VMware® Infrastructure 3

Advanced Technical Design Guide

~and~

Advanced Operations Guide

Two books in one!

Ron Oglesby
Scott Herold
Mike Laverick

Chapter 2 – Virtual Infrastructure Architecture

VMware is using a different marketing approach this time around with the newest release of their enterprise virtualization platform. With the previous version, each component of the virtual infrastructure was labeled and sold as a separate SKU. This caused quite a bit of confusion to end users trying to build a large-scale infrastructure with exactly what licensing components needed to be purchased to enable all of the features that were required.

To combat this, VMware took a new approach of combining the most used components into a single bundle called VI3, or "VMware Infrastructure 3". We have run into a great number of people that still insist this stands for "Virtual Infrastructure 3", which is definitely not the case[1]. VI3 comes in three different flavors, each of which contains a different level of functionality through enabled components and features. To start things off we want to discuss the available components and features that make up the VI3 suite. Much of this information will not be new to many readers of this book, but you never know, you may learn something.

There is actually a good chance that you will not read the term VI3 outside of the title of this book and this chapter. VI3 has many components that while they do interact with each other they, for the most part, are configured and managed as standalone components that fall well outside the generic umbrella of VI3.

ESX Server

ESX Server 3.0 is the single piece that is absolutely required to run a virtual infrastructure in an environment. The ESX Server component is the virtualiza-

[1] Even though it would have made a lot more sense since we feel it is quite foolish to include your company name in an acronym that makes up a product name.

tion software that enables you to actually run virtual machines. It is also the single most complex piece of the VI3 virtualization platform, and as such, will receive a majority of the attention throughout this entire book. We will go into more detail than most individuals will ever need to know about ESX Server throughout the rest of this chapter.

Virtual SMP

Virtual SMP is an add-on module for VI3 that provides the capability to configure multi-processor virtual machines. It is important to point out that you do not need Virtual SMP to use ESX Server itself on a multi-processor host; you only need it to create VMs that use multiple physical host processors. VMware has bumped up the number of processors that can be assigned to a single guest operating system from two to four in VI3. Virtual SMP will be discussed in more detail later in this chapter when we discuss the core 4 resources in depth.

VirtualCenter

VirtualCenter is an integral part of any virtual infrastructure that consists of more than a handful of ESX Servers. The server component of VirtualCenter is not included in the VI3 pricing structure, but VirtualCenter Agents, which are required to communicate with the management server, are. The VirtualCenter Management Server runs on a standalone Windows host and contains a database backend to store configuration and performance data. The VirtualCenter Agents are included with ESX 3 that provide the required communication channel between the ESX hosts and the VirtualCenter Management Server.

As previously mentioned, every VirtualCenter Management Server will require its own license that is independent of the standard VI3 licensing. The good news is that in a vast majority of the VI3 implementations a single management server is more than enough to manage and monitor the entire environment. Many of the advanced features provided by VI3 actually require the use of VirtualCenter. When we discuss the features that VI3 provides we will make sure we indicate which ones do have such a requirement.

The VirtualCenter component is such an important part to a VI3 implementation we will have several chapters in this book that are dedicated to functionality

features that are only available if VirtualCenter exists. In addition, a majority of the Administration Book is focused on configuration and management of the entire infrastructure using VirtualCenter components. Chapter 4 is the first chapter that is dedicated to the configuration and use of VirtualCenter.

A High Level Look at the VI3 Features

If you are reading this book there is probably an extremely good chance that you are either using VI3 or you have made a fairly solid decision that VMware will be your platform of choice for your virtual infrastructure. Knowing that, we want to simply highlight the advanced VI3 features that you are probably well aware of in this chapter. Throughout the rest of the book we will discuss the best way to use each of these technologies for a variety of situations. If you notice terms that are unfamiliar to you, please remember this is just a quick primer. Everything will be fully explained in their respective chapters where we discuss them in detail.

VMware VMotion

To this day VMotion remains one of the greatest technologies in virtualization and is probably one of the biggest drivers towards server virtualization. VMotion provides the capability for virtual machines running on a shared storage architecture to have their hardware resources shifted from one host server to another in real time with no impact to the guest operating system. This single technology provides unparalleled levels of availability and allows for hardware and virtualization software management of an entire virtual infrastructure without having to shut down a single virtual machine. Figure 1 shows a simplistic view of how VMotion functions in a typical VI3 environment.

Figure 2- 1: VMotion

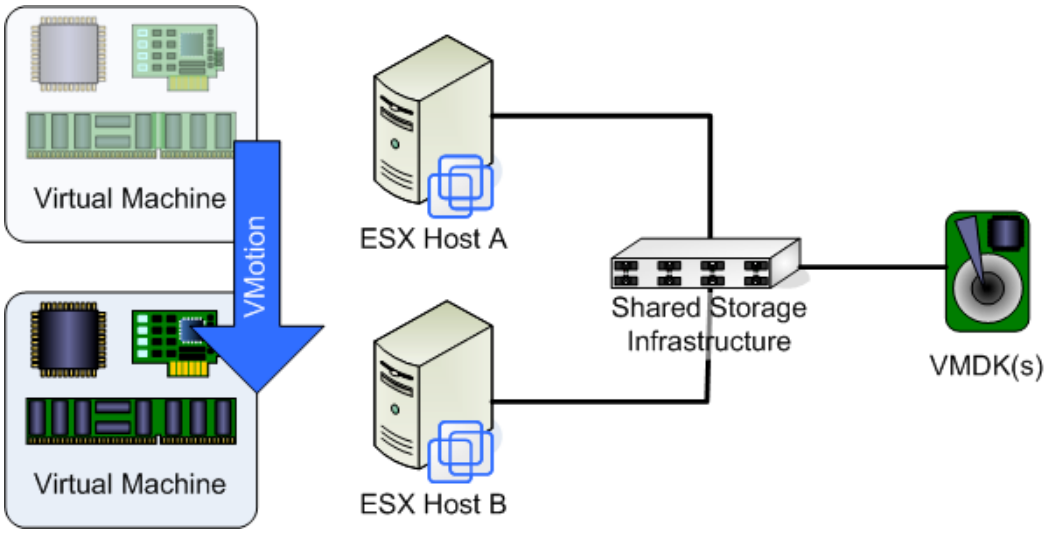


Figure 1 is deserving of a brief explanation as to what is going on. To start, a virtual machine is running on ESX Host A. The configuration files and data for this virtual machine are stored centrally on a shared storage infrastructure through one of several mechanisms that will be discussed in Chapter 5. Because both ESX Host A and B can see this same storage, the virtual machine can technically run on either system. The magic of VMotion comes when ESX Host A can move the processor, memory and network workloads to ESX Host B without interruption to the migrated workloads and while remaining invisible to the guest operating system. It is important to note that VirtualCenter is a requirement in order to leverage VMotion for your virtual machines.

A VMotion migration is actually performed with only a few seemingly simple steps. These steps are made possible solely due to the fact that the VMkernel (which we will learn about a little later in this chapter) provides an abstraction layer between the virtual machine and actual physical hardware. Even though the physical hardware platform is changing, the VMkernel makes sure the virtual machine doesn't notice anything is changing physically.

The memory state of the virtual machine is "snapshotted" on the source host. This allows the source host to first send the memory state of the virtual machine to the destination host.

After the initial memory transfer is complete the virtual machine configuration settings are sent from the source host to the destination.

The virtual machine settings are checked for availability at the destination. Enough CPU and Memory resources must be available. In addition, the destination host must be able to see the VMDK file(s) used for virtual machine data and must also have the same virtual switches configured to ensure network connectivity is maintained after the migration. If any of these checks fail, the VMotion process exits and returns to normal operation on the source host.

The final memory state is transferred from the source host to the destination host. This makes sure that any changes that occurred while the initial memory state was copied and configuration settings were verified are updated at the destination host.

Control of the virtual machine is taken by the destination host. This includes transferring the SCSI reservation for the VMDK file(s) and sending an arp request from the virtual machines NIC's to dictate the MAC address has switched network ports. This dictates the end of the process and the VMotion migration is complete.

There are several key uses for VMotion in an infrastructure that opens possibilities that aren't available in a physical infrastructure. One of the most obvious uses is for ESX host maintenance. Although most physical systems, especially the ones used for VMware ESX infrastructures, are built with many redundant components. A failure of any one of these components is not a major issue, but at the same time, should not be ignored. By using VMotion, the virtual machines running on the host requiring maintenance can be moved to alternate hosts in the infrastructure. This capability is not limited to hardware maintenance. VMware has been known to release occasional patches that require a reboot of the ESX host to take effect. Before beginning the patch process for a host the running virtual machines can be easily migrated to alternate locations using VMotion.

Another function that commonly leverages VMotion technology is balancing resource utilization across ESX hosts. As a virtual infrastructure grows both in the number of virtual machines hosted and in resource utilization within existing virtual machines overall utilization across multiple hosts in an infrastructure become unbalanced. It is not an uncommon practice to go through and reanalyze resource utilization on a monthly basis and make adjustments by shifting virtual machines across the infrastructure. VMotion can be leveraged to remove any downtime that would normally be associated with this activity.

There are several challenges with manually using VMotion for the mentioned tasks. In the event that an ESX host requires maintenance you need to know where you can safely VMotion your virtual machines to. Without proper planning it is quite possible that a host can be overloaded, which would have a negative performance impact on the virtual machines running on that host. This leads into another major challenge that also exists in the workload balancing scenario which is the fact that the VMotion process is very manual. The amount of change in some environments also requires that resources be closely watched on an ongoing basis. Failure to do so can lead to negative virtual machine performance. VI3 can simplify these uses of VMotion through a technology called Distributed Resource Scheduling, or simply DRS.

VMware DRS

VMware DRS technology takes VMotion to the next level. By leveraging VirtualCenter, DRS has the capability to make recommendations or, depending on the setting, automatically VMotion your virtual machines to balance the workload evenly across multiple ESX Hosts. By organizing groups of virtual machines into Resource Pools, you can ensure that these specific groups of servers have access to the resources they need when they need them, regardless of which host is providing these resources. The configuration possibilities of DRS are quite advanced and will be discussed in full detail in Chapter 8. Many people fear letting VMware make these decisions and there are some minor flaws with some of the logic behind the process, but overall this innovative technology is yet another thing that sets VMware apart from the rest of the virtualization world.

Figure 2- 2: DRS

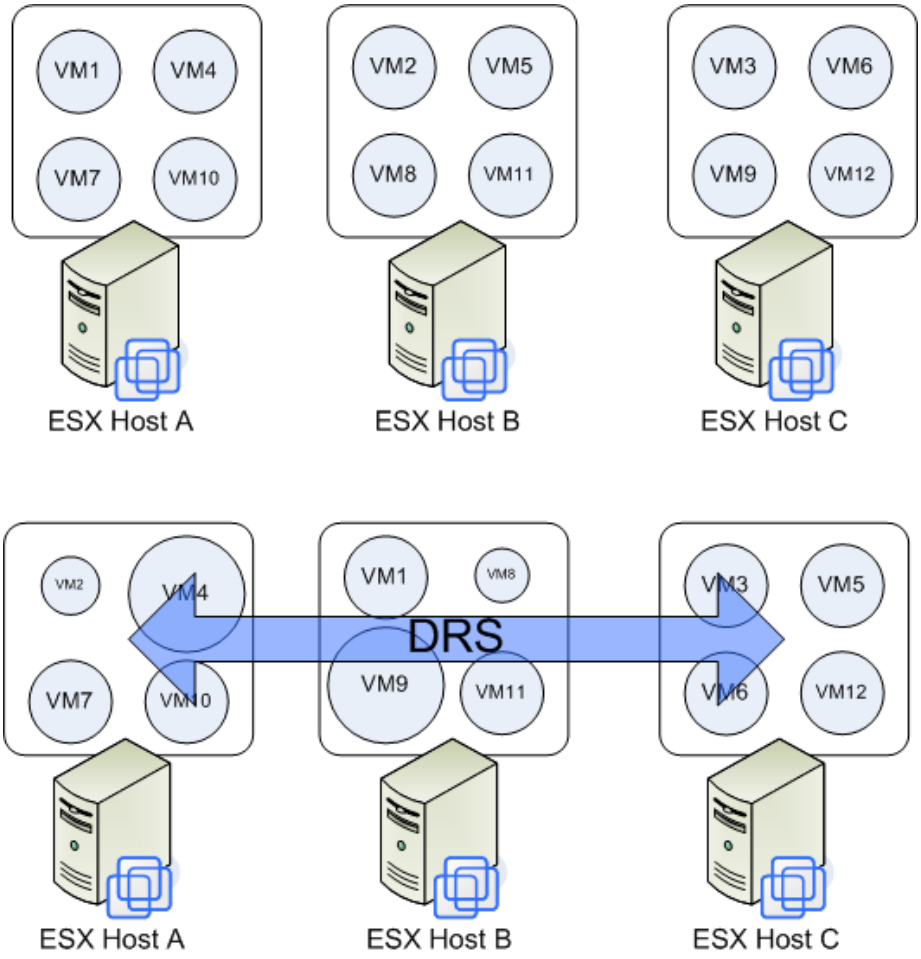


Figure 2-2 shows a high level example of what DRS is capable of. The top row shows a group of 12 virtual machines, all using the same amount of resources. Knowing this almost never happens in the real world; the bottom row of hosts shows how specific virtual machines leveraging more resources (VM4 and VM9) are combined with servers using fewer resources (VM2 and VM8). While this is a simplistic view, we can start to immediately the benefits of what DRS can provide on a large scale.

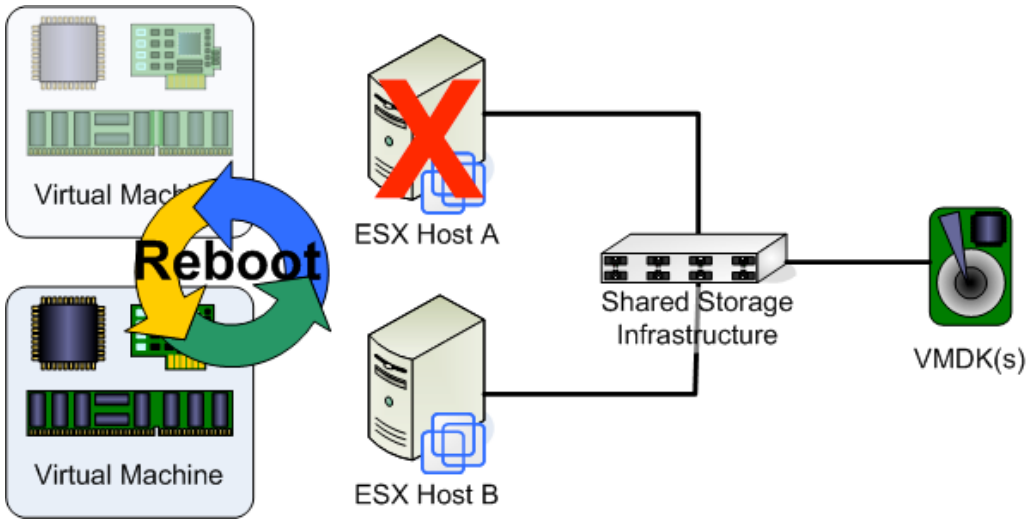
In order to use VMotion and DRS both a source and target ESX server must be online and available, as the migration process is performed without guest impact. In order to provide resource availability to virtual machines in the event of a host failure, VMware has provided VMware HA.

VMware HA

VMware HA, or High Availability, provides automatic protection of virtual machines in the event that a host failure occurs. The recovery time in the event of host failure depends on the amount of time it takes the effected virtual machines to reboot on alternate ESX hosts. Many advanced application architectures have internal dependencies that dictate one system must be running before others can function. HA meets this requirement by allowing users to specify priority to each virtual machine configured in a host cluster. In the event of a host failure VMware HA will do its best to power everything back on properly without requiring end user interaction.

There are two key requirements to enable VMware HA in an infrastructure. First, VirtualCenter must be managing all hosts involved in the high availability cluster. The only way to have this level of control over a virtual infrastructure is through the use of this centralized management console. The most important requirement for VMware HA is leveraging a centralized shared storage infrastructure. Much like VMotion, every host needs to have access to the virtual machine configuration and data files to properly boot it in the event of a hardware failure. **Figure 2-3** will give you a rough idea of how VMware HA functions within an environment.

Figure 2- 3: VMware HA



It is extremely important to note that using VMware HA will not allow your infrastructure to run uninterrupted. If a host failure occurs, each virtual ma-

chine running on that host will need to be booted on an alternate ESX Server. A small amount of downtime will be noticed for each virtual machine while it reboots. This can cause issues with application architectures that are dependent on the component that is rebooting. As an example, if a Database virtual machine has to reboot due to a host failure additional systems may need to be rebooted as a result, even though they are running on a host that hasn't experienced any issues. VMware HA does not address this issue so external processes will need to exist to resolve potential issues.

In addition, additional resource capacity to run the virtual machines must be available within the infrastructure. We will discuss properly building an N+1 virtual infrastructure in Chapter 3. If VMware HA is being used in conjunction with VMware DRS, VirtualCenter will actually have the capability to look at each existing host in the host cluster to determine the best place to recover each virtual machine based on current resource utilization. Using VMware HA on a centralized storage infrastructure can help protect against a local hardware failure, but what happens when there is data loss on the storage infrastructure, or worse yet, the entire storage infrastructure becomes unavailable. It is critical to have a solid backup recovery/disaster recovery plan and the next component of VI3 addresses just that.

VMware Consolidated Backup

The final add-on component for VI3 that we are going to mention here is VMware Consolidated Backup, or more commonly known as simply VCB. A major challenge with a virtual infrastructure has been backing up the environment in a specific backup window. While we will go into more details of all of the available backup options in Chapter 11, we feel it is important to mention the basics of VCB here.

VCB is a separate component that must be installed on its own physical Windows 2003 server, and no, it cannot be the same system as your VirtualCenter Management Server. When configuring shared SAN storage for use within a virtual infrastructure, the same LUNs must be made available to the VCB "Proxy" Server. With the proper SAN configuration VCB can be run in one of two ways. First, it can mount the VMDK files of the virtual machines as drive letters to the proxy server. The individual files on the VMDK can then be accessed by a typical tape backup agent and backed like any other file on the VCB proxy server. The second method that can be leveraged with VCB is as an im-

age level backup. This mode will allow VMDK files to be backed up as image files, typical to how backups are managed using existing third party backup tools.

Figure 2- 4: DRS

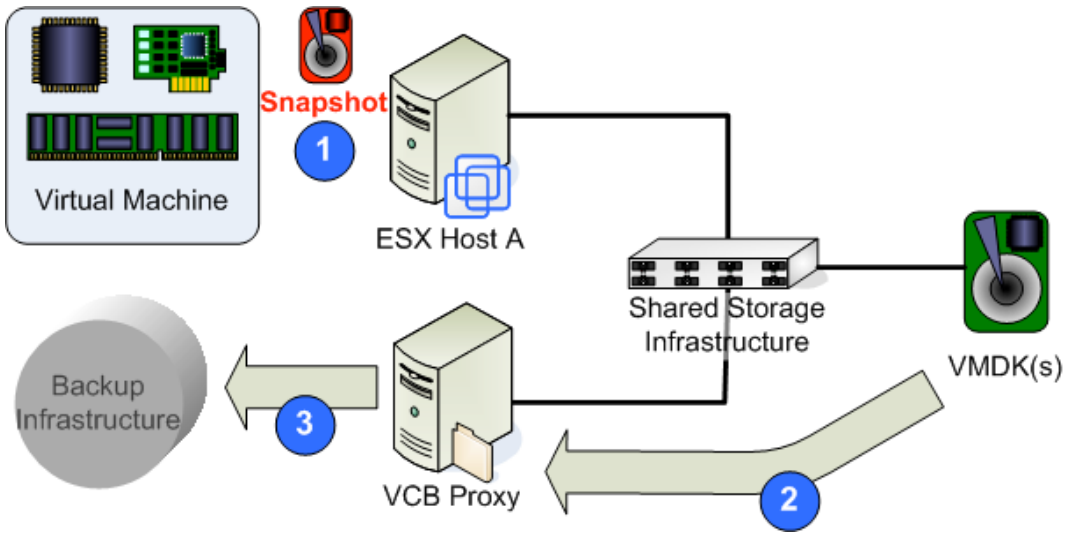


Figure 2-4 highlights the basic steps involved in a VCB backup job.

A snapshot of the virtual machine is taken. This unlocks the VMDK file(s) for use on another system.

The VMDK file(s) is mounted to a VCB Proxy server as a drive letter or a full image export is performed.

A backup server grabs the individual files or the exported image and moves it off to short or long term storage in the enterprise backup infrastructure.

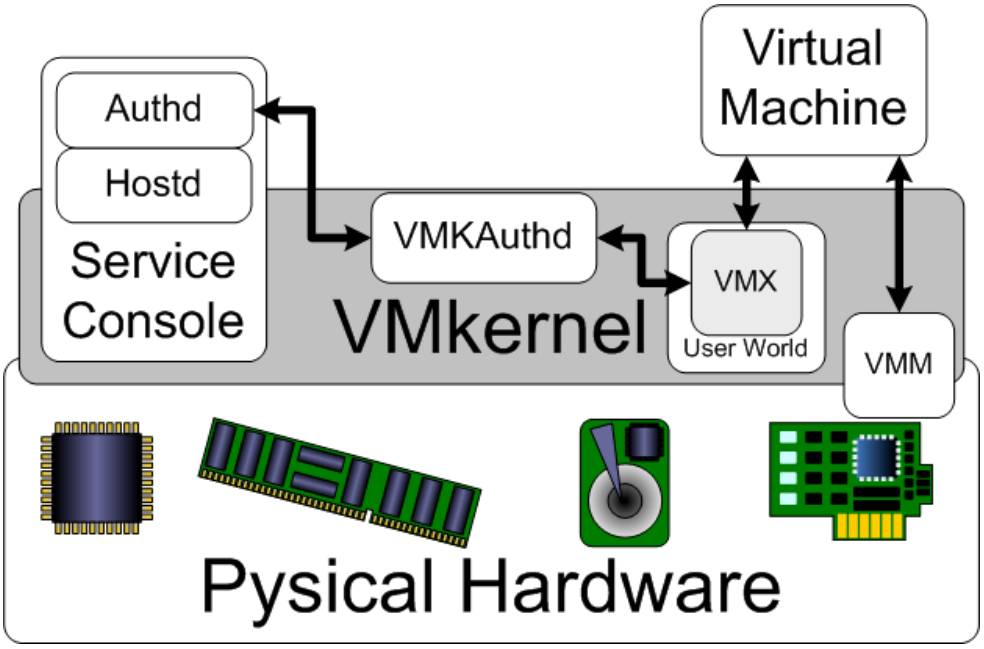
The main advantage of using VCB over one of the available third party tools is that no host resources are used while backing up the infrastructure. While this may seem like the ultimate solution, there are quite a few intricacies in planning for VCB. Please make sure you blast through Chapter 11, where we will discuss this and other solutions in painstaking detail, before deciding that this is the only backup solution for you.

Now that we have described all of the things VI3 is capable of providing it is time to start digging into the details of how it works. The rest of this chapter will focus strictly on ESX Server 3. The other functionality will be fully described in the remaining chapters of this book.

ESX Server 3 Architecture

Now that a majority of the typical sales and marketing information that most people reading this book aren't interested in is out of the way, it's time to truly dig into the architecture of ESX Server 3 to see what really makes it tick. VMware ESX Server 3 is an advanced operating system with an intricate architecture. While we will not go into details of every last aspect of the internal workings, a feat that would be an entire book upon itself, we will go over the common components that are interacted with most commonly. Figure 2-5 shows a 10,000 foot view of the primary architectural components that we will discuss in this section.

Figure 2- 5: ESX Architecture



Service Console

The service console of an ESX server is commonly referred to as the Console Operating System, or COS. VMware has put forth an impressive effort to deemphasize the use of a COS in the new ESX 3 architecture and remove all references of the term from people's memory. Their long term goal is to eventually make it completely disappear entirely. To please VMware and help them in their efforts we will make every attempt to use the currently proper term "Service Console" in this book. In the event we miss an instance or two, Service Console should be considered synonymous with Console Operating System.

The Service Console is a Red Hat Enterprise Linux 3 (Update 6) based Linux distribution that is heavily modified and stripped down to provide a lightweight management interface of the VMkernel to the end user. A primary function of the Service Console is to bootstrap and turn over full control of all hardware resources to the VMkernel. The service console is assigned a default value of 272 MB of physical host memory to execute processes and manage the small amount of physical hardware that the VMkernel doesn't manage. It runs as a privileged virtual machine inside the VMkernel and is subject to resource allocation and scheduling alongside other virtual machines running on the ESX host. Overall, the Service Console is responsible for quite a few things that are vital to the proper operation of ESX.

User Interaction with ESX – The Service Console is responsible for managing the various methods to communicate with the ESX host. In order for an end user to interact directly with the VMkernel the Service Console must be used in some fashion. Several services are run in the Service Console that allow user interaction with the host using various methods such as:

- Direct Console Access
- SSH Access to the Console
- Web Management Interface (Which is also responsible for providing the SDK programmatic interface to ESX Hosts)
- Proprietary Communication Methods from 3rd Party Software

Managing Secure Access to the Host – When a user communicates to the ESX host using one of the above methods there are several security mechanisms available in the Service Console to prevent unwanted access. The Service Console man-

ages user authentication using standard Linux authentication mechanisms. In addition, an iptables firewall is enabled by default that allows only the type of access to the system necessary for support and management. For details on properly securing an ESX host please reference Chapter 9 of this book.

Running Support Applications - There are many command line tools available that allow a user to manipulate either Service Console or VMkernel information. These applications can be used to perform most any Linux function inside the management interface. While interfacing with the Service Console you will probably run across quite a few recognizable commands that are available in any standard Linux distribution. In addition to these standard utilities VMware has provided several of their own tools to both ease Linux management and interact directly with the VMkernel for virtual machine management. Unfortunately, with VMware deemphasizing the use of the Service Console, many tools that were available in previous versions of ESX are now missing. If there is one thing that VMware should learn from Microsoft it is that you do not mess around with an administrator's command line tools because THEY feel they are no longer needed.

Manage Access to Non-Core Hardware – The VMkernel directly controls access to CPU, Memory, Disk, and Network hardware resources. In order to trim some of the fat from the VMkernel, VMware decided that any non-critical hardware resource should still be managed by the Service Console. Some devices that must be emulated on a virtual machine and accessed through the service console are:

- Serial Ports
- Parallel Ports
- USB Ports
- CD-ROM Drives

VMkernel

As previously mentioned, the Service Console is responsible for bootstrapping the VMkernel. The big question is "What is the VMkernel?"

The VMkernel is VMware's core operating system that assumes responsibility for all hardware management and resource scheduling, and other major virtualization tasks on the ESX host. The process in which the Service Console bootstraps the VMkernel is similar (but by no means identical) to the way in which Microsoft DOS was used to bootstrap Novell Netware. When the VMkernel takes over the hardware resources of the host, the Service Console is warm booted and managed as a virtual machine within the VMkernel.

The VMkernel is what makes virtualization with VMware ESX Server possible. The following is an overview of the primary functions that the VMkernel is responsible for.

Scheduling CPU, Memory, and Storage Resources – The VMkernel is responsible for scheduling and ensuring virtual machines have access to the resources they require. We will talk about several mechanisms that the VMkernel uses to prioritize resource assignment when multiple virtual machines attempt to access more resources than are available. You may also notice that the VMkernel doesn't manage network scheduling. The simple reason for that is that there is no scheduling of network resources. Due to the time sensitivity of the TCP/IP protocol, any form of delays due to scheduling will potentially cause transmission issues.

Manage Memory Page Tables – The VMkernel has some very advanced memory virtualization and tracking mechanisms that, under the proper circumstances, can actually allow you to over-allocate memory resources of a host for virtual machine utilization. We will discuss these various mechanisms in detail when we discuss memory virtualization later in this chapter.

Manage the Virtualization Storage Subsystem – In addition to having a network stack integrated into the VMkernel, the newest revision of ESX also has an enhanced storage subsystem. This has support for a limited amount of storage types and file systems. When combined with the network stack, there are several low-cost storage alternatives available that have previously eluded virtual infrastructures. Detailed specifics on storage types and file systems are highlighted in Chapter 5.

Manage the Virtualization Network Stack – Unlike previous versions of VMware ESX Server, ESX 3 has a network stack built into the VMkernel. While this increases the size of the VMkernel footprint, it opens essential functionality for

low-cost storage alternatives. An additional benefit of having a network stack available directly to the VMkernel gives further flexibility to the virtual infrastructure through several virtual networking enhancements while minimizing the overhead caused from virtualization. Some of these features include, but are not limited to the following. If you do not understand all of the terms, don't worry, Chapter 6 is where all the terms will be fully described.

- Virtual Switches
- NIC Teaming
- VLAN Tagging
- Port Groups

Support for Loadable Modules – Much of the VMkernel's advanced functionality is made available through loadable modules. Having standalone modules that can be loaded into the VMkernel provide a simpler mechanism for updating the specific functional components without having to update the entire VMkernel.

Support for User World Processes – User World Processes are new to ESX 3 and are specially compiled binaries that are managed and scheduled by the VMkernel. These processes are similar to standard Linux processes with the exception that they are run in the VMkernel space and have no impact on the Service Console. The most significant benefit of this is that processes that were previously pinned in the Service Console were forced to only run on CPU0 of the physical server. Since User World applications are managed by the VMkernel, they have access to all CPUs in the physical host.

The User Worlds provide limited Linux syscall support for User World Processes. It is important to note that User Mode Processes are not standard Linux processes and must be compiled against the proper VMkernel headers to properly function. VMware has made the use of User Worlds available to members of their Community Source Program and several ISVs have begun to take advantage of this technology. Many of VMware's internal processing has also moved into the User World applications, which has lowered virtualization overhead on tasks and processes that were common in ESX 2.X.

VMX

The VMX is a User World application (vmware-vmx) and is actually the primary driver for the creation of User Worlds in the first place. In previous versions of ESX, VMX applications were run directly in the service console and were responsible for memory and CPU virtualization overhead. Now that these processes are scheduled by the VMkernel, the memory requirements of the Service Console are drastically lowered and there is no longer a tiered memory assignment/virtual machine relationship for the Service Console.

There are many functions that are often overlooked in a virtual infrastructure that are made possible by the VMX process for a virtual machine.

- Emulation of non-critical hardware resources through a Service Console Proxy (Video, CD-ROM, Serial Ports, Parallel Ports, etc)
- Bootstraps the virtual machine
- Communicates with User Interface components of ESX (Remote Console, etc)

By running "esxtop" in the service console you can actually view the various vmware-vmx worlds and how much utilization they are leveraging in the VMkernel. You may need to use the "e" command and choose a virtual machine GID to properly view the worlds, including VMX threads that are supporting that virtual machine.

VMM

The VMM, or Virtual Machine Monitor, acts as the traffic cop between a virtual machine and the VMkernel. There is one VMM process per virtual machine, and within a VMM process there is one thread per virtual CPU configured for that virtual machine. The VMM has several functions based on the type of resource being requested. The VMM has varying access to the physical hardware of the system to help enhance the overall performance of the VMkernel.

In the event that a virtual machine is requesting CPU resources the VMM determines if the instruction can be executed directly on the hardware, or if the VMkernel must be used to virtualize the call in a secure protection ring of the

processor. Don't worry; we make this easier to understand in a few moments when we start talking about actual CPU virtualization.

In regards to memory, the VMM is responsible for presenting non-contiguous physical memory pages as contiguous to the virtual machine. In addition to presenting these pages, it is also responsible for maintaining maps of the "virtual" memory pages back to the physical. This tracking opens up some unique opportunities in the virtualization space that we will talk about in the memory virtualization section on this chapter.

The VMM also has the task of passing other hardware calls to their proper components. Network and storage I/O traffic is handed off to the VMkernel to be processed by their respective stacks that are built-in to the VMkernel. Non-critical hardware calls are passed by the VMM into the VMX, which emulates the hardware functionality. Hardware access of CD-ROM, Serial Ports, Parallel Ports, and USB Ports are examples of hardware devices passed to the VMX.

Like the VMX components loaded into the VMkernel, you can also see VMM processes in the esxtop Service Console application. As before, you may need to expand the world groups to properly see the various VMM threads.

Hostd

Hostd is a daemon that runs within the Service Console that is responsible for the management of the ESX host. It provides the necessary communication link between external applications like VirtualCenter and the VMkernel. Hostd uses loadable libraries to manage various aspects of managing an ESX host server such as:

- Host Information
- Host Configuration
- Virtual Machine Inventory
- Virtual Machine Control
- Organize Performance Data

As mentioned earlier, VirtualCenter communicates with the Hostd daemon when specific functions are requested that affect a host server. Additionally, applications that leverage the VMware SDK to communicate directly with a host also communicate through Hostd using a SOAP interface.

Authd / VMKAuthd

Within the ESX infrastructure, VMware needed to provide a secure mechanism to manage virtual machines and minimize the risk of external access corrupting the environment. The Authd and VMKAuthd work in conjunction to provide a secure ticketing solution that allows that secure access to the VMkernel for system management and interaction.

The Authd daemon sits in the Service Console and manages connections and authentication requests coming into the host using TCP port 902. Since we never want unsecure access to the VMkernel, the Authd process will authenticate the user requesting access, and assuming the proper credentials are specified, supplies a ticket to the remote client. In addition to assigning the ticket to the client, the Authd process also sends the ticket to a location that the VMKAuthd daemon, which is a User World process that listens on TCP Port 903, can also access. The client will communicate with the VMKAuthd daemon and the secure ticket is verified. Once the ticket is verified, the client can directly access the VMX, which if you remember also exists in the User World space. Remote console connections and virtual machine management may now occur directly with the VMX in the User World without using any Service Console resources.

The reason for the multi-step ticketing mechanism is because of the native security provided by the VMkernel and User World applications. The Authd daemon, or any process in the Service Console for that matter, has no way to directly access or turn control over to a process running in the User World. Generating a shared ticket and allowing the VMKAuthd daemon to grab it for verification is what ultimately allows the client to communicate directly into the User World space.

Now that you hopefully have a solid understanding of the magic that makes virtualization possible using VMware ESX Server 3, its time we take a look at all of the advanced functionality that the VMkernel brings. We have made several

references to advanced functionality of the primary resources involved in virtualization. It's time we take an in-depth look at what all of the various VMware virtualization components opens up for your environment and how they allow you to maximize the physical resources available on your server.

Core 4 Resources – In Depth

There are four core resources that you need to strongly consider when you review and design your virtual environment. These resources are what we originally titled the "Core 4". Properly understanding and configuring these resources are essential to maintaining a stable virtual environment. This section focuses on these "Core 4" resources and how they pertain to VMware ESX Server 3 and its guests.

Processor

Assuming you are utilizing a processor that meets the requirements of ESX server, your guest will see the same physical processor that's installed on the host server through processor virtualization. By presenting the host processor type to the guest, the VMkernel does not need to perform any emulation to ensure compatibility between the virtual machine workload and the physical hardware. This information is essential when dealing with operating systems that load a different kernel that is based on a specific processor architecture. Trying to run a virtual machine's Intel instructions on an AMD processor would require emulation of the processor type and would literally destroy the performance on an ESX host with the number of processing cycles required to perform the translation.

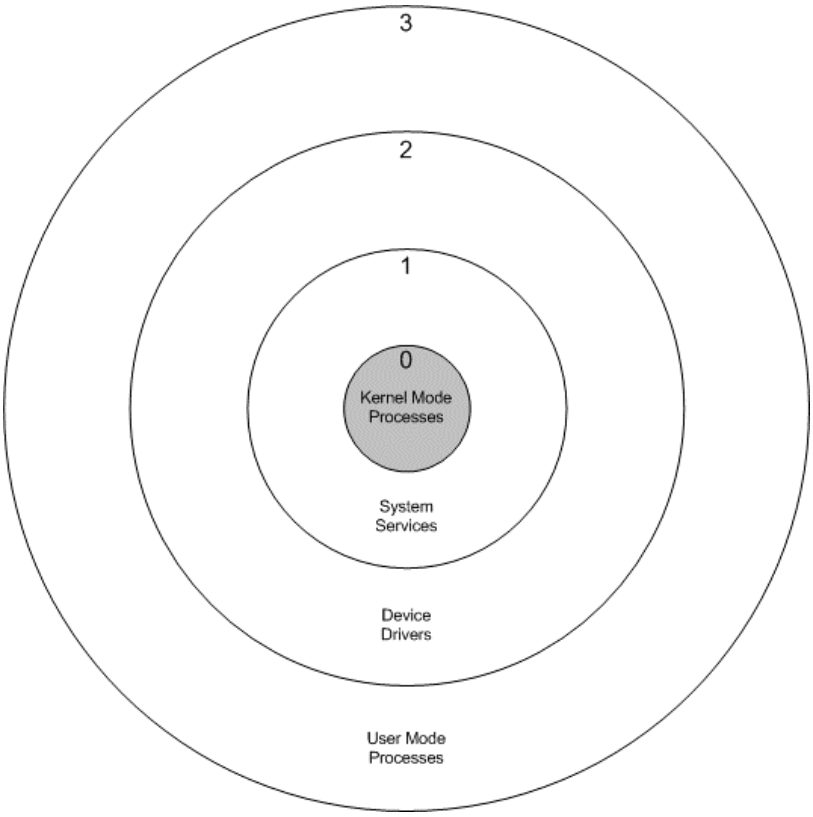
Virtual machines are assigned virtual CPUs as they are built. Each virtual CPU acts as a slice of a physical CPU core on the ESX host. New in ESX 3 is the capability to provide up to four virtual CPUs to a single guest operating system. When we start talking about Virtual SMP a little later in this chapter, you will see why that does not equate to 4X the performance of a single virtual CPU virtual machine. Overall, any single ESX host can have up to 128 virtual CPUs active at any given time. Remember, the VMkernel requires scheduling resources for each of these virtual CPUs, and at a point the amount of overhead required for scheduling will start to degrade the overall performance of all virtual machines running on the host.

Execution Modes

ESX 3 has two different data execution modes depending on the type of workload being requested; Direct Execution Mode and Virtualization Mode. Direct Execution mode is by far the preferred method of data execution and ESX will leverage this whenever possible. This mode allows the VMM running the virtual machine to directly access physical hardware resources of the host to process data. By running an instruction directly on the underlying hardware, the VMM does not need to virtualize any processing calls and can generate near native performance of the requested instruction. The amount of instructions typically being executed in Direct Execution mode is higher than those in Virtualized Mode, but there are always exceptions. To determine the overall utilization of Direct Execution Mode calculations in a guest operating system you should look at the overall percentage of User Mode Process utilization. Direct Execution Mode is used in ESX for all processing in Protection Rings 1-3 of the CPU as shown in Figure 2-6.

Virtualization Mode is used when direct execution of an instruction is not possible. In the case of ESX Server 3.0 this is any call that requires Protection Ring 0 access to the CPU. Ring 0 of a CPU cannot be presented directly to a guest operating system in ESX. Intel and AMD are creating methods in which a virtual Ring 0 (commonly called Ring -1) can be presented to the guest for direct process execution through the VMM. ESX 3 cannot take advantage of this functionality at this point. Instead, the VMM must send the instructions to the VMkernel to execute the instruction and return it to the guest. The VMM virtualizes this process to make the guest think that it is running in an actual Protection Ring 0. Due to the fact that the instructions must be virtualized and executed within the VMkernel there is virtualization overhead associated with these calls. The amount of overhead varies depending on the type of instruction and the overall workload being processed by the VMkernel. You can tell how often a guest operating system needs to run in Virtualization Mode by capturing the total percentage of Kernel or System Mode Process utilization.

Figure 2- 6: Processing Rings



Fortunately for us, the VMM responsible for running each virtual machine knows when an instruction requires Virtualization Mode and has the capability to properly move itself into the proper execution mode. When the requirement for Virtualization Mode has been met, the VMM will move itself back into direct execution mode.

Hardware Virtualization Enhancements

As mentioned earlier Intel and AMD have new processors on the market that have virtualization enhancements integrated into the processors themselves. At this point in time VMware ESX 3.0 does support some of the functionality, but not all. As an example, in order to run a 64-bit guest operating system ESX requires the use of a virtualization enhanced processor. This allows the VMkernel, which is a 32-bit kernel to run 64-bit instructions for the 64-bit guest. Alternatively, ESX 3 does not have the capability to present a virtual Ring 0 to the guest operating system for Direct Execution Mode processing of Kernel Mode Processes.

Multi-Core

One of the greatest things to happen to processors in the recent years is the creation of Multi-Core processors. While the performance increase of multi-core processors was acceptable in ESX 2, VMware has made sure that ESX 3 is fully aware of multi-core processors and initial results show excellent performance with these new processors. One of the major challenges with multi-core processors as the chip vendors keep cramming more into less is the amount of bandwidth available for memory access. We will only continue to see performance increases with these processors if the system bus is capable of keeping the rest of the system up to speed with CPU I/O. So far, VMware has been keeping with per socket licensing as opposed to per core licensing, which has also been keeping people extremely interested in the newer processors.

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%. What Intel is actually doing with this technology is presenting two logical processors to the operating system for each physical processor installed. The primary use for this technology is for enhancing task scheduling in operating systems.

VMware ESX Server 3 has full support for hyper-threaded processors. The additional logical processors presented to ESX are packaged with the physical CPU and are numbered adjacently. For example, processors 0 and 1 would be physical CPU1 and its logical counterpart and processors 2 and 3 would be physical CPU 2 and its logical counterpart. 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 attempt to use CPU affinity for pinning a virtual machine workload to a specific processor on a host.

VMware's resource scheduler is hyper-threading aware and is capable of properly balancing resource utilization across physical processors when possible. Without this intelligence it could be possible for CPU instructions to run on a physical CPU and its logical pair while a second physical processor on the system is sitting idle. The increase that a system receives from hyper-threading is

dependent upon how well the application running on the system utilizes the system cache.

Be aware that some of the newer Intel processors that support multi-core processing do not have hyper-threading extensions included in the processor architecture. Please review each processor chipsets functionality before determining if hyper-threading is to be used in your environment.

Virtual SMP

As we previously mentioned, Virtual SMP is an add-on module for VI3 that provides the capability to configure multi-processor virtual machines. While Virtual SMP can provide enhanced performance to your virtual machines, there are several guidelines that should be strictly followed. Virtual SMP can just as easily negatively impact the performance of an environment.

- Administrators should never start off by configuring a virtual machine with multiple virtual processors.
- Once upgraded to a Virtual SMP virtual machine, it is extremely difficult (and in some cases impossible) to properly downgrade a Windows guest.
- Utilizing Virtual SMP slightly increases CPU overhead of an ESX host.
- Virtual SMP should only be used if an operating system and application is fully capable of leveraging SMP extensions. Single threaded applications are not Virtual SMP candidates.

While performing best practices analysis of environments we've noticed that there are quite a few people that still start off by deploying every virtual machine as a Virtual SMP system. The added virtualization overhead from this configuration can e the source of significant performance problems as the environment becomes more utilized. By only utilizing Virtual 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.

Now that VMware has provided the capability to assign up to four virtual CPUs to a single virtual machine extreme caution must be used when virtualizing en-

terprise workloads. Even though VMware provides us with the ability to build enterprise level applications in a virtual infrastructure, we still feel that if that much horsepower is required that the system be built on a standalone physical machine. When host CPU workloads are pushed to their limits, there are several mechanisms that can be leveraged to make sure virtual machines get the CPU cycles they need.

Scheduling

The VMkernel was designed to provide a high level of interaction with the processors, allowing ESX to dynamically shift resources on the fly to running virtual machines. For example, if you have three virtual machines that are running in a primarily idle state on the same host as a SQL 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 virtual machine to effectively run the new application. Generally, a good rule of thumb is to allocate 6 virtual processors per core, although we've worked in some environments where seeing 8-10 virtual processors per physical core not out of the question. The types of processors also need to be considered when performing these sizing decisions. These numbers may be slightly lower if older hardware is being reprovisioned for virtualization, and slightly higher as new processor architectures hit the market. ESX does have a hard limit of 128 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 of your infrastructure. This limit can be broken down any number of ways mixing single processor and Virtual SMP virtual machines.

If further sharing 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 machine 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

virtual machines when the host becomes limited on processor cycles. As you add more virtual machines 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 machine created the allocation to existing machines decreases, which will slightly decrease their performance when the host system is under heavy load.

CPU Reservations and Limits

Within VMware ESX Server 3 you may assign a minimum (reservation) and maximum (limit) value in Megahertz for the processing resources of a virtual machine. By setting a reservation you are telling ESX to always ensure a particular virtual machine has the specified horsepower available when it needs it. This does not mean that a virtual machine must use the total reservation. If a virtual machine is sitting mostly idle, other virtual machines on the same host may use the reserved resources. If the idle virtual machine has a sudden need for CPU resources to process a workload, the reservation guarantees that the CPU cycles that were previously scheduled elsewhere are immediately available. The main thing to consider if you are going to be using CPU reservations for your virtual machines is that you must have the resources available before you can turn a virtual machine on. As an example, if you have a 4GHz processor with four virtual machines, each with a 1GHz reservation, you will not be able to power on any other virtual machines if the four machines with a reservation are all running. The default value for a newly created virtual machine is "0". This means a virtual machine has no guaranteed resources unless a reservation is set. To assign priority when virtual machines become contentious for CPU cycles and there is no reservation, CPU shares are leveraged.

Specifying a limit for a virtual machine has the exact opposite effect as setting a reservation. A limit provides a hard value in Megahertz that a particular virtual machine may not exceed. There are two key instances in which setting a limit on a virtual machine can be beneficial to the virtual infrastructure. First, by setting a limit you can tell ESX to allocate a maximum MHz value to a problematic virtual machine. This would provide a level of protection to the other virtual machines trying to compete for the same host resources as a machine that has runaway processes wasting CPU cycles. The second instance in which a limit can be helpful is for managing the expectations of application owners or developers on a low-utilization ESX host. If there are a small amount of virtual ma-

chines on a new ESX host there will be no resource contention and each VM will have access to the resources that it needs. As the host becomes more utilized by adding new virtual machines the performance of the original applications will slowly deteriorate. By setting a limit when the original virtual machines are first configured there will be no surprises later on as more VMs are introduced into the infrastructure.

Setting reservations and limits are independent of one another. You can choose to set only a reservation, only a limit, or specify both values for a virtual machine. Not setting any values gives ESX full control over the processor allocation of your virtual machines. While this is sufficient for small deployments with a low number of virtual machines, it may be crucial to manually adjust these values as the virtual infrastructure grows.

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 where the processing for a virtual machine occurs. Not only can you specify that a virtual machine is guaranteed a minimum reservation of processor and that it has a high level of priority in receiving additional allocations, you can also specify the exact processor or group of processors that proves the resources. Using this methodology you have complete control over how a production infrastructure reacts on a box and can ensure that every virtual machine has the proper resources without interfering with other critical applications.

Again, 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 CPUs, leaving the remaining processors inaccessible to the virtual machine. Application servers such as Citrix servers can be pinned to different physical CPUs to minimize the amount of scheduling required for kernel level calls from multiple systems. Support servers may then be pinned to the remaining processors and be allowed to share those resources amongst each other. 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.

Using processor affinity should be considered an advanced configuration and should only be leveraged if there is a specific need. If advanced VI3 functionality such as DRS and VMware HA will be used in the environment there can be severe implications to using affinity that need to be considered. As virtual machine resources move across hosts in a pool, the affinity settings will also attempt to follow. If other virtual machines are using affinity, there may be conflicts with reservations that will not allow a virtual machine to receive all of the resources the reservation requires.

Memory

Memory easily has some of the most advanced functionality available to VMware ESX Server. Memory is used in several locations in the ESX infrastructure and VMware employs several memory saving techniques to allow for the most optimal use of the physical hardware.

VMware ESX Server requires a certain amount of memory to be assigned to the service console. Versions of ESX Server prior to 3.0 required a varying amount of memory based on the number of virtual machines that the host would run. It was not uncommon to see hosts configured with anywhere from 200-800MB of memory. In addition, environments that grew beyond expectations often had to have their memory setting changed, which required a host reboot. The reason for this dynamic memory configuration was because the Virtual Machine Monitors (VMMs) that we described earlier had to run inside the Service Console.

In the new ESX 3 architecture the VMMs now run in a User World, whose resources are assigned and controlled by the VMkernel. The default value of 272 MB of memory should be more than enough for nearly every ESX Server environment. The only time it may be necessary to increase the amount of memory in the service console is in the event that multiple monitoring or backup agents are being run in the service console. The amount of memory that must be increased completely depends on the applications and agents running.

Any memory that isn't assigned to the service console is automatically given to the VMkernel for allocation and management. The VMkernel will assign memory to each powered on virtual machine based on that machines memory assignment. In addition, each virtual machine will consume a small amount of

memory for virtualization overhead. The amount of overhead memory depends on the configuration of the virtual machine itself such as the total amount of memory assigned. Alternatively, you will typically see 64 bit guests leveraging more overhead memory than 32 bit guests.

The VMkernel memory manager used some advanced techniques when allocating memory resources to virtual machines. There are several methods that the VMkernel uses to optimize the system performance, and in some cases allow a host to allocate more memory to virtual machines than is physically available to the host.

Basic Over-commitment

One of the simplest memory management techniques employed by ESX Server 3 is basic over-commitment. This method allows you to assign more memory to virtual machines than what exists on the physical host. As a simplified example, a host with 1GB of memory can have three virtual machines configured each with 512MB of memory. The premise behind this being not all memory from all virtual machines will be active at the same time. Virtual machines that are not actively using memory pages can temporarily give them to other guests that are more active. When the virtual machine that gave up the memory resources becomes more active, the VMkernel will wipe the memory pages clean and reassign them back to the original guest.

Basic over-commitment works well when the total amount of active memory doesn't exceed the amount of physical memory installed in the ESX host. When this condition is met the VMkernel must use some alternate mechanisms to allow virtual machines to continue running. Like processing resources, ESX has the capability to dynamically optimize utilization by using "shares" to provide priority to virtual machines. When memory pages become available on a highly utilized system, shares dictate which virtual machines get access to the physical memory pages and which need to use an alternate mechanism to store memory.

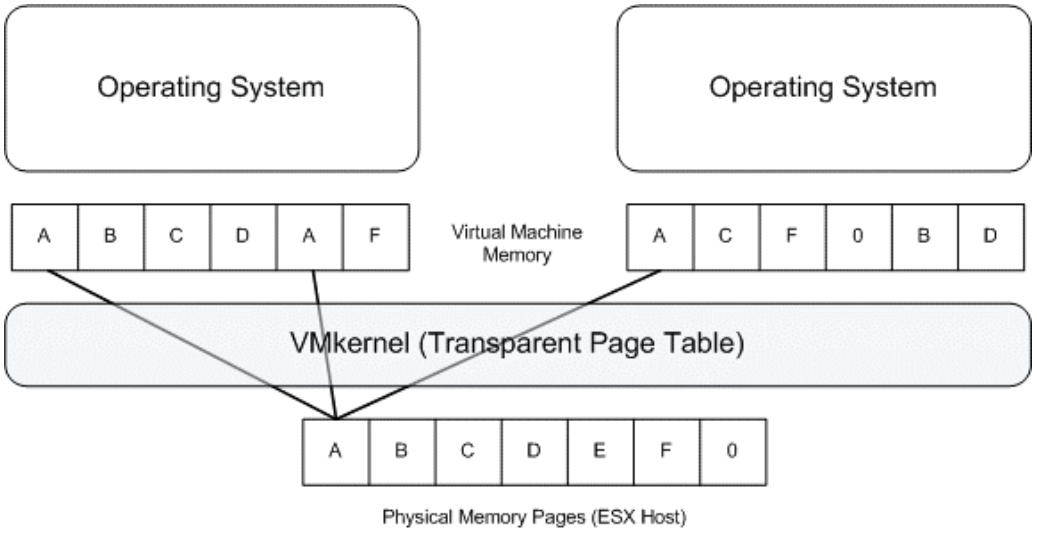
These methods provide basic memory management for your virtual machines. VMware ESX Server also leverages several other techniques to allow for the over-commitment and management of memory resources: Transparent Page Sharing, ballooning and swapping. While we recommend only using sharing

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 contain common pages found on all similar operating systems. The same can be said about applications that run inside the guest operating systems. The VMkernel contains a Transparent Page Table that keeps track of the memory pages in the virtual machine and maps them to the memory pages in the physical memory in which they are stored. A basic overview of this technology is highlighted in Figure 2-7.

Figure 2- 7: Transparent Page Sharing



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. The VMkernel automatically performs Transparent Page Sharing takes place automatically it is the only memory allocation method that takes place without the host running at maximum memory use.

Ballooning

When over-commitment is occurring beyond a point that Transparent Page Sharing is optimizing the system performance VMware has provided functionality to let the guest operating system decide what memory it wants to give back for sharing. This is made possible by using a memory reclamation process called "ballooning." Ballooning consists of a vmmemctl driver installed in the virtual machine that communicates with the VMkernel. This driver emulates an increase and decrease in memory pressure on the guest operating system and forces it to place memory pages into its local swap file. Once the memory is paged locally on the guest operating system, the free physical pages of memory may be reallocated to other guests. Since the ESX host sees that memory demand has been reduced inside the virtual machine it will instruct vmmemctl to "deflate" the balloon, thus reducing 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.

Swapping

ESX has its own application swap file that is configured when the system is built. This file is independent of both the Service Console 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 beyond the point that Transparent Page Sharing and Ballooning can recover memory, the VMkernel takes it upon itself to swap memory to the ESX page file. This differs from ballooning in the fact that the VMkernel decides what memory to swap, regardless of whether the guest is using it or not. This technique may be useful in a development environment where paging will have less of an impact on the overall performance of the environment, but should be avoided at all costs on a production host.

Idle Memory Tax

When memory share allocation takes effect, VMware provides a mechanism to prevent virtual machines from hoarding memory they 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.

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. This process is significantly slower than accessing memory that is located on the same bus as the processor. VMware ESX 3 is a fully NUMA aware 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.

Storage

Storage in VMware ESX 3 comes in several flavors. In addition to the traditional local SCSI and Fiber attached SAN options, ESX 3 also supports new low-cost storage alternatives in the form of iSCSI and NFS. When dealing with storage for ESX 3 there are two things that need to be considered; the installation point for ESX Server itself and storage space required for storing your virtual machine data.

Install Point

A requirement of VMware ESX Server 3 is that it must be installed on a server to properly function. The installation of ESX server itself takes about 1.2 GB of storage space. You are not alone in thinking "This will not come close to filling up my 146GB SCSI drive". ESX 3 is optimized for performance and many packages that can bloat the installation are not included. Having a small storage footprint for the VMkernel and other installed packages fits perfectly into the Boot from SAN model, in which a small LUN can be presented to the host and leveraged as the installation point. This would separate the entire ESX

host, including configuration files, from the physical hardware. We will go into much more detail in Chapter 5 when we discuss advanced storage planning, but feel it is important to highlight some of the primary differences between a Local install point and a boot from SAN model.

Local Installation

- Very inexpensive solution
- Easy to implement
- Eliminates external factors from affecting operation

Boot from SAN

- Separates install from hardware
- Easy recovery from hardware failure
- Better use of storage space (less wasted)

As you will see in the next chapter, there is a limit on the amount of space that can be utilized on a large local hard drive when dealing with ESX. If the decision is made to leverage local storage you are better off going with smaller but faster drives. You will also need to be prepared to waste a lot of storage space.

Virtual Machine Storage

In addition to storage space required to install and run VMware ESX Server there is also a need to dedicate storage space to virtual machines that will be run in the environment…a LOT of storage space. VMware has several supported types in ESX 3 which include: Local SCSI, SAN, iSCSI, and NFS. There are advantages and disadvantages to each solution that will be discussed in Chapter 5. As you dig further into the planning and implementation of your virtual infrastructure, you will quickly find that storage becomes one of the most expensive components in the entire environment.

VMDK Files

We say VMware requires a lot of storage for running virtual machines, but why is that? The answer lies in the virtual machine file structure leveraged by VMware ESX Server. Many people who are familiar with VMware Workstation or VMware Server (GSX) may be familiar with the practice of "Copy on Write" disks. As more data is added inside a virtual machine, the virtual machine disk file grows. This method is NOT the case inside ESX server. When a virtual hard drive (or VMDK) file is created, all space is immediately allocated. A virtual machine that has a 40GB data drive assigned to it will immediately take up 40GB of storage space on the ESX host, regardless of how much data is stored in the virtual machine.

Although we are withholding all of the good information for Chapter 5, we do feel it is important to quickly describe a VMDK file. A VMDK file is actually made up by a set of files (typically two) that acts as a physical hard drive device inside the guest operating system. Inside the virtual machine, a VMDK file appears as and is accessed like any typical SCSI drive in a physical server.

VMFS

VMware stores VMDK files inside a file system that is customized for speed and high availability called VMFS (we make an assumption that this stands for either Virtual Machine File System or VMware File System). In versions of ESX server prior to 3.0, VMFS was a flat file system with no capability to store or organize files in subdirectories. Every VMDK file was stored directly on the root of the file system. We noticed that this design made some VMware ESX implementations quite interesting when it came to data management at the VMFS level. Because VMFS was a flat file system, the only items worth storing inside a VMFS partition were either VMDK files or REDO files (which are the old name for Snapshot files, which we will discuss shortly).

ESX 3 has now introduced a new version of VMFS, which is simply called VMFS3. The file system now has a directory structure available in which subdirectories can be used to organize data per virtual machine. In addition, more information is stored within each virtual machine's subdirectory such as virtual machine configuration information and log files. Having these files available inside VMFS is what makes rebooting a virtual machine using VMware HA possible, assuming you are using a centralized storage infrastructure of course.

Snapshots

We previously made a brief mention of REDO and Snapshot files but didn't really describe them. Both REDO and Snapshot files are extremely similar. REDO files were used in versions of ESX prior to 3.0, and Snapshot files are the new standard. Since this is technically a book on ESX 3, we will leverage the use of the name "Snapshot". Creating a "Snapshot" for a virtual machine forces the VMkernel to create a Snapshot file alongside the VMDK file in the VMFS file system. As soon as this action occurs, the VMDK file is unlocked and all data changes are written to the Snapshot file. Once a Snapshot has been established, a virtual machine can be reverted to its original state, or the Snapshot data can be applied into the VMDK file. Either of these actions removes the Snapshot from the virtual machine.

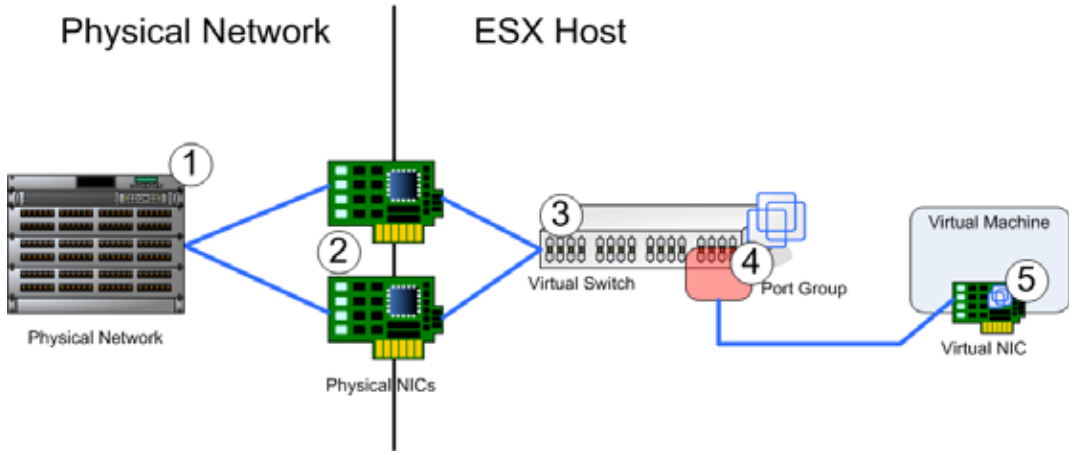
New to VMware ESX 3 is the capability to create multiple branches from a single VMDK file using multiple Snapshots. We will go into more detail on this advanced topic in Chapter 5. There are actually several instances in which Snapshots can play a crucial role in your virtual infrastructure.

- Unlocking VMDK files for image level backups
- Testing application changes/upgrades before permanently applying them
- Branching several development configurations from a single template virtual machine

Network

Networking in VMware ESX 3 is another topic that is deserving of its own chapter in this book. Although Chapter 6 is completely dedicated to the advanced networking strategies of ESX 3, we feel it is important to describe some of the basic terms and give you an overview of the networking capabilities of the VMkernel. Figure 2-8 displays a basic configuration and highlights the major components involved in ESX networking.

Figure 2- 8: Network Infrastructure



Physical Network – In order for your virtual machines to communicate with the rest of your environment the ESX hosts must be connected to the physical network. We will discuss several options in Chapter 6 to enhance performance and provide high availability to your virtual infrastructure.

Physical NIC – Multiple physical NICs can be leveraged on an ESX Server for load balancing and high availability. VMware ESX can use 10, 100, or 1000Mb networking, and VMware is currently working on support for InfiniBand networking. The maximum number of physical NICs varies based on network card vendor and speed.

Virtual switch –A virtual switch is just what its name describes—it emulates a physical switch for the guests that are configured to utilize it. From 1 to 32 physical NICs can be used to create a virtual switch on an ESX host. In addition, a special type of virtual switch can be created without any physical NICs. This type of switch will be discussed in Chapter 6.

Virtual Switches can be configured with multiple ports that again, act similarly to ports of a physical switch. When you create a virtual machine with a virtual NIC and assign it to a Virtual Switch you use one port of that virtual switch. The default number of ports assigned to a virtual switch in ESX Server 3 is 56, and the maximum number of ports that you can assign to a single virtual switch is 1016.

Virtual switches are responsible for load-balancing virtual machines across all physical NICs used to create the switch. If one physical network switch port or Physical NIC were to fail, the remaining Physical NICs 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.

Port Group – Port Groups are logical groups of ports of a virtual switch that have common configurations. Port Groups are not assigned with a static number of available ports. Any time a virtual NIC is configured to use a specific Port Group within a virtual switch it uses one available port of the virtual switch and the Virtual NIC is added to the Port Group.

As previously mentioned, several common configurations are shared amongst all Virtual NICs assigned to the same port group. These configurations include:

- VLAN Configurations
- Security Settings
- Bandwidth Control Policies
- Load Balancing Mechanism

Virtual NIC – In order for virtual machines to communicate over the network they must contain at least one Virtual NIC. A Virtual NIC is mapped to a Port Group on a Virtual Switch which is made up of one or more Physical NICs on the Physical Network (See how it all comes together at the end?). A single virtual machine can have as many as five Virtual NICs installed. It is uncommon that a standard virtual machine configuration would need more than two NICs. We have seen some instances of virtual machines acting as routers, which would have a requirement to have a connection to multiple VLANs. The default driver presented to the virtual machine for the Virtual NIC is an enhanced version of the AMD PCNET adapter. While this adapter can run on older systems with a standard driver, the VMware Tools must be installed to take advantage of advanced functionality such as Gigabit connectivity.

This section is just a primer of what to expect in Chapter 6 where we will discuss all of the available configuration options including their advantages and disadvantages. We will also go into a bit more detail of each of the highlighted components from this introductory chapter.

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.