VMware® Infrastructure 3

Advanced Technical Design Guide

~and~

Advanced Operations Guide

Two books in one!

Ron Oglesby
Scott Herold
Mike Laverick

Chapter 3 - ESX 3.0 Implementation

Now that you have an understanding of how ESX Server works, we need to move on to making decisions about your new environment. This chapter focuses on hardware selection, hardware design, and the basics of ESX installation in your environment. It should be noted that step by step procedures for installation are found in the operations section of this book, here we are focusing on the design of the environment not the step by step configuration.

In this chapter, we'll look at the various elements affecting the performance of ESX servers. Then we'll examine various real-world server sizing strategies so that you can create your own strategy. Lastly, we'll go through the basic installation choices for ESX and describe the configuration options available to you.

ESX Server Hardware Design

Server sizing in ESX environments is quite a bit different than server sizing in traditional server environments. Since the ESX host will have multiple virtual machines simultaneously accessing the ESX server resources, the hardware tends to be much more robust than that of standard servers.

To adequately create your server sizing strategy, it's worth inspecting how server sizing works in ESX environments. To do this, we'll focus on each of the major server hardware components and how they affect the virtual environment.

Before addressing hardware, however, there are a few things that you should keep in mind.

First of all, when you design your ESX servers, you need to make sure that you have "real" server hardware. Desktop computers turned on their side do not constitute "real" hardware. VMware ESX Server has a very strict hardware compatibility list. Unlike some operating systems, you will not be able to run ESX successfully on hardware components that are not on the HCL. At this point I usually will hear the argument that since ESX is really Redhat, "We can get it to work." With ESX this simply isn't true. Even if you can get the hardware to work for the service console that is Redhat based, it does not mean that

the VMkernel will recognize the hardware you are "forcing" into the system. Save yourself hundreds of hours and maybe your job, and get a server or components from the VMware HCL.

Now, let's get started with our exploration of server hardware in ESX environments. We'll begin with memory utilization.

ESX Server Memory Usage

When estimating the amount of memory you will need to account for in an ESX build, it is important to not only allow for some service console memory, but to also take into account memory sharing, over allocation of memory, and the amount of memory used by the VM's themselves. Every VM on your ESX server (that is powered on) will use some memory. The amount of "real" physical memory in use will depend on a large number of factors, all of which play into your basic design.

If ESX Server had a completely flat memory model, where no memory was shared between VM's, calculations would be simple. You would need to purchase enough physical memory for your host to supply each VM with the amount of memory you wish to assign to it and enough memory for service console operation. Some engineers actually do this to ensure that little, if any, memory swapping occurs. However since the ESX memory model is not flat and memory is shared and reclaimed in the environment, it is important that you take these memory 'tricks' into consideration.

In addition to the memory sharing, you should also take into account the types of VM's you will be hosting. Basically, you should create a list of the types of OS's you will support, the applications they will host, their environment (Prod, dev, QA etc), and the general amount of memory you wish to assign to the VM's. In Chapter 7 we discuss creating a VM standard for Memory and other configurations, but here we will describe some real world recommendations and discuss the other design attributes that go into sizing the memory on your server.

The amount of memory put into a server is closely tied to the number of processors in your system. The reasoning behind this is simple; you will create most of your estimates for number of virtual machines per host based on the number

of processors in that host. This number (the number of VM's per host) will determine the amount of RAM you will need to design into that host. If you are going to use dual processor servers and are expecting 8 to 10 VM's per dual processor server, then you need to have enough memory in the server to support those 8 to 10. This is much different than using an 8-way server and expecting 45-60 VM's per host.

So how much memory is enough to support those 8 to 10 or 45 to 60 Virtual Machines? Well the design answer can be found in the following items:

- Workloads placed on the VM's and their memory requirements.
- Performance requirements for VM's (Performance SLAs).
- OS's being hosted.
- Memory standards for VM's.

Workload of the VM's

So what is a workload? Simply put, the workload is the applications or types of applications running on the VM's. Different applications (workloads) require different amounts of physical resources, including memory. But workload is much more than just the application. Imagine a development Exchange server that you use to test upgrades. This server will require less memory than a production VM hosting Exchange mailboxes for 500 users, thus there will be different loads placed on the system by these two machines.

Test and Dev Workloads mixed with production VM's

The other question about workloads is how you are going to spread them out in your environment. Some companies choose to separate the production VM's from dev and test VM's, in doing so they create specific farms or specific hosts for production and specific hosts for dev and test environments. The big advantage in this configuration is that it guarantees that test VM's will not impact the performance of production VM's (which is still kind of hard to do in a properly design environment). The obvious drawback of this model is that you could be creating an environment where all of your "eggs" are in one basket, and a single ESX server failure will increase the number of production VM's affected.

On the other side of this equation are environments that mix test and production VM's on the same hosts and within the same clusters. In this configuration, administrators generally allocate a higher number of resource shares to production VM's to help ensure that test VM's do not negatively impact production VM's. The big advantage of this model is that test and dev servers are often idle. This idle time can allow for better performance of the production VM's since fewer of the server's resources are in use all the time. Of course depending on the security or network model in your environment, this configuration (test and dev mixed with production) may not meet your requirements.

Advantages of Mixing Dev, Test, and Production Environments:

- Dev and test servers are idle the majority of the time, offering best performance for Prod VM's.
- During a host failure, a smaller number of Prod VM's are affected.
- Dev servers often require less memory resources, allowing you to allocate more to Prod VM's.

Disadvantages of Mixing Dev, Test, and Production Environments:

- A runaway process on a test box creates the possibility that performance can be impacted on a production VM.
- It may require more network ports if the prod and test networks are kept separate.
- It may not meet your company's security or network design.

The mixing of workloads and resulting improvement in overall production VM performance is a large factor in some decisions. Your VM configuration, the amount of memory you allot for each VM, and your hardware may keep you from having to install memory expansion boards in your servers, resulting in a reduction in your server costs.

Real World Recommendation: Mix workloads when possible and when security policies allow. Mix High memory VM's with VM's that require little physical memory. In addition, assign less memory, memory shares, and processor shares to test and dev VM's than you do to your production VM's.

Performance Requirements for VM's

Performance SLA's will have a large impact on the memory design in your ESX server. As stated in the previous chapter, ESX allows you to control (to some extent) how memory is shared and distributed among VM's. Memory design for your host should take into account performance SLA's for your VM's. If your VM's are required to act identically to a physical server, or as close to a physical server as possible, you may not be able to over allocate memory in your environment.

Some of this configuration may also depend on your previous decisions about mixing (or not mixing) dev and test workloads with production VM's. The idea is that you may be able to over allocate more memory if you have dev and test VM's on the host. If the host server only has Production VM's on it and your performance requirements dictate a high SLA, you may not be able to over allocate memory.

Real World Recommendation: Mix guests that have stringent SLA's and loose SLA's. This will allow you to give more shares and or allocate more memory overall to the guests that need it, without having to purchase huge amounts of memory.

Operating Systems Being Hosted

Obviously the types of operating systems being hosted will have a major impact on the memory design for the ESX Server. The memory required to host 10 Windows 2003 Server guests is much higher than the requirement for hosting 10 Windows NT Servers. Take this comparison a step farther and compare a Windows 2003 Server guest with a guest running Linux to test a simple firewall. The difference can be huge.

Recently we have been helping clients test Windows Vista at numerous sites. One item we noticed was that Vista is a complete dog below 1.5GB of memory. 1GB was barely usable, 1.5 it gets to the point of OK, and at 2GB it seemed to run fine. Compare this to a Windows 2003 Standard web server that runs just fine at about 512MB of ram.

In addition to all of this confusion, the more VM's you have that have the same types of OS's, the more you will be taking advantage of ESX's memory sharing abilities. We have found that when running all like OS's you will generally see about 20 percent memory savings due to sharing. The long and short of it is that you have to create a list of servers that will be hosted or project what will be implemented in the environment to gauge the amount of memory you will need.

Advantage of Mixing guest OS's

- Mixing guest OS's often gives you a mixed workload and therefore better performance.
- It keeps you from creating "silos" of ESX server for specific Os's.
- It creates a simple, easy to understand environment; any VM can go anywhere.

Disadvantage of Mixing guest OS's

- Savings from memory sharing will be reduced.

Memory Standards for VM's

In chapter 7 we will discuss creating a VM standard, but here we need to at least understand the theory. Much like physical hardware standards, you need to create a VM hardware standard. For memory, this will mean a standard memory configuration per type of VM being built. Much like physical hardware, people tend to over allocate memory to VM's. Often we will visit a site where the admin in charge of VM's has basically enabled every option and given every VM as many resources as possible. He may have a test Windows 2000 VM with 2GB of RAM assigned and Virtual SMP for a server that is only used to test ADSI scripts.

The problem with this type of over allocation is that it is not a linear move in cost. With a physical server 2GB of memory can be attainted very cheaply if using 512MB DIMM's. Of course to be able to support a large number 2GB VM's you will need a large amount of memory in the ESX host. This often requires that you purchase all 2, 4, or 8GB DIMM's and possibly a memory expansion board. As you can see, the cost will not be the same MB for MB.

It is important to understand and have memory standards for your environment. Once you have these standards you can use them, the workloads of the guests, OS types and environments to determine the amount of memory needed.

Real World Recommendation: Create a standard that is realistic. 2GB of memory for every VM regardless of actual requirements is not good engineering, that's called over engineering. Instead create a standard that allows you to provide good performance for all VM's and allows you to change memory configurations as needed. Often we will recommend that about 512 to 768 MB of memory per Windows 2003 VM is a good starting place. This can also be adjusted up or down depending on development or production environments (this is detailed in Chapter 5). If the server requires more memory (say 1-1.5GB), then increase the memory for that specific VM. This increase is often offset by the VM's that are running at 512 or 768MB and will keep the average at or a little less than 1GB per VM.

So Much Memory per Processor

On servers hosting nothing but production VM's, we like to recommend 4 GBs per processor core in the host ESX Server. Note that it says CORE not just processor. This falls nicely in line with the average of 4 VM's per processor core that most environments see. Additionally, when factoring in the memory allocated to the Service Console, memory saved through sharing, and any lost to virtualization overhead, you still will be able to allocate an average of a GB or so per VM. If you expect to allocate more than this, up the memory. I recently completed a project where we used 24GB of memory in dual core, dual processors servers. This equated to 6GB of ram per core, and about 1.5GB per VM on average.

If you plan on running development or test VM's only, then you have the ability to run a much higher ratio of VM's to processors. In addition, you can more easily over allocate memory to VM's in test and dev environments since these servers are often idle and performance is not as important as it is in production environments. We have found that in a number of test environments the average ratio is about 5 to 7 VM's per processor core.

The long and short is that we like the 4 GB range per processor core for most environments. It allows you enough memory to host production VM's and still allows you the flexibility to over allocate for test and dev environments.

ESX Server Processor/Core Usage

When it comes to sizing ESX Server processors and cores, don't take the time to beat yourself up over the difference between 2.8 and 3.2 gigahertz processors. Processor speeds (clock speeds) are dictated by Intel, AMD and the hardware vendors, and you're pretty much forced to take what they offer. While speed itself can be important, the real decisions should be made around processor and system bus architectures and the number of processors / cores per processor per host. In most cases, you need to figure out whether your server will be a dual or quad processor blade, a standard dual, a quad-processor chassis, Eight-way, or maybe even a 16 processor box. Then toss on top of that whether you want to run dual core or a quad core system.

In deciding how many processor cores you want in your servers, keep in mind that you will have to find a balance between number of cores in a server, the amount of VM's hosted per server, and the number of ESX servers you are willing to manage. Also consider the amount of risk associated with a hardware failure on your ESX host. There is a big difference between a 16 processor host failing with 80 or 90 VM's on it and a dual processor dual core host going down with only 16 VM's on it.

Each VM will have to share processor time with the other VM's. Even if a VM is doing nothing at all, servicing no requests, hosting no files, etc., some processor will be used. In this design, if one VM does not need much processor time, then more processing power is available for other VM's. Due to the way ESX server shares the processor and the fact that each VM's processor requests can (by default) be executed on any processor on the server, it is fairly easy to scale up ESX. But one thing to remember is that the more processors you have, the greater your risk of running into a bottleneck in another part of the system.

In a perfect world, your processor utilization would constantly be in the 80 percent range. This would indicate that you didn't waste money buying too many processors, but also that no VM's are waiting for processing bottlenecks.

Single core Vs. Dual core Vs. Quad core Vs. AMD vs. Intel….

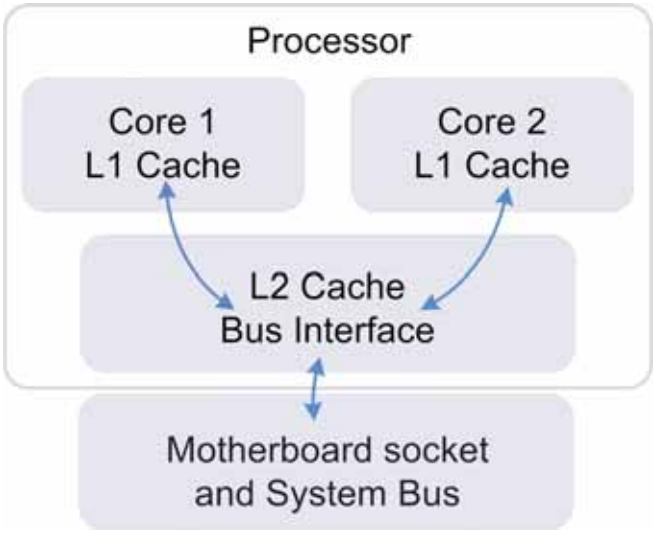
Which one to get? AMD dual core, Intel Dual core, Intel VT dual core, AMD V dual core, Intel quad core…. The reality is that there are numerous (read- way too many) white papers, spec sheets, opinion articles, and blogs out on the Internet right now detailing which is better, which manufacturer is better, why clock speed is not as important as bus speed, why Intel beats AMD, why AMD beats Intel, blah blah blah.

In the real world of x86, the operational world where most of us happen to live, we wouldn't notice the difference between an AMD and Intel processor that are both dual core with similar clock speeds on a similarly configured server. Sure we could throw a bunch of math at them to determine how long they take to finish a certain task, or run SQL and Exchange load tests at them to find their minute differences, but the reality is that a good processor in an ESX server is a good processor. After that it really comes down to cost and personal preference. But, for argument's sake, to get some people their fill, cause a stir in the forums, and a rise in hate e-mail we receive, we will discuss the "multiple core" question here, what makes a good ESX processor and possibly some processor virtualization while we are at it.

Multiple Core Processors

These days there is much talk about multi-core processors. The advent of which really came from the increasing difficulty at making a single core processors faster. Remember that in 2004 you could purchase a 3.0 GHz Pentium processor, well you still can in 2007. So manufacturers of processors turned to two major strategies to improve (read- sell) more processors. The first was to continue to integrate new improvements onto the processors for things users use like multi-media functions. The second was to continue to increase the speed/performance of the processor by adding more cores to a single processor essentially doubling the number of threads that could run at any instant and allowing more cycles to be executed.

Figure 3.1: Basics of a dual core processor



In figure 3.1 we show the basic concept behind a dual core processor. A single processor contains two independent cores. These cores interact with the system through a single bus interface (for good or bad) and share and L2 cache. From an ESX perspective the increase in the number of cores per CPU socket is a good thing. Knowing how VM's are scheduled on the processor, and understanding that each core looks and acts like a unique processor, you can see that an increase in cores allows for you to simultaneously execute more virtual machines on a single socket. This of course can increase the number of VM's you can run on a specific system or decrease the cost of the system by reducing the number of processors you need to purchase. An additional benefit of this is that VMware licenses ESX Server per Socket, not per core. Eventually this may change, but right now it is allowing you to get near quad processor consolidation ratios for the cost of a dual processor license. Buy it while the buying is good!

At the time of writing (end of 2006 beginning of 2007) dual core processors are very prevalent and quad cores are being introduced into most server lines. Maybe by the time you read this sentence, 8 core processors will be available. What you need to determine is the number of cores you want to put in a server and this decision is a balance between the number of cores and amount of memory in the system. Too many cores and you run out of memory before processor, too few cores and you have a processor bottle neck.

One point that should be made is that a core is not a processor. Based on the concept above, you can see that the cores do need to share the bus interface, meaning, that while a dual core processor offers you more processing power than 1 single core processor, it may not run even in performance to two single core processors. As for quad cores, well the verdict is still out on those at this time. Our first recommendation would be to never run a single Processor system with ESX no matter the number of cores. If you have the option of a single proc quad core or a dual proc dual core, buy the latter.

Most architects design their ESX systems using 4 to 16 cores per host. Using an average of 4 to 5 VM's per core you can accommodate (generally) 16 VM's on a 4 core system about 32 on an 8 core system (some customers see higher ratios, some lower). Designs using 4 to 8 cores per server tend to be a good balance among the number of VM's per host, cost of the host, and number of hosts that have to be managed.

Dual core, dual processor systems (4 total cores) are generally the most cost effective as you are using a traditional 2 processor chassis that is pretty inexpensive. As quad cores become more prevalent (and the cost comes down) it will be possible to get 8 cores in this chassis as long as the amount of memory you need is not outrageously priced.

If you want scale with large servers, the quad processor, quad core servers (like the HP DL 585 using AMD processors) have had excellent results as ESX hosts. The memory needs in these systems are extremely high and in some cases you may have to choose slower memory to get the required amount installed into your systems. The up side of these systems is that they can host a TON of virtual machines. I have personally seen a series of 585's with 48 GB of ram running live VM's up into the high 40s and low 50's. That's a serious consolidation ratio.

AMD vs. Intel

This is kind of like "pick your religion." Personally I am an AMD fan. I like AMD because I like to be different, and I believe in the concepts behind their memory architecture and that it may make an impact on my ESX environment. That being said I have done more projects with Intel processors and have never seen a case where either processor showed a significant advantage in number of

VM's hosted on a like configured system. Pick your processor and enjoy, worry more about the three C's: cores, cache, and cost.

Virtualization at the processor level

Where to begin, where to begin? I guess the first place to begin is that ESX 3.0 does not care about processor virtualization. It doesn't use it, doesn't care if it's there. ESX was designed, built, and in production long before the Intel VT and AMD V were in real silicon. In the current rev of ESX it actually is SLOWER for them to use the virtualization at the processor than to continue to execute Ring 0 calls the way they do now.

It is my assumption that in the future VMware will find a way to take advantage of these hardware changes, but the reality currently is that V and VT were put in place for use by Xen Source systems and possibly Microsoft's next virtualization platform.

Real-World Processor Recommendation

Processors are very fast and very cheap. Buy systems with at least 4 cores possibly 8. Never buy a system using a single socket; start at dual processors (multi-core) and work your way up. Stay in the 4-8 core range, and you are safe. If you venture into the 16 or 32 core per server arena be prepared to spend a ton of cash on memory. It can work, it's just expensive.

ESX Server Hard Drive Usage

In most environments your ESX server hard drive configuration will depend on your company's storage strategy (if they have one). The hard drive configuration for your ESX host will look extremely different if you plan on using SAN or NAS for storing VMDK files instead of local storage. In Chapter 5, we discuss storage design, if you plan on designing your SAN solution for ESX based on this book, you may want to jump ahead after reading this. This section will purely focus on the local drives and partitions for the ESX host.

In the 2.5 version of ESX, VMware introduced its boot from SAN feature. This feature was pretty much made for the Blade servers now being implemented by

many organizations. Personally, I feel that boot from SAN is really not needed in the ESX world. If done correctly with a SAN your ESX server becomes an appliance that executes VM's but does not store or configuration information. Knowing this and knowing that storage on the SAN is generally so much more expensive than local storage, using the SAN for the OS of an ESX server (including swap space and all) is pretty much useless, but that's just my opinion.

Because of the way that hard drives are typically used in ESX environments, you don't need very much storage space on individual servers. Typically VMDK files are stored on a SAN solution, and the only things that are stored locally are the console operating system and its own configurations. In ESX 2.5 local VMFS based swap was used for VM swapping. This has been moved to the VMFS volume hosting the VM's (generally on the SAN). Also in 2.5 the VM configurations also were stored on the host. These too have been moved to the SAN. The ESX servers (over several versions) have become more and more appliance like.

In general, a mirrored set of drives will work for almost any installation using SAN for its VM storage. If you are going to use local disk to store VMDK's then you should plan on a couple of large raid volumes. Maybe a mirrored set for the ESX Service Console and a large raid 5 volume for VMFS storage.

Storage Installation Options

One of the first steps in the ESX setup process is configuring the installation of the console operating system. Based on a combination of VMware best practices and our personal experiences with VMware, we recommend a partitioning scheme that will create a very flexible environment regardless of how you move forward with ESX. Below is a table that is duplicated in the Operations Guide, these are our recommendations and notes, but may need to be adjusted based on your specific requirements.

Mount Point	File System	Fixed Size	Size in MB	Force to Primary	Purpose
/boot	ext3	X	250	X	Core Boot (img) files
n/a	swap	X	1600	X	Swap for Service Console
/	ext3	X	5120	X	Main OS location
/var	ext3	X	2048		Log files
/tmp	ext3	X	2048		Temporary Files
/opt	ext3	X	2048		VMware HA Logging
/home	ext3	X	2048		Location of users storage
NA	vmkcore		100		VMkernel Panic Location
	vmfs	Fill to remaining disk if you want			Local storage is only required for VM Clustering

ext3

EXT3 is the primary file system of Linux. As the Service Console is based on Redhat Linux it is the one we will use. There are two other propriety file systems which are only accessible by the VMkernel. These are vmkcore and VMFS version 3. We will cover VMFS in more detail after the install has completed in the storage chapter.

/boot

This is where core boot files with an img extension are stored. After powering on ESX, the master boot record is located and boot loader is run. In the case of ESX 3.x this is now the GRand Unified Bootloader (GRUB). GRUB then displays a menu which allows you to select what .img to execute. Img files are image files and are bootable. They are analogous to ISO files which can also be bootable. I've doubled the VMware recommendation to err on the side of caution. Previous VMware recommendations have made this partition too small. This gave people problems when upgrading from ESX 2.x to ESX 3.x through lack of disk space.

/swap

This is where the Service Console swaps files if memory is low. I've chosen to over-allocate this partition. The default amount of memory for the Service Console is 272. VMware usually takes this number and doubles it to calculate the swap partition size (544MB). The maximum amount of memory you can assign to the Service Console is 800MB. This is how I derived the 1600MB value. This means if we ever choose to change the default amount memory assigned to the Service Console – we do not have to worry about resizing the swap partition. It doesn't have a mounting point as no files from the administrator are copied there.

/ (referred to as the "root" partition)

This is the main location where the ESX operating system and configuration files are copied. If you are from a Windows background, you can see it a bit like the C: partition and folders coming off that drive like C:\Windows or C:\Program directory. If this partition fills, you may experience performance and reliability issues with the Service Console, just like you would with Windows or any other operating system for that matter.

/var

This is where log files are held. I generally give this a separate partition just to make sure that excessive logging does not fill the / file system. Log files are normally held in /var/log. But occasionally hardware vendors place their hardware management agent log files in /var.

/tmp
In the past, VMware has recommend using a separate partition for /tmp – which I have always done in ESX 2.x as well. As I have plenty of disk space I have made this larger than it really needs to be.

/opt
Several Forum members have seen the /opt directory fill up very rapidly and then fill the / partition. This location is also *sometimes* used as a logging location for hardware agents. In VMware HA has been seen to generate logging data here as well. So I create a separate partition for it to make sure it does not fill the / partition.

/**home** (Optional)

Technically, you don't need a separate partition. In the past VMware recommended one for its ESX 2.x in production. This was due to the fact that VM's configuration files such as the vmx, nvram and log were stored in /home. In ESX 3.x all the files that make up a VM, are more likely to be located on external storage. I still create it for consistency purposes – and if I have users on the local ESX server those users are more likely to create files there – than in a directory coming off the / partition.

vmkcore

This is a special partition used only if the ESX VMkernel crashes, commonly referred to as a "Purple Screen of Death.." If that happens then ESX writes debugging information into this partition. After a successful boot the system will automatically run a script to extract and compress the data to a "zip" file in /root. This file with tar.gz extension can be sent to VMware Support who will endeavour to identify the source of the problem. These PSODs are normally caused by failures of RAM or CPU. You can see a rogue's gallery of PSOD's at

http://www.rtfm-ed.co.uk/?page_id=246

vmfs
VMFS is VMware's ESX native file system which is used for storing all the files of the VM, ISO's and templates. Generally, we use external storage for this. The only case for using local storage for your VM's is when you do not have access

to external storage. So here I am assuming you have access to external storage, and therefore, you have no need for a local VMFS partition.

Hard Drives and Controllers

Obviously the hardware on the ESX host should be utilized to its fullest extent. One of the most under-designed aspects of the server is often the drives and controllers in the host. New VMware admins tend to load up on memory and processors at the expense of really hot disk controllers and fast drives. Depending on the load placed on your ESX server you may want to investigate implementing separate controllers for the service console and the local VMFS partitions. If you plan on storing VMDK files locally, this is probably a good idea and will separate traffic bound for the VM VMDK's and increase your overall performance. If you are not going to store VMDK's locally, then local disk design is not as important. The local disk will essentially only be used by a single system (the service console), and 99% of the writes to this disk will be log files. Use a simple RAID 1 and maybe a hot spare, and your service console will be well covered. A RAID 5 is obviously possible, but it could be over kill as you are adding a lot of disk space for an OS that will only consume 5-10GB.

On the controller side of things it is probably a good idea to use a hot SCSI controller with at least 128MB of cache of using local VMFS. Most of the off the shelf RAID controllers from HP and IBM now come with 64 or 128 MB of cache. It is also important to ensure that the write cache is enabled. I have seen certain instances where these settings have come from the manufacturer disabled, and I hate paying for things that I don't get to use.

If you are using local storage for your VMDK files and would like to separate them to a separate controller, but still save some money, your best bet is to use the integrated RAID controller for the service console and the additional RAID controller for the VMFS partition that will host VMDK files. If separate controllers are not possible, then shoot for a different SCSI BUS on a local controller.

As a final thought on this, we should probably talk about drive speed. There is often an argument between buying 10K or 15K drives. My opinion is up in the air. If the local drives are only used for the service console, I would say it is ok to use 10K drives and save the money. If you are going to use local disk to host the VMDK files it wouldn't hurt to move up to 15K drives for everything. This

will help to ensure optimal performance for your guest VM's and doesn't add too much cost to the overall system.

PCI Cards and PCI Slots

One of the common mistakes most engineers make when designing their hardware is choosing hardware (a form factor or model) before defining all their requirements. The engineers (or sales guys from some hardware vendor) focus on Processors, Cores, and amount of memory. But they skip the number of PCI slots, type of slots, and what you really need in the system.

For example, let's assume you have decided (after reading the book) to have 2 fiber HBAs for redundancy and two dual port nics along with the two onboard nics. This is a total of 2 fiber ports and 6 network ports on your server. Ideally (for redundancy purposes) you would maintain separate HBA's instead of getting one dual port HBA and have 2 dual port NICs instead of one quad port nic. But you have ALREADY chosen (or have had dictated to you) a model of server that only has 3 PCI slots… you obviously cannot fit 4 PCI cards (two HBA's and two NIC's) into three slots, so you wind up either with a single point of failure on the NIC's or on the HBA's and this could have been alleviated by simply defining these types of items prior to picking hardware.

PCIe,x,y,z, oh me, oh my…

So which to use and what to look for? Once you have start to dig into the PCI slots on your proposed server you will notice a number of options or specs around the PCI slots in your system, such as PCIe or PCI Express and PCI-X. Often confusing, you would think these hardware guys could come up with a better name. Anyway, PCI-X is the slower of the two and essentially is based on PCI, kind of like a PCI version 2 but PCI is 64bits wide as compared to the original PCI that was 32bit. PCI-X is even backward compatible and able to accept older PCI cards. Essentially this "older" technology is slower and shares available bandwidth on the bus. PCI Express was developed to solve some of the issues with PCI and PCI-X specifically around IO and dedicated bandwidth.

PCI-Express is a whole new ball game from a motherboard/performance perspective. While PCI-X may have similar throughput numbers on the speed side,

PCIe has a distinct advantage when you start to use multiple cards on the same bus (back to our design with multiple nics and HBA's) and can handle multiple simultaneous IO operations much better than PCI-X due to bandwidth allocation and interfaces for each PCIe device.

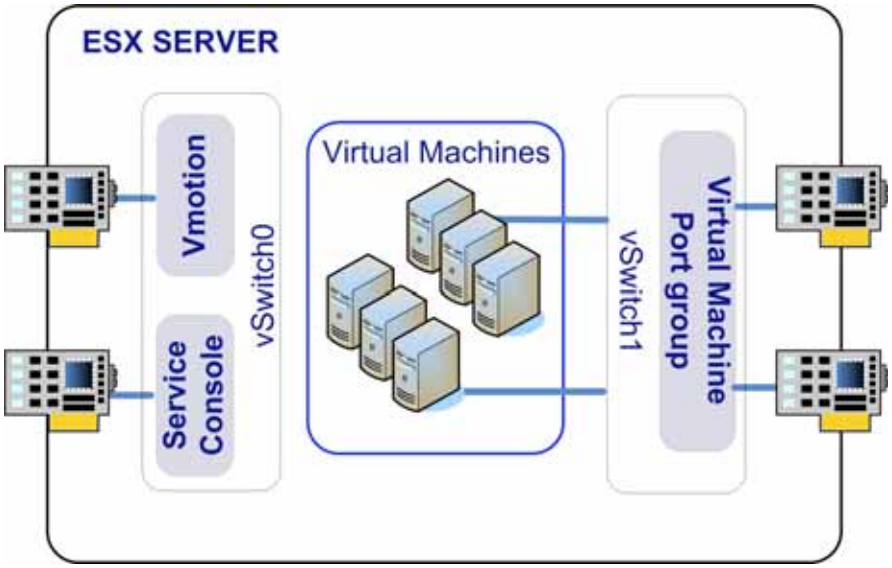
The long and short is the PCI-Express should be used where possible in ESX servers. If you have only a limited number of PCI-Express slots use them for your storage controllers/HBAs, and use the PCI-X slots for the NIC cards.

If you do not have the option of using PCI-Express and are stuck with PCI-X or limited PCI slots, not to worry. While one is better than the other it doesn't mean your system won't perform well, it's just not optimal but will more than likely perform just fine.

ESX Server Network Connectivity

Each ESX Server should, at a minimum, have at least 2 network cards. While it's possible in ESX 3.0 to run with only one NIC almost every server on the planet comes with 2 onboard nics. In a 2 NIC configuration both cards assigned will provide redundancy to the other with one card acting as the primary for the service console and another acting as the primary for Virtual Machine use. While this configuration will function, and provides a little redundancy, it limits other network based functions (like VMotion) that you may want to use and does not allow for physically separate management networks for the service console. For a minimal configuration with redundancy and VMotion functionality, you may want to design at least a 4 NIC configuration as seen in Figure 3.2.

Figure 3.2 Typical ESX Server Network Configuration



Service Console NIC Configuration

Referring to Figure 3.2, the two nics on the left are configured as part of a single virtual switch. One nic is utilized by the service console for management tasks. ESX management by virtual center, interaction with the console via SSH, etc. A port group policy (better explained in the networking chapter) is used to configure this nic as the primary nic for the service console. VMotion functions are assigned to another using the service console nic for failover. This would require that that VMotion traffic run across the production network (not as big a deal as it might seem) and if you isolate the traffic with VLAN's that port trunking be configured on the upstream physical switch ports.

Virtual Machine Port groups and Switches

In this example, the two network cards on the right combined are both configured as active NIC's on a single virtual switch. This configuration not only supplies additional bandwidth to the VMware guests, but also provides fault tolerance in the event of a NIC or upstream switch failure. It is important to size these NIC's appropriately. In this configuration we often use Gigabit network cards. Using 100 MB as a baseline for each VM, this configuration easily supplies enough bandwidth for 16 - 20 VM's. For sizing purposes it is good to assume at least 1 Gig NIC per 5-10 VM's. If fault tolerance is required, you will

have to make a decision about bandwidth availability during a NIC failure. Is it acceptable for say 16 VM's to run from a single GB NIC if one NIC fails?

From a real world perspective I personally have looked at the utilization on 1000's of servers during consolidation analysis engagements. Generally only about 1 % ever average more than 1 or 2 Mb of utilization. Sure, a handful of heavier used servers will use more, but most traffic is intermittent with VERY short spikes of any real utilization. You will be able to see this once you analyze your own environment carefully.

Virtual Switch

While it is possible to assign standalone physical nics to virtual machines for network connectivity, there is no redundancy involved. If the NIC or connection were to fail, the guest operating system would not be able to talk to the network. To prevent this downtime, ESX allows you to create virtual switches (alluded to before). These virtual switches allow you to bond or team up to 32 physical nics for use by virtual machines. For more detailed information on VM networking please jump to Chapter 6. For now, you just need to understand that you will need at least 1 virtual switch in your environment.

Real-World Network Connectivity

We definitely recommend a minimum of a three NIC configuration for most production systems. If you plan on implementing VMotion, then a fourth NIC should be used to isolate that traffic away from the NIC's used by the VM's. If you are planning to implement servers that are larger than 4-ways, you may want to look into adding another two network ports to accommodate the amount of bandwidth that will be required by the VM's or you may need to add nics to connect to separate physical networks you need to support. A good rule of thumb is 8 to 12 VM's per Gigabit network card dedicated to VM's. If you are looking at an eight core server and possibly 30, 40 or more VM's, you should have at least two and possibly three or four NIC's dedicated to VM's.

As an alternative, you can break this down into bandwidth requirements if you wish to get very granular. This requires that you estimate the amount of bandwidth required by each VM, then add that total up to determine the number of

NIC's you need. Obviously we have done this assuming 100MB or a little less per VM in using the 8-10 VM's per NIC.

Real-World Sizing Strategies

Now that you know the basics of how server hardware components work in ESX environments, you need to think about your strategy for sizing your servers.

The objective is simple: you want to build your servers to be big enough to support your VM's and keep the number of hosts you need to manage down to a reasonable level, while still making the servers small enough that you don't break the back purchasing them.

At first, this statement may seem extremely obvious. Nevertheless, there's plenty to think about when you get ready to size your servers.

Why Should You Care About Server Sizing?

Server sizing is not about buying the fastest processors and the most memory. When it comes to server sizing, the maximum number of VM's a server can support is less important than the cost for each VM on that ESX server. If you build a 16 core server that can handle 64 VM's, but two 8 core servers would have cost you 25 percent less and would have handled just as many VM's, you may find yourself out of a job. In addition, server sizing is finding a balance between hosting as many VM's as possible, while still maintaining a manageable number of hosts and keeping the cost per VM at a reasonable level.

A proper server sizing strategy involves creating a balance between too many small servers and too few large servers. For example, it's possible to build a sixteen core server with 64 gigabytes of memory. But just because you can build one gigantic server for all of your VM's, should you? There are plenty of servers out there that have 48-64 VM's on them; however the cost per VM in these solutions begins to rival the normal cost of adding a blade into a blade chassis.

Server Sizing Options

By building several smaller ESX servers, you're able to increase the redundancy of your ESX environment. In this strategy you not only build in redundant components but build redundancy through the number of systems. If you build one gigantic $60,000 server and something happens to it, all of your VM's are down. However, if you build three $20,000 servers and you lose one, only one-third of your VM's are affected.

Your server environment will ideally balance between the two extreme options:

- Build a few gigantic servers.
- Build many small servers.

Our recommendation - take the middle road.

Option 1: Build a Few Gigantic Servers

Drive space, processors, and memory are so incredibly inexpensive these days that many people are transfixed by the idea of creating a few massive servers that can each support tons of VM's. They like the concept of only having a few servers to manage and the fact that they can spend money on large chassis, redundant drives, processors, tons of memory, NIC's, power supplies, etc.

The general issue here is that the price / performance ratio of big servers might not be there. When looking at building large servers (read- 8 and 16 processor servers – potentially with multiple cores), the cost per VM when compared to a similarly configured 4-way may be 25 percent more. Of course a benefit of having large ESX servers is that you have fewer ESX servers to manage. This will reduce time spent for monitoring the host, patching, upgrading and managing the host, and of course reduce the "number" of servers in the datacenter.

Advantages of Building a Few Gigantic Servers

- Fewer hosts to manage and monitor
- Less time spent upgrading and patching hosts
- Large chassis always have plenty of PCI and memory slots for all the additional components you need to jam into an ESX Server

Disadvantages of Building a Few Gigantic Servers

- Single point (or fewer points) of failure
- Possibly an increase in cost per VM

Option 2: Build Many Small Servers

Instead of building a few gigantic servers, you might choose to build a large number of smaller servers. This option lessens the risk that one system's failure could take out a significant number of your VM's.

When thinking about building multiple smaller servers, two advantages become apparent: redundancy and scalability. Because you have multiple servers, you could lose one without a large part of your environment being down. (This means that you might not get paged if this happens, allowing for a full night's sleep.) Also, you can schedule servers to be taken down for maintenance or to be rebooted without affecting large portions of your environment.

Furthermore, you might be able to support more VM's with the same amount of money. Or, you could look at this as being able to save money. Many ESX administrators also like the fact that building multiple small servers gives them more flexibility to dynamically deploy and re-deploy VM's as requirements and resource needs change.

Another benefit in favor of multiple, smaller servers, is the ease with which servers are managed and provisioned today. Many companies are leveraging blade servers to build large farms of redundant servers. (Think of it as "RAIS"—Redundant Array of Inexpensive Servers.)

Advantages of Building Many Small Servers

- Redundancy
- Often a lower cost per VM
- Flexibility, redeploy applications and move them around as needs shift

Disadvantages of Building Many Small Servers

- Some utilization might be wasted
- Additional ESX hosts to manage
- Really small servers tend to be limited by number of PCI Slots or amount of memory you can install

Option 3: Build Servers in the Middle Ground

Finally, you have the option to move into the middle ground. Instead of going to either extreme you can find a middle ground like a quad processor server or even a dual processor dual core server. In a number of engagements we have found that the quad processor and dual processor multi-cores are the "sweet spot" when comparing price, performance, and manageability.

With all else being equal, quad processor servers often fall right into an environment's sweet spot. Quad processor servers are more of a commodity than the 8- and 16- processors servers, which lowers their prices when compared to the larger servers. In addition, they often have the flexibility in their design to allow you to configure Network cards, HBA's, and other components for your environment. This is often a huge advantage over blades and 1U's.

One big recommendation we can make is not to make a chassis selection until you design your network and storage solution. Often in designs with clients, they want to jump right to the selection of the specific model of server. We generally steer them back to deciding the number of NIC's, types of redundancy option, number of HBA's etc. Then, once we have that information you can say for sure how many PCI slots you need, and what type. This more often than not dictates the servers you can buy. When this is done in reverse, I have seen clients purchase servers with only 3 PCI slots, then want to have full NIC redundancy for 2 additional dual port nics and have 2 Fiber Channel HBA's- four PCI cards in a three PCI slot chassis.

Generally, the disadvantage seen in quads or large dual core duals is either from a political front where someone is set on one technology over another, or the fact that your ESX environment will still be too small for quads and would be better off with just a couple of small chassis dual processors.

Advantages of Building Mid-size servers

- Often sweet spot for price per VM
- A good balance between number of VM's and number of hosts
- As flexible as small servers if your environment is large enough

Disadvantages of Building Mid size servers

- Might not provide enough redundancy in smaller environments
- Additional ESX hosts to manage when compared to large servers

Installation Alternatives for Host Servers

The first step in designing your host build is to determine what is going into your base build. Regardless of whether you hand build each ESX host or you script the entire process you need to determine what you want to accomplish with the script or the build check list. Below is a list of some of the common items/tasks in the build process and in some cases (like for the agent installs) our notes on their installations:

- Hardware agents from HP, Dell or IBM
- Management and Monitoring Agents
- Anti-virus, if you must
- NTP Configurations
- Authentication modules
- Firewall settings

At a minimum we recommend you have a solid base ESX build that includes a step for installing hardware agents, NTP configuration, external authentication configuration and of course modifications of the firewall settings to make these items possible. Anti-virus, third party management or monitoring agents are always nice to have in the host system but not a full fledge requirement unless dictated by your company policy.

Basically, for the build you have three options: script the build, build by hand, or use image based builds. We outline the build options below.

Scripted Installations

Scripted installations can cover many alternatives from using a basic VMware created kickstart installation, a custom kickstart file you have created with follow up shell scripts, or scripted builds based on deployment tools like Altiris. In any of these scenarios the goal is the same; to create a repeatable installation process that removes the "human factor" from the build. A side benefit of this is that process is quicker than a normal hand installation since no one is stopping to read a document or checklist or looking for media to put in the CD drive.

The drawback of using scripted installations (in most shops I have been in) is that there are only a handful of people, if not a single person, that has the ability to edit the files. Scripting and maintaining scripted installations of ESX will require a basic knowledge of kickstart and shell scripting for Linux. Generally ESX Server is found in predominately Windows environments, and the staff lacks this knowledge. If you plan on going down this route I suggest getting a book on Linux scripting (specifically something that covers shell and kickstart basics).

Advantages of scripting the installation

- Creates a repeatable installation that removes the human factor
- Can create a build process that is faster than being built by hand

Disadvantages of scripting the installation

- To do it right it takes a basic knowledge of kickstart and shell scripting
- Has to be maintained and possibly requires numerous scripts for numerous hardware platforms

The Operations Guide section of this book (specifically the command line chapter) has the specific step by step process for creating an unattended install. We debated having processes in for deployment tools like Altiris, but decided that keeping up with changes in various products and individual environments may be a bit too much to bite off.

Hardware Agents (Dell, HP and IBM)

While most small organizations don't take often take advantage of tools like HP SIM, Dell Open Manage, or IBM Director, we firmly believe that you should in your ESX environment. Now that you are designing this consolidated environment where 10, 20 or even 40 guests may be operating on a single server, a failure on that server's hardware becomes more of a problem than a hardware failure on a single server. In most instances these tools will allow you to be notified of the issue, if not notified PRIOR to the issue becoming a full fledged failure of the system.

I have actually received calls from people (and you may have seen this too) where they had a server with a RAID 5 configuration loose a drive. They never noticed it in a system console (no monitoring tools installed) and never saw the red light on the front of the drive. The server continued to run just fine because that is what RAID was intended for. Of course it ran just fine until the second drive failed and the system crashed. Wouldn't it have been nice to know that was coming?

Anyway, below we show some install steps for each of the three major hardware vendors. These steps were tested on a base ESX 3.0 build and the version of the agent has been noted. I recommend you check your vendor and VMware's websites to determine the latest agent you should be running as these agents sometimes change on a monthly basis. Also note that these are listed in alphabetical order… we have no preference. ;-)

Thanks to the RapidApp Engineering team that put together these steps on agent configurations for different vendors.

Dell OpenManage Agent

As of this writing Version 5 is the only version certified to run on ESX 3. The agent comes as a tar ball so you will need to copy it from a central location or mount up the CD on the host (Either of which can be scripted.

HP SIM Agents

As of this writing the current version for ESX 3.0 HP Insight Management agents are 7.5.1.A. Like all agents, as this changes, these instructions may change but could act as a good starting point. To install the Insight Management agents, download or copy the hpmgmt-7.5.1a-vmware.tgz to the ESX 3.0 tmp directory. Untar the file by typing:

tar -zxvf hpmgmt-7.5.1a-vmware.tgz

or

srvadmin-services.sh start

To check to see if the agents are running point your browser to:

https://fully.qualified.name.com:2381

IBM Director (VMM)

In order to install the IBM Director agents, you need to open the ports needed for the agent, perform the installations, then start the agents. It should also be noted that IBM Director has a module called Virtual Machine Manager (VMM) that integrates with the VMware VirtualCenter application. We recommend you configure both the host and VirtualCenter portions detailed below.

To enable the ESX firewall for ibm director:

esxcfg-firewall ---enableservice ibmdirector

HP SIM Agents

As of this writing the current version for ESX 3.0 HP Insight Management agents are 7.5.1.A. Like all agents, as this changes, these instructions may change but could act as a good starting point. To install the Insight Management agents,

download or copy the hpmgmt-7.5.1a-vmware.tgz to the ESX 3.0 tmp directory. Untar the file by typing:

tar -zxvf hpmgmt-7.5.1a-vmware.tgz

from tmp/hpmgmt/751/ console prompt type:

./installvm751.sh

srvadmin-services.sh stop

or

srvadmin-services.sh start

To check to see if the agents are running point your browser to:

https://fully.qualified.name.com:2381

Conclusion

As you can see, defining what will be installed on your server is very important (there are tons of options). The Operations Guide has steps for installing all of the "normal" configurations such as NTP, firewall settings and authentication modules. Once you have nailed down what you want installed on your hosts, jump on ahead to the operations guide to build your scripted installation (don't worry, it's a script, you can always modify it later)