough data to properly fill the peak bandwidth rate.

If the application or guest operating system being filtered starts to display errors such as dropped connections or failed ICMP requests, you should consider increasing these values accordingly.

Disk

Like processor and memory resources, the VMkernel applies a share-based allocation policy to disk access. (Note that we're talking about disk access here, not disk space usage.) The way ESX determines disk access resources is by tracking what VMware calls "consumption units." Each SCSI command issued to a disk resource uses one consumption unit. Depending on the amount of data transferred in the request, additional consumption units are allocated to the guest. When disk resources become stressed on the host system, the share values will determine how many consumption units a guest can utilizes at a given time. After share allocation kicks in, it acts no differently that it does for CPU or memory resources, so we don't need to discuss it again. When choosing the default settings provided by VMware, the following values are used: Low = 500, Normal = 1000, and High = 2000. There are several options available that we'll discuss in Chapter 4 that maximize the utilization of disk resources and yield the best performance available.

Summary

If you managed to get this far then you have more than enough information to build and configure your own ESX server. By understanding the various components of ESX and how they operate, you're well on your way to managing an ESX host. In the next chapter we'll take this information and apply it to the configuration of your first ESX server. By knowing the information presented in this chapter you'll have a good understanding of the various options presented to you during the installation process and why we make the recommendations that we do. Whether you're planning an environment of 2 hosts or 50, the information in this chapter will act as a reference baseline for the rest of this book.