VMware® Infrastructure 3

Advanced Technical Design Guide

~and~

Advanced Operations Guide

Two books in one!

Ron Oglesby
Scott Herold
Mike Laverick

Chapter 5 - Storage

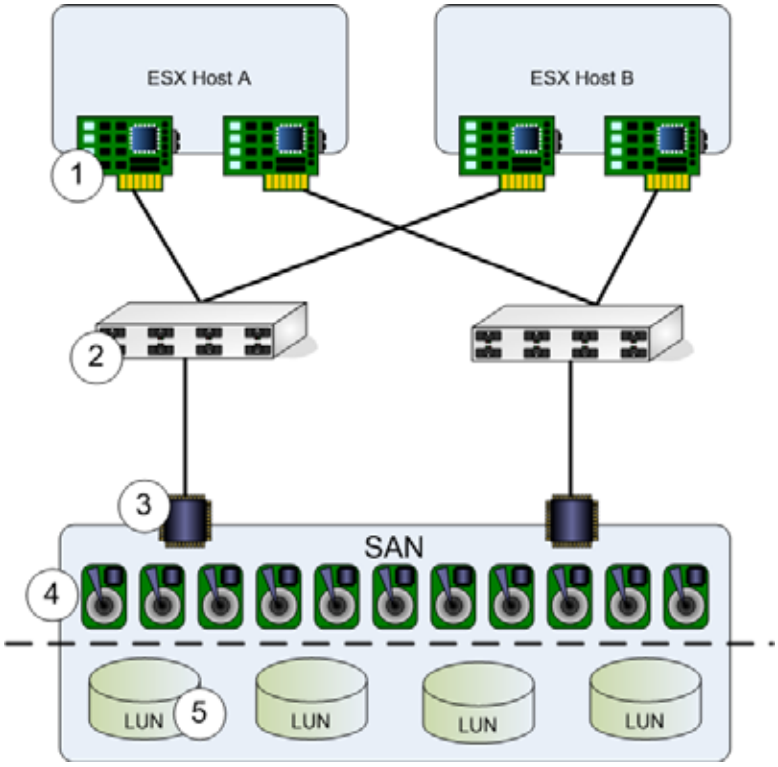
In the last chapter we went through the different local partitions and storage requirements to setup ESX server. In this chapter we dive into the real meat of ESX storage: VMFS partitions for VM disk storage. VMFS 3.21 is the current version of the VM File System created by VMware. VMFS was created to provide storage for the extremely large files that make up VMs. In turn, the requirements for large amounts of storage put a special emphasis on local and SAN based storage for your VMs. This chapter delves into the various uses of storage in an ESX environment.

Storage Components

The storage infrastructure that supports an ESX environment is often one of the most complicated and misunderstood aspects of a virtualization project. Fortunately, over the years, the popularity of virtualization has made its way down to the storage teams and there is better awareness now than there was in the past.

If following the best practices of not only ESX storage configurations, but also the best practices defined for storage infrastructures, your environment should be laid out similarly to the example illustrated in Figure 5 – 1. While this diagram illustrates a typical Fiber Channel SAN infrastructure, we will find that iSCSI follows a near identical design, only using a network communication infrastructure as opposed to fiber.

Figure 5- 1: SAN Infrastructure



Fiber HBAs

The key component that must be installed in the ESX server for fiber-based SAN storage connectivity is a Fiber HBA. According to the VMware documentation there is a physical limit of 16 HBAs that can be installed in a single system. Again, this is overkill. Except for the most intensive transactional workloads, two HBAs should suffice for your host configuration. The most we would ever recommend putting in a single system is four, assuming you are looking to virtualize Messaging or Database applications that require more dedicated throughput.

Fiber HBAs come in many flavors. We are typically seeing 1 Gb fiber HBAs being phased out and slowly being replaced by 4 Gb fiber, but a solid majority the implementations we are performing are using 2Gb fiber. A single 2 Gb fiber adapter gives us a theoretical maximum bidirectional throughput of about 400MB/Sec, meaning you can concurrently read AND write at about 200MB/Sec each, not that you can read OR write at a total of 400MB/Sec. 1Gb adapters will be about half that speed, 4Gb adapters will give you about double.

iSCSI HBAs

When planning to use iSCSI in a production environment you will want to leverage iSCSI HBAs. These adapters are used to encapsulate Standard SCSI commands inside TCP/IP packets for remote block level storage. By providing adapters specifically for this purpose you will offload the processing required for the storage infrastructure to the hardware.

iSCSI HBAs must leverage a network infrastructure to communicate with backend storage. A true production iSCSI implementation will have its own independent network infrastructure so as not to interfere with regular network communications. Based on this fact, the maximum speed per path in an iSCSI implementation is 1 Gb with a maximum capability of two iSCSI HBAs (paths) per host. While this may not be enough for the most intense of enterprise workloads, it is definitely nothing to ignore based on the cost per performance you receive with an iSCSI infrastructure. Based on the imminent release of 10 Gb networking not far off for VMware ESX, we are expecting these speeds to increase drastically and give some significant competition to the performance of fiber-based SAN infrastructures.

SAN Switches (Used with Fiber HBAs)

SAN switches are not unlike network switches in that they provide a mechanism for a multiple hosts to communicate to a centralized storage infrastructure. The configuration and use of SAN switches typically falls within the responsibilities of the storage group in an organization. It is highly unlikely that you will come across a configuration that does not use multiple SAN switches, thus providing multiple paths to your SAN infrastructure. The importance of the storage infrastructure definitely demands a very highly available solution to keep hosts and storage arrays constantly communicating with one another.

Network Switches (Used with iSCSI HBAs and NFS)

Network switches come into play when using a form of network based storage for your virtual infrastructure such as iSCSI or NFS. It is highly recommended that a separate network infrastructure from the production communication network be used for the storage infrastructure. This ensures proper flow of incoming and outgoing network communication traffic does not interfere with disk activity, and vice versa. As a worst case scenario, a separate VLAN should

be used for the storage infrastructure to ensure all the Windows broadcast garbage stays away from storage communication.

The use of network switches is discussed in significantly more detail in Chapter 6 where we discuss networking in ESX Server.

Storage Arrays

Storage arrays are the key to a centralized, redundant, and high speed virtual infrastructure. Storage arrays are used to manage and configure very large amounts of storage for enterprise applications such as Database, Messaging, and of course, Virtualization. VMware ESX supports fiber based and iSCSI based storage arrays across a wide variety of vendors.

While the speed and capabilities vary widely between the various vendors and models of storage arrays, there are several key components that are generally found regardless of who is providing your centralized storage.

Storage Processor (sometimes called controllers)

Both Fiber and iSCSI storage arrays contain at least one, and optimally two storage processors (SP). Storage processors manage the front end access of hosts back to actual SAN storage LUNs. SAN's typically come in one of two configurations: Active/Active or Active/Passive.

Active/Active arrays are ones in which a single LUN may be accessed down either or both storage processor concurrently. This allows for the greatest flexibility in load balancing I/O communication down multiple paths. It should be noted that ESX cannot take full advantage of Active/Active SANs, as there is a limitation in the VMkernel itself which only allows ESX to communicate to a given LUN down a single path at any point in time. It is possible within ESX to determine which HBA a LUN uses in an Active/Active configuration at the VMkernel level, making Active/Active arrays the optimal choice for load balancing your SAN environment for your virtual infrastructure.

Active/Passive arrays are those that only allow access to a given LUN down a single path at a time. Multiple LUNs may be accessed down alternate storage

processors. The SP that is not active for a particular LUN will act as a failover SP for that LUN. It is possible to dictate which SP is preferred on a per LUN basis on the SAN, but this cannot be specified on the ESX host.

Physical Disks

Prior to virtualization, many Windows administrators could have gone their whole life thinking that a SAN consisted of some REALLY large hard drives. The truth is, SANs are made up hard drives that are typically smaller than those provided in new server deployments; there just happens to be a lot of them.

While we won't go too far into actual SAN management and configuration in this book, it is important to inform ESX administrators what it is that makes SANs so fast and fault tolerant. The simple answer is "Lots of disks". SANs provide volumes to hosts by using typical RAID striping mechanisms across a large number of disks. Typically, the more disks (spindles) that are provided, the faster the disk access will ultimately be. There is, of course, a point of diminishing return which is based on your actual SAN infrastructure and how the storage processors communicate with the disks (Typically internal Fiber or Serial Attached SCSI).

Some of the more critical data LUNs not only use RAID striping, but also mirroring to create some unique SCSI configurations such as RAID 1+0, in which each disk has a mirrored pair and the data is then striped across all mirrored pairs.

It is not uncommon for an implementation to call for multiple RAID configurations to be leveraged for various purposes in their infrastructure. As an example, a set of RAID1 volumes may be created for the sole purpose of hosting operating system partitions, as performance is not important, but redundancy is. A set of RAID5 volumes are then created for standard data partitions where speed is critical as well as lightweight protection from disk failure. The most critical of data volumes may be configured in a RAID10 configuration, which provides not only high performance, but also an extremely high level of protection of the underlying data. It is very important for the ESX administrator to understand the backend configuration of the storage infrastructure to properly match the role of the virtual machine disk files to the proper SAN volume on the backend infrastructure.

LUNs

Knowing what we now know about the backend storage infrastructure we can describe what a LUN is. LUN itself stands for "Logical Unit Number" and refers to a logical device that is presented to a host as a single physical disk. LUNs in an ESX infrastructure are typically assigned to multiple hosts to enable VMotion, DRS and HA functionality. When we discuss VMFS partitions later in this chapter you will have a better understanding as to how this is possible.

Each LUN is assigned a LUN ID when it is presented to the ESX hosts. It is extremely important that these LUN IDs stay consistent across all hosts in the ESX cluster; otherwise it is possible to run across some strange disk signature errors on certain hosts. It is also important that each LUN be given a unique ID. Any two LUNs that are assigned to the same host with the same ID will be assumed to be the same LUN. ESX will natively provide failover capabilities in this instance, and if it has to fail over it would be quite bad for your data not to exist on the second LUN. For this reason, if a LUN is presented down multiple paths to the same host, it must be presented with the same LUN ID down both paths.

A total of 256 LUNs may be assigned to any single ESX host with a maximum size of 2TB per LUN. In addition, you want to limit the number of hosts that can access a single LUN to prevent a bottleneck when accessing the data. Based on some of the other limitations of building a virtual infrastructure, the greatest number of hosts that should ever theoretically have access to a single LUN is 32, which is the maximum host size for a DRS cluster. Trying to leverage LUNs across multiple large DRS clusters is not a good idea and will eventually result in choking your bandwidth to the point that the LUN is unusable.

LUNs can be used in several ways, which we will discuss throughout the duration of this chapter.

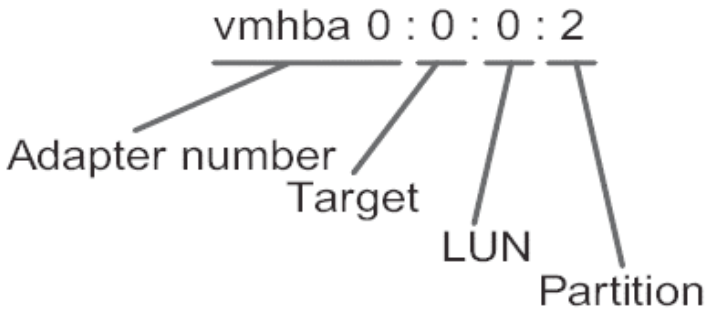
A single LUN can be configured with a VMFS file system

Multiple LUNs can be spanned together to form a single VMFS file system

A single LUN may be configured for direct access by a virtual machine without using the VMFS file system (This is called a RAW Device Mapping, or simply RDM)

Identifying LUNs in VMware ESX Server

You may have already been poking around ESX and noticed the number for your RAID controller or Fiber cards; it looks something like this:



Before we get to deep into this you need to understand first how VMware identifies the HBAs in your system. This numbering scheme can be seen from either the Virtual Infrastructure Client or from within the service console. Generally when dealing with SAN LUNs or other disks presented to the server, you will see the first three sections of this identifier as ESX sees it: a simple disk attached to the system via a Virtual Machine host bus adapter (vmhba).

The first section shows the adapter number, the next section shows the SCSI target ID, and the final section shows the LUN number. When this numbering scheme has four sections, the fourth section is used for partitions on specific LUNs—not by the adapter itself.

Storage Types

So, I know I want to implement ESX, but have not decided on the storage solution yet. Which should we use in our environment: Local SCSI disks or remote SAN, iSCSI, or NFS storage? Well the answer may well be a combination of the various available technologies. While it is possible to run ESX without a local

VFMS partition, that doesn't mean it's the best solution. Also, while it is possible to use all local VMFS partitions for virtual machine storage, it may not offer the flexibility you need in your environment.

Local SCSI

In many small environments local SCSI disks are how ESX is first implemented. Then as disk needs or the environment grows, SAN connectivity is implemented and the VMs are moved to the centralized storage.

The primary reason for using local storage for your virtual machines is cost. When comparing local SCSI disk costs to that of SAN storage, local disk is much cheaper. In addition, if you are a small environment that does not have a SAN right now, the initial implementation of a SAN can be a substantial investment.

The major drawback to not using centralized SAN storage is that you lose the ability to move virtual machines quickly and easily from host to host. Touted VMware features like VMotion, DRS, and HA require that you have a SAN infrastructure in place to use them. In addition, if a host fails and you are using local storage, you must recover the virtual machine disk files from either a backup solution or some other form of network storage used for disk file backups. Where, if you had a SAN, you could simply restart the virtual machines using the existing disk files on a new host.

Configuring SCSI Controllers

The first thing you need to be aware of is that the VMkernel only supports a specific set of controllers. The current list of supported controllers can easily be found on VMware's support site and should be referenced before configuring hardware for ESX.

The next thing you will have to decide on is how to configure your controller and the options available to you. If you have a controller capable of creating multiple arrays, you have a lot of flexibility. If you have multiple controllers in your box, you are really set up.

Like database servers, it is best from a performance perspective to split up the operating system and the application. In this case (though it is a crude analogy) it is best to split up the service console from your virtual machines. Having two controllers will allow you to store your service console on one controller and your virtual machine disks and configurations on another. If you only have one controller with the ability to create multiple arrays then that is the next best option you have. Split the service console onto the first array and the VMs onto the second. Spread the VMFS partition for the VMs across as many spindles as possible. A RAID 5 would be my recommendation for the VMs while a simple mirrored set (RAID 1) is more than sufficient for the service console.

If you don't plan on storing VM disk files on your local storage, then a single RAID controller – even if it only supports a single array – will suffice. But this section is really focusing on the configuration for the local storage of VMs.

Advantages of using local SCSI Storage:

- Cost. Local SCSI storage will always be cheaper than a SAN solution
- Simple to implement; requires no special SAN experience of any additional configuration within ESX

Disadvantages of using local SCSI storage:

- Makes recovering virtual machines from a failed host more complex and time consuming
- Eliminates your ability to use features like VMotion, DRS and HA
- Reduces the overall scalability of your implementation

Remote Storage

The exact opposite advantages and disadvantages of local storage applies to Remote storage for ESX. While a remote solution requires additional expense and configurations, the benefits are enormous.

Remote VMFS partitions allow you a number of benefits over local storage. The simple use of VMotion, DRS and HA is a huge benefit to any environment. But add on top of the ability to have a fast, central repository for virtual machine templates, the ability to recover virtual machines on another host if you

have a host failure, the ability allocate large amounts of storage (we're talking terabytes here) to your ESX servers, and the list goes on and on.

The real idea here is that a remote implementation offers you a truly scalable and recoverable ESX solution. The options and features available to remote storage users are enormous, while the limitations found in local SCSI storage are like an anchor around an ESX admin's neck.

In this section we will discuss Fiber and iSCSI SAN configurations as well as NFS shares for VMFS partitions. For now let's just look at the advantages and disadvantages of using remote storage for your VMFS volumes.

Advantages of using Remote Storage

- Allows for VMotion, DRS and HA functionality
- Allows for a centrally maintained template location
- Allows for fast recovery of virtual machines in the event of a host failure
- Provides a more scalable storage solution for your environment
- Allows for VMs to be clustered across hosts (Fiber Only)
- Allows for physical to Virtual Clustering (Fiber Only)
- Provides the ability for your virtual machines to use RAW disks, just like a physical machine

Disadvantages of using Remote Storage

- More complex to manage and implement
- More expensive than local storage

Fiber Channel

Fiber Channel SAN is the most common type of storage used for VMware ESX implementations due to its performance and scalability. Fiber Channel SANs provide block level access to storage LUNs which allows ESX to have direct access to the physical disk. As the name implies Fiber Channel infrastructures connect the host to the enterprise storage through high speed fiber connections. The current maximum supported speed for Fiber Connectivity in ESX is 4 Gb per Fiber HBA, which equates to roughly 800 MB/Sec of bidirectional

throughput (~400 MB/Sec Read and ~400 MB/Sec Write). By loading up two 4 Gb HBAs in your system and properly balancing your data LUNs down multiple paths you can easily achieve the required performance of 95% of your infrastructure through shared bandwidth, which is quite impressive.

A major benefit of a Fiber Channel storage solution is the fact that the entire storage infrastructure is separate from the network infrastructure. The only communication flowing across a proper SAN infrastructure is disk I/O. Not only does this help from a performance standpoint, but also from a security standpoint as well.

Advantages of Fiber Channel Remote Storage

- Has been around a long time and is in use within many organizations
- High performance access for multiple servers
- Standalone infrastructure to support disk access
- Block level access to underlying LUN

Disadvantages of Fiber Channel Remote Storage

- Supporting infrastructure is more costly than alternate remote storage options

iSCSI

While it was possible to configure virtual machines to communicate with iSCSI disks in previous versions of VMware ESX, it typically had poor performance overall. With ESX 3, VMware has built iSCSI access into the VMkernel to provide the same level of block access to LUNs as a SAN infrastructure. This means that ESX can now use iSCSI SANs to create LUNs and configure VMFS file systems. At this point, iSCSI can be leveraged in one of two modes; through a hardware initiator, which requires specific hardware for storage communication, or through a software initiator that is integrated into the VMkernel network stack.

The iSCSI protocol takes standard SCSI calls and encapsulates them into TCP/IP packets for transport to a remote device. Because of this, iSCSI often requires additional overhead because of its dependency on the TCP/IP network stack. By using the proper hardware initiator with a TCP Offload Engine

(TOE), you can significantly reduce the processing overhead incurred from iSCSI by letting the hardware do the menial task of managing the TCP/IP stack. This option is not available in a software initiator, and the overhead of managing the network stack is managed by… yup, the VMkernel.

A maximum of two iSCSI HBAs can be installed in any given host; each currently capable of speeds up to 1 Gb/Sec. This equates to a theoretical bidirectional throughput of about 250MB/Sec per HBA, which is a bit slower than Fiber Channel. Technology is already well down the path of providing 10Gb networking capabilities to ESX and the VMkernel, at which point iSCSI SANs may start outperforming some of the more expensive fiber based solutions in place. It will probably be a slow and costly process for organizations to start implementing a 10Gb solution in their environments, and while VMware will support it, we don't anticipate wide adoption for some time.

It is entirely possible to configure iSCSI communication on your regular IP network, but definitely not recommended. iSCSI infrastructures should consist of a separate physical network as to not interfere with regular network traffic, and vice versa. Not only will physical separate provide the best performance, it will also provide the best security for your iSCSI I/O infrastructure.

Advantages of iSCSI Remote Storage

- Excellent cost for performance
- Provides centralized storage over a low cost communication infrastructure
- Block level access to underlying LUN
- Will soon exceed the speed of Fiber Based SAN solutions

Disadvantages of iSCSI Remote Storage

- Performance not yet to that of Fiber Based storage
- Will be costly to upgrade to 10Gb networking for higher speed
- Often a newer component to many organizations

NFS

In addition to supporting iSCSI, VMware ESX also has built in support for NAS devices through the use of the NFS protocol. This allows for a second

method of providing low cost network based storage to your virtual infrastructure. When ESX 3 first came out and people started dabbling with NFS support, there was a quick judgment made that it should only be used for storing virtual machine template files and that the performance was substandard for virtual machine utilization.

It has been found, over time that NFS actually performs extremely well in a properly configured and structured environment. NFS has been around for a good 20+ years now and has proven to be a very reliable centralized file system with performance that only improves over time based on network enhancements. Initial testing results that have come back show that not only can NFS be used for running virtual machines; it is actually extremely fast when using some of the newest available hardware.

An ESX host has the capability to map up to 256 NFS volumes, but there are several reasons we wouldn't recommend this that we will discuss in the next section. For a SMB organization looking to get started in virtualization, leveraging NFS is an extremely low cost way to provide a centralized virtual infrastructure that has support for the advanced virtualization technologies such as VMotion, DRS, and HA. VMware addresses a major challenge of virtual machine sizing in NFS implementations by allowing the use of thin provisioned VMDK files. This reports to ESX that the full size of the disk is being utilized, but in actuality, data is provisioned and used only when requested by the guest. This allows virtual machines to be stored in an optimized fashion for use on NAS devices, which often have stricter limitations surrounding free space than other types of centralized storage.

Access to NFS volumes is performed through the VMkernel network stack. Like iSCSI, it makes sense to provide a separate network infrastructure for your NFS for ESX implementation. It is often not possible to do this, as NFS infrastructures have typically existed for some time in an organization. Changing the way NFS is managed specifically for a virtual infrastructure will ensure network activity meant for storage resources does not interfere with network activity for regular communication and vice versa.

Advantages of NFS Remote Storage

- NFS has been around for 20+ years and is a very stable and reliable protocol

- Often an existing component in most organizations
- Defaults to thin provisioned VMDK files for optimized storage utilization
- Provides centralized storage over a low cost communication infrastructure
- Will increase in speed as 10Gb networking is more widely adopted

Disadvantages of NFS Remote Storage

- Performance not yet to that of Fiber Based storage
- Contains more overhead than Hardware iSCSI Initiators (No TOE)
- Will be a costly upgrade to 10Gb networking for higher speed

Limitations of Network Based Storage

There is a common architectural issue with all types of network based storage in the fact that there is a single bottleneck in the uplink to the storage device. In many virtual infrastructures there could be as many as 20-25 hosts, each with multiple gigabit connections, communicating with a centralized storage infrastructure. If there is a lot of disk activity across the environment it will not be uncommon for the generated throughput to be greater than what is physically connected to the storage system. In this event you will see a spike in dropped network packets on your ESX hosts. If you notice performance problems in a network based storage infrastructure, look at the logical aspect of your throughput as a first step. It may just be that you have oversubscribed your SAN connectivity. Resolving these issues will require a plan of rebalancing network and disk I/O to fit within the constraints of the storage infrastructure, or possibly even changing to a different type of storage infrastructure for specific virtual machines altogether.

VMFS Intro

The primary storage for virtual machine information is the VMFS volume. VMFS is a unique journaling file system created by VMware as a low overhead, high performance file system. The primary concern when they created this file system was to be able to create a file system that could handle extremely large

files, and provide as near to disk access speeds for virtual machine usage as possible

The VMFS-3 file system has some unique characteristics and limitations that make it different than traditional file systems.

- Ability to span across multiple disks/LUNs
- Ability for multiple ESX Servers to concurrently access files on the same VMFS-3 volume
- Up to 256 VMFS volumes per ESX system
- 2 TB per physical extent
- Up to 32 physical extents per VMFS volume
- Maximum size of 64TB VMFS size (using extents)

So what does this all mean to you? Let's review each of these features quickly and see what they mean.

Ability to span across multiple disks/LUNs

So what this really means is that you can create a single VMFS volume that spans multiple LUNs or multiple disks. The benefit here is obvious: you can create enormous volumes. It is important to note that creating a volume with multiple extents does not stripe the data across multiple spindles; it simply extends the volume onto additional disks. Thanks to the fact that VMFS-3 now uses LVM to manage volumes, losing one disk in the extent will not cause the entire volume to fail, but it can be difficult to predict which virtual machines won't work and troubleshoot issues when one of the extents fails.

Ability for multiple ESX Servers to concurrently access files on the same volume

This is really the functionality that enables VMotion and VMware HA in the event of a host failure. The idea here is that your ESX hosts can all see the same LUN(s). Because of this, the host almost becomes irrelevant when it comes to storing disk files. Since multiple servers can use the same LUN, you can simply

stop a VM on one host and start it on another. Or if the host fails, you can simply point a different host to the existing configuration files.

This is made possible since the VMkernel does not lock the entire VMFS file system. The locking is done at a file level, which allows multiple hosts to access the same volumes, but only allows one host at a time to have access to a virtual machine's VMDK file.

Up to 256 VMFS volumes per ESX System

This one is pretty obvious. 256 VMFS volumes can provide a lot of space. Generally we see ESX systems with no more than 10 or 15 VMFS volumes exposed to them. So having the ability to go to 256 allows you more scalability than you may need. We will discuss why coming anywhere near this limit is definitely not a best practice when building a proper infrastructure when we talk about sizing strategies later in this chapter.

2 TB Per physical extent and 32 extents per volume

These two are pretty much interwoven. The first means you can have up to 2 TB per LUN exposed to ESX that you are going to create a VMFS volume on. And if you have a need to create a single large VMFS volume, you can have up to 32 different physical extents (read LUNs) that comprise a single VMFS volume. Using 2nd grade math we can quickly determine that the physical maximum for a single VMFS volume is 64TB when using the maximum number of maximum sized extents.

Differences between VMFS2 and VMFS3

Outside of the primary difference of VMFS-2 being a flat file system and VMFS-3 having an actual directory structure the major differences between the file systems have to do with size and reliability. Nearly every configurable maximum is increased in VMFS-3. In addition, the introduction of journaling which provides better data consistency recovery from system crashes.

A final item to note that will become quickly evident to ESX users is that the canonical VMHBA names for LUNs and volumes have been replaced with

UUID signatures for volume management. In VMFS-2 there were issues where the same LUNs were given different IDs across different hosts. VMware ESX 3 will now use the unique UUIDs for management of these volumes.

While ESX 3 does have the capability to load and read VMFS-2 volumes, it cannot perform write operations, thus making it impossible to run VMFS-2 based virtual machines on ESX 3. The VMFS-2 functionality is provided solely to allow ESX 2 to ESX 3 upgrades.

Now you understand the basics and limitations of VMFS. It's a high performance file system made to handle a small number of large files. Now it's time to get down into the how's and why's of VMFS. Let's take a look at what knowing all this information allows us to actually do.

Naming Standards for VMFS Volumes

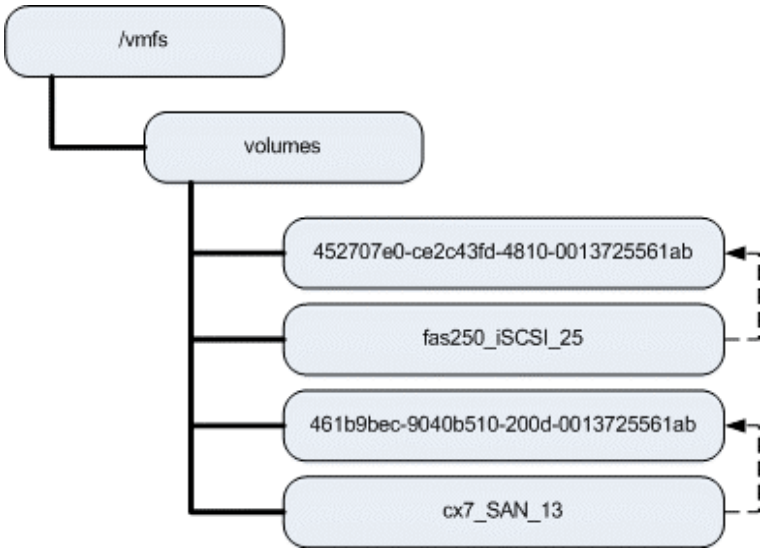
One thing that will become quickly apparent, if it hasn't already, is that we are big fans of incorporating naming standards into any user namable component within the virtual infrastructure. When creating a naming standard for your VMFS volumes you should use something that makes it quickly identifiable not only to you as an ESX administrator, but also to a storage administrator. Naming something "VMFS0", "VMFS1", etc does not make troubleshooting easy. Use something a little more conventional such as "cx7_SAN_13", which would indicate I am using an EMC CX700 storage array and my VMFS volume is on a SAN LUN with an ID of 13. The same works for iSCSI as well. Take "fas250_iSCSI_25" as example of me using a NetApp FAS250 iSCSI LUN with an ID of 25.

Since NFS is a network file system based solution, a slightly different approach is needed to identify an NFS mount. Assuming all ESX mount points are in the same root directory on the NFS server, I use the simple naming convention of "Servername_MountPoint". This allows me to quickly and easily identify the NFS server or device and the exact storage mount point I am using to store my virtual machine data.

Default File and Directory Structure

We've stated earlier that unlike previous versions of the VMFS file system, VMFS-3 is no longer a flat file system and contains an actual directory structure. As we will find out later in this chapter, this is a key feature that is required for the proper operation of VMware HA. The root path for all VMFS volumes on an ESX host is /vmfs/volumes/. Each VMFS volume is assigned a unique UUID and has a listing under this root path. If you set up a friendly name as instructed in the "Naming Standards for VMFS Volumes" section, you will also see a symbolic link that points your friendly name back to the ugly UUID.

Figure 5- 2: Directory Structure



Directories

Each virtual machine or template that is created in the infrastructure is given its own directory on the VMFS volume that was specified as the virtual machine's datastore during the creation of the virtual machine. The directory name will match that of the virtual machine specified during configuration. If multiple VMDK files are specified after the initial creation of the virtual machine that span across multiple LUNs, there will be a virtual machine directory on each VMFS volume that contains a VMDK file. This model allows for easy tracking of your virtual machine's configuration information, even if it does span multiple volumes within the infrastructure. By default, there are no other subdirectories contained inside the virtual machine's configuration directory.

Files

The VMDK files aren't the only objects contained in these virtual machine directories, there are several files that are critical to the operation of the virtual machine. Since most of these files have been touched on in other areas of this book, I will only lightly discuss them here.

VMX – The virtual machine configuration file that specifies the detailed configurations of CPU Count, Memory, Disk locations, and other critical options for the virtual machine

NVRAM – Contains the BIOS configuration information for the virtual machine

LOG – Contains the log information of the virtual machine

VSWP – The VMkernel swap file that is used for the virtual machine when a host has over allocated its memory resources

VMSD – Contains information about the location and path to the VSNP files when using snapshots

VSNP – A snapshot file indicating the VMDK file is in a non-persistent mode and that changes are being written to this snapshot file

VMDK Disk Format

In order to create and assign hard drives to your virtual machines when using centralized VMFS storage you will need to leverage VMDK files. A VMDK file represents a physical hard drive device that is presented to your guest operating system. When connecting a new VMDK file to a virtual machine, the operating system sees it as a non-partitioned physical drive. This drive can be partitioned and formatted using a variety of file systems including NTFS and ext3. Up to 60 VMDK files can be assigned to any guest operating system, allowing for quite a bit of flexibility in creating a partitioning scheme in the guest operating system. When sizing a VMDK file, ESX has a limitation of 2 TB per VMDK.

A typical VMDK file consists of two files on the VMFS file system. The first is the header file, which contains critical information about the disk such as its size and geometry. It is typically a very small file and can be read in any plain text editor. The second file is the flat file which contains your actual virtual machine data. The flat file will always report as being the same size as the size of the hard drive you specified when building the virtual machine. ESX is unlike Workstation and Server in the fact that when a VMDK file is created you have no choice but to assign all disk space at the time of creation. This keeps the file system clean and sequential for VMDK file access. There is one exception to this rule and we will discuss that when we talk about thin provisioning VMDK files later in this section.

Creating VMDKs with Different Provisioning Methods

There are actually four different methods of provisioning VMDK files (outside of Raw Device Mappings, which we discuss later) in ESX server. Each of them provides unique functionality around creation time, access speed, and disk space allocation.

Zeroed Thick

Creating a zeroed thick VMDK file is the default action when creating a new VMDK file from the Virtual Infrastructure Client or with the "vmkfstools –c" command. At creation time, a zeroed thick VMDK file is completely allocated at creation time, but existing data blocks are not wiped clean immediately. When a virtual machine attempts to read data on one of these data blocks for the first time, the block is filled with random data. This means that the VMDK file can be very quickly created, and the data that may have been on the disk prior to being formatted with VMFS cannot be read by a guest operating system. Since this data must be written to the disk before it can be read by the guest, there is a small performance hit the first time the data block is accessed. This action only occurs once per data block, so if the same block were to be changed a second time, it would not need to be wiped clean.

Eager Zeroed Thick

Creating an eager zeroed thick VMDK file can only be done with the vmkfstools command and specifying the "-d eager zeroed thick" option. The difference between eager zeroed thick and zeroed thick lies in the fact that an

eager zeroed thick VMDK file will have all data blocks zeroed out when it is created. This means the VMDK file will take longer to create than a zeroed thick disk. The exact amount of time is dependent on the size of the disk and the speed of the underlying storage infrastructure. Creating an eager zeroed thick VMDK file will not incur any performance penalty the first time a data block is accessed, as the data has already been zeroed out. If creating a VMDK file for a transactional system such as a database, messaging or file server, it makes sense to manually create your VMDK file with this option from the service console.

Thick

I'll start by stating that using this creation method is a major security risk and it should not ever be used. Instead of just calling it quits there and saving myself some writing, I'll actually explain why such a bold statement is necessary. When you create a thick VMDK file, it is done in the same fashion as creating a zeroedthick file. All space is allocated when the file is created, but no data is wiped clean. The downside is that it is not wiped clean the first time it is accessed either. Any data that may exist in the data blocks that make up the VMDK file can quite easily be accessed through easy to write code or easy to find undelete utilities. If you care about data integrity in your environment (typically Sarbanes Oxley or HIPPA will dictate that for you), never ever create a thick VMDK file.

Thin

A thin VMDK file is a new and unique method in an ESX environment. It is the default creation option if a VMDK file is created on an NFS volume. What it means is that data blocks are allocated and zeroed out as they are used, making it the most efficient in regards to storage space utilization. The down side of this becomes evident if you have multiple thin VMDK files on the same NFS or VMFS volume through fragmentation. Since data blocks are only assigned when they are used, if multiple files are being created, moved, etc in the same volume, your performance is going to suffer because these files will become extremely fragmented extremely quickly. This performance impact isn't as significant in an NFS environment, as the block data is abstracted from the ESX host, but it is recommended that you not use thin provisioning of a VMDK file when using VMFS volumes on your ESX host.

Snapshots

Snapshots are not an entirely new component in ESX. Previous versions used a very simplistic REDO file method to create a point in time snapshot of a virtual machine. The legacy REDO files were only linear and there was a maximum limit of two concurrent REDO files for any given VMDK file. The new snapshot format is identical to that used in VMware Workstation and VMware server. When using snapshot files you can not only create a linear checkpoint structure, but can also create complex parent-child relationships to provide significant flexibility around various recovery or testing points. Snapshots are taken at a virtual machine level, whereas REDO files were taken at a VMDK level. This means that when a snapshot of a virtual machine is created, all compatible disks are captured in that snapshot.

A single virtual machine can contain up to 32 levels of snapshots, which should provide for even the most complex configurations of a virtual machine. A typical snapshot consists of three major components:

Guest Memory State (Optional) – Performs a memory dump to disk of a running virtual machine so it may be recovered to the exact point and power state at the time the snapshot was taken. If guest memory is not included with the snapshot, the server will boot from scratch in a crash consistent state.

Guest Configuration State – Captures the configuration of the virtual machine hardware such as number of processors, assigned memory, virtual hard drive configuration, etc.

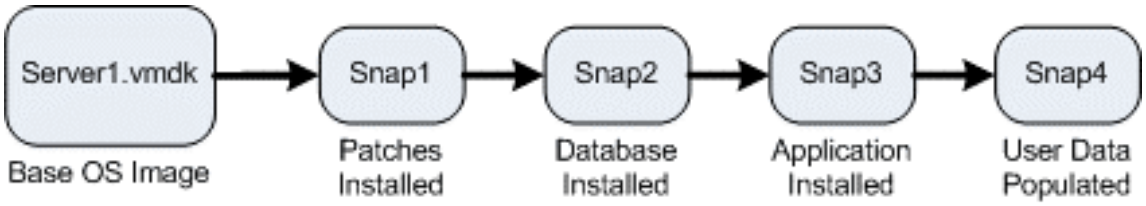
Guest Disk State – Cuts off disk access to the parent disk and creates a snapshot disk file (child). Any new file system changes are written to the snapshot file and the parent is not modified until the child snapshot is deleted or the user reverts back to the checkpoint.

Linear Snapshots

Linear snapshots are the easiest to visualize and manage in a virtual infrastructure. A linear snapshot configuration consists of the same virtual machine with several point-in-time checkpoints that do not branch in different directions. Each snapshot has one parent image and one child image, with the exception of

the primary VMDK file not having a parent and the final snapshot not having a child.

Figure 5- 3: Linear Snapshots

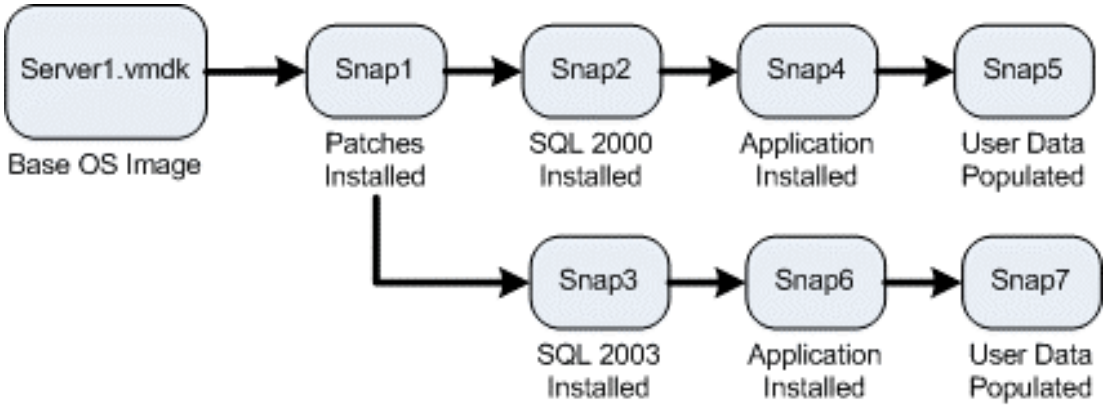


The key to applying a name to a snapshot is to describe the activity that occurred up to the point in time that the snapshot was taken. As an example from the image above, the database was installed between the time after Snapshot 1 was created and Snapshot 2 was taken. If you want to revert back to a point in time and install a different database platform, you would need to revert back to Snapshot 1, which invalidates any of the data that occurred from that point forward in a linear format. This is where nested snapshots come into play.

Nested Snapshots

Nested snapshots function on the premise that a parent VMDK can have more than one child, but a child can still have only one parent. This allows multiple configuration branches to be formed for more complex configuration and test scenarios.

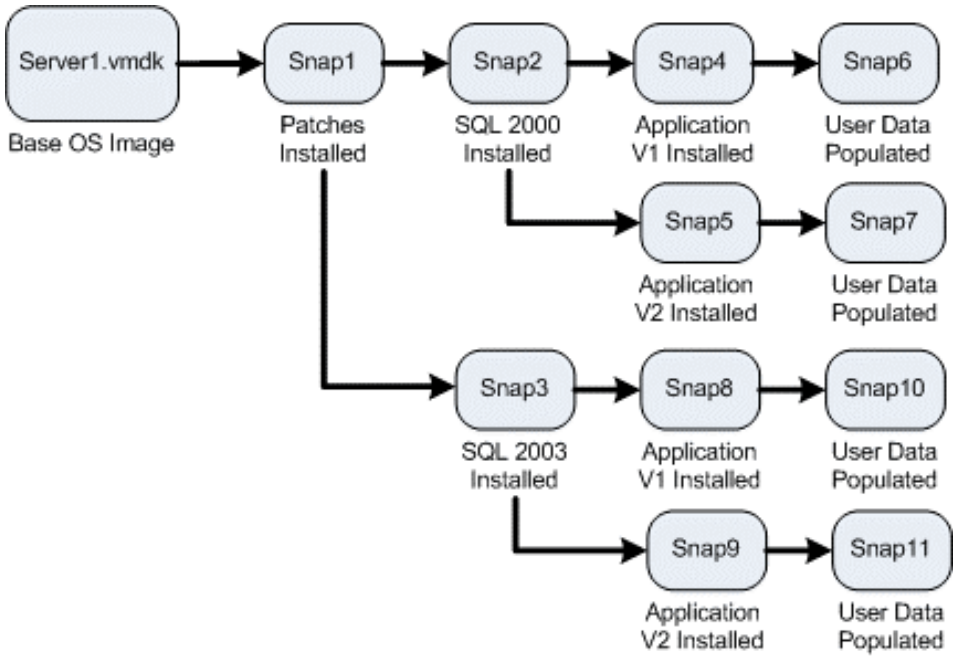
Figure 5- 4: Nested Snapshots



In this configuration we can see that a new branch has been made by creating a second child snapshot to the Snapshot 1 parent. This allows a tester to more easily switch back and forth between not only different database platforms, but also the application and user data that was added to that platform. By simply flipping between Snapshot 5 and Snapshot 7, the user can completely change their database environment to perform testing against otherwise identical configurations.

We can take a look at one final example to show how easily a nested snapshot infrastructure can get in what most would consider a typical test scenario for an application developer.

Figure 5- 5: Advanced Nested Snapshots



As you can see here, we have not only nested our database with two different versions like we did in our previous example, but we have also introduced two versions of our application. By changing between Snapshots 6, 7, 10, and 11, a user can very easily test four unique scenarios quickly and easily on an enterprise ESX infrastructure.

When using Snapshots, you want to be extremely conscious of the amount of free space you have in your VMFS volumes. Snapshots constantly grow as data inside the virtual machine changes. Devastating results have been known to occur if a VMFS file system is filled to its maximum capacity and snapshots are being used on virtual machines. If you intend to use snapshots for activities such as backup or creating multiple checkpoints, make sure you plan a sufficient amount of free space into your VMFS volumes to support the rate of change in your virtual machines.

Independent Mode Disks

There may be a situation in which you would like to exclude a particular disk from a server snapshot. This is typically in a case where you only want to capture the base operating system for recovery purposes and do not want to capture a large data volume. You can specify individual VMDK files that you wish

to exclude by changing them to Independent Mode VMDK files using the Virtual Infrastructure Client.

Persistent and Non Persistent VMDK Files

When using Independent Mode Disks, you must specify how the data will actually be written to your VMDK files. There are two access modes that you can set on a per VMDK file basis.

Persistent Mode - This is the default access mode for all newly-created VMDK files. VMDKs in this configuration behave exactly like a drive on a physical machine would. Once you make a change to the file system, its permanently written to the VMDK file. It's recommended that unless there is a specific need to use a different access mode, that persistent be used, as it provides the best overall disk performance.

Nonpersistent Mode - VMDK files configured in a nonpersistent mode discard all changes made to the file system since the point in time in which the access mode was changed to nonpersistent. When the virtual machine is powered off (not rebooted), any changes that were made to the VMDK are ignored. This is handy in kiosk situations where multiple people have access to a machine through some form of remote connectivity.

If someone were to delete key files, you can instantly return the system to its original state. Another situation in which nonpersistent mode may be convenient is for training classrooms. Several students may be given their own virtual machines as their lab to configure a specific system. They can install applications and reboot several times. At the end of the day, a simple power off of the VM will place the machines back to their original state.

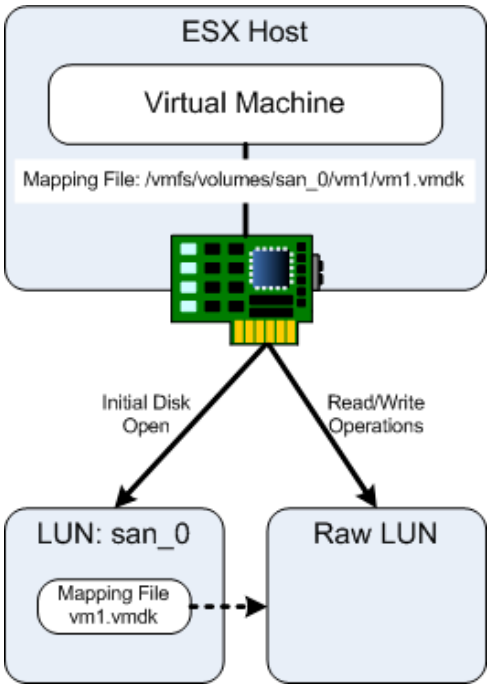
When using nonpersistent access mode, it's best to completely configure the server then change the disk access mode to nonpersistent. This ensures you always go back to the expected configuration on the VMDK file.

Raw Device Mappings (RDMs)

ESX 3 provides a way to map LUNs directly to a virtual machine without having to mess around with VMFS partitions and VMDK files. This is done by configuring a disk mapping file on an existing VMFS partition that each ESX host in a cluster has access to. When using disk mapping files, you are basically using a VMDK type file stored on a VMFS volume that instructs the VMkernel that the actual data is stored directly on a SAN LUN. The virtual machine that you want to leverage the Raw LUN will actually point to the mapping file as the location of its virtual hard drive.

The mapping file manages the metadata for the LUN that the virtual machine uses as its disk. The metadata is basically the locking information for the LUN and the physical location of the LUN. Notice here that we say LUN and not partition. Disk mapping files only support mapping directly to an entire LUN and not a partition that exists on a LUN.

Figure 5- 6: RDM



When using an RDM it is important to note that the mapping file is used to instruct the VMkernel where to write the data, not as a continuous proxy mechanism to the Raw LUN itself. When access to a disk is requested, the

VMkernel reads the mapping file, performs the necessary locking and reads the physical location of the Raw LUN. Subsequent I/O activity is then written directly to the Raw LUN itself without having to leverage the mapping file again (until the virtual machine is powered off and the disk can be unlocked and released).

The virtual machine itself will actually treat the Raw LUN as a regular physical disk and it will be partitioned directly with an NTFS or EXT file system, without having to use VMFS and VMDK files on the LUN itself. The advantage of using the mapping file is that regardless of which type of RDM you configure, you will still have the capability to perform VMotion, DRS, and HA activities of virtual machines with RDMs configured.

There are two different access modes for RDM disks inside VMware ESX, each with unique characteristics: Virtual and Physical.

Virtual RDM

When configuring a virtual mode RDM, the access to the RAW LUN is fully virtualized by the VMkernel. You have the exact same capabilities that you would if you were using a standard VMDK file such as the ability to add redo logs, and import and export the contents of the disk just like a normal VM disk file. Inside the virtual machine, you would see this disk just as you would a typical VMDK file as well. It would appear in Device Manager as a VMware SCSI disk device.

Virtual mode RDMs are often used for very large volumes that already exist and it doesn't make sense to migrate data over to a VMDK file. This is typically found on File, Database, or Messaging servers. By keeping the VMDK in virtual mode, you are afforded with the benefits of being able to add snapshots to your virtual machines and back them up by taking advantage of virtualization specific backup software.

Physical RDM

In physical mode, the virtualization of the SCSI device is limited. As a matter of fact, all SCSI commands from the virtual machine are sent to the physical device with the exception of the Report LUNs command. This mode is often

used when you are required to run SAN management agents within the virtual machine that require lower level access to the physical device. When using physical mode RDMs, the disks are seen within the operating system with their physical characteristics.

Another scenario in which physical mode RDMs are typically used are when physical to virtual or virtual to virtual cluster across hosts is required. Physical mode RDMs remove SCSI locking from the list of responsibilities of the VMkernel and passes it on to the guest operating system, which is a requirement for using MSCS in a virtual environment.

It is very important to note that since near nothing is virtualized, and SCSI locking is not managed by the VMkernel, it is not possible to use VMware snapshots when using a physical RDM. This is the trade off you have to give when you need lower level access to your physical disks.

When to Use RDMs

We have already highlighted several instances of when RDMs should be leveraged. One thing we want to make clear is that RDMs should NOT be used as a means to significantly increase performance of your virtual machine. Testing has shown that there is little to no performance gain by using an RDM over a VMDK file on a VMFS partition. In addition, there is no performance gain when choosing to use a virtual or physical mode RDM. The main reason you would want to use an RDM is for flexibility. If you need to move a LUN back to a physical host, whether it is for performance or other reasons, it is MUCH easier to do if you can simply rezone an existing LUN that has a standard file system on it.

Advantages of RDMs

- Allows MSCS clustering involving a virtual machine
- Eases data migrations of large data volumes that already exist in a virtualization project
- Provides the flexibility to move data volumes quickly from a virtual machine back to physical (Sorry VMware, it happens)
- Allows low level access to the disks inside a virtual machine (Typically used in conjunction with SAN snapshot software)

Disadvantages of RDMs

- Increases the amount of micro management of storage configurations inside the virtual infrastructure
- Must keep track of physical and virtual RDMs to know which capabilities are available to specific virtual machines.
- Cannot use VMware snapshots when using physical mode RDMs

Storage Layout and Design

The gritty details of ESX storage infrastructures come out when it is time to take all of the previous concepts and actually apply them into usable configurations. This is where VMware consultants and SAN engineers truly make their money.

Sizing

Determining the number and size of LUNs required in your environment is one of the most important things that you will do when dealing with your SAN configuration for ESX. The first thing you have to do when determining the number and size of LUNs you need is to estimate how much writable disk space you will present to the ESX servers in total (i.e., how much space are your VMDK files going to take up plus room for configuration files and growth). And the second thing is how you are going to break these up into usable LUNs.

Remember, with ESX 3 you are no longer simply storing static VMDK files on a virtual machine, there are other files you need to consider as well such as configuration and log files, but most importantly virtual machine swap files. By default, every virtual machine that gets created has a default memory reservation of Zero, meaning ESX will not set aside any dedicated memory resources. To compensate for this, a VSWP file is created that is equal to the amount of memory assigned to the virtual machine. If the system becomes over-allocated in regards to memory resources, it will begin to swap virtual machines with a lower number of shares to their swap file, effectively slowing down memory access for that virtual machine.

This, of course, gets added into the virtual machine configuration directory and must be counted towards the “used” space for your virtual machine. There are

several things we can do to help limit or eliminate this requirement, but it does require reconfiguration (and a power off and power on) of existing virtual machines. If the changes are applied to a virtual machine template, each newly deployed virtual machine will maintain the settings assuming the amount of assigned memory doesn't change.

Leave the default value and take the storage utilization hit. When in doubt, just leave the default values. They are called defaults for a reason.

Increase the memory reservation to be 50% of the virtual machines memory resources. This limits, but does not eliminate, the ESX host's capability to overallocate memory. At best, you can achieve an overallocation of 50% of your memory resources, which is still a VERY high number, and it doesn't use as much storage for your VSWP files. This is the overall recommended option, but does add slight management addition to your virtual machine deployment process.

Increase the memory reservation to be 100% of the virtual machines memory resource. This will eliminate any disk allocation to the VSWP file but it will not be possible to overallocate memory to your ESX hosts. The ability to overallocate resources is one of ESX stronger selling points so this option should be avoided.

Create a separate centralized VMFS volume on low cost storage and reconfigure each virtual machine to place their VSWP file on this alternate location. This option is a management nightmare and should just be avoided. You don't want to have to deal with planning more storage layout than you need to.

In addition to simply sizing the LUNs, additional thought should go into the workload characteristics of your virtual machines and how one virtual machine could potentially impact another. This will come into play when actually determining the layout of your virtual machines onto the allocated LUNs.

There are four schools of thought around how one would typically size their LUNs for virtual machine use:

Small Number of Larger Volumes

Using a small number of larger volumes is the easiest solution to manage and is probably the most common method found in a vast majority of ESX infrastructures. Using this method, LUNs that are capable of handing 12-15 virtual machines are created. A vast majority of the time it is found that the sweet spot for this configuration is within the range of 400-500 GB LUNs. If we use a typical example we see quite often of a 10 host ESX cluster of 4-way dual core systems, we can assume we can safely run 30 virtual machines per host. 300 VMs / 15 VMs per LUN = 20 LUNs. This is a very manageable number of LUNs within a virtual infrastructure. The one downside to using larger LUN sizes is often times you will find that you waste a bit more space than if you use a larger number of smaller LUNs.

Advantages of Small Number of Larger LUNs

- Easy configuration and management model
- Allows for large VMDK file configurations (100's of GBs)

Disadvantages of Small Number of Larger LUNs

- Typically find more wasted space on LUNs than with other models
- Increased bandwidth to each LUN limits load balancing capabilities

Large Number of Smaller Volumes

In this scenario a user may determine that they would like to have better control over their virtual machine VMDK configurations if they created smaller LUNs that contained the proper amount of space to store approximately 5 virtual machines. These LUNs would typically be sized somewhere between 100-200 GB. When using this methodology an end user needs to consider the sheer amount of LUNs that may be attached to a single ESX host while remembering that every host in the cluster needs to see every LUN that contains a virtual machine for the purposes of VMotion, DRS, and HA. Taking our common configuration of a 10 host cluster of 4-way dual core systems, we can assume we can run safely run about 30 virtual machines per host. 300 VMs / 5 VMs per LUN = 60 LUNs. All of a sudden the ESX administrators aren't simply doing virtualization management, but storage management as well.

Advantages of Large Number of Smaller LUNs

- Better control of utilized and unutilized space

- Lower bandwidth to each LUN which maximizes load balancing capabilities

Disadvantages of Small Number of Larger LUNs

- There are instances where large VMDK files will be required, which minimizes the overall effectiveness of the virtual machines
- Extra management by the ESX administrator to track and manage large amount of LUNs

The Hybrid Solution

Of course we are never satisfied with providing a solution that is either on one extreme or the other. A common practice that we find, and will often recommend, is the use of a tiered architecture in which several LUNs of various sizes are created and assigned to the virtual infrastructure. This solution provides a lot of flexibility by providing support for large VMDK files, while still maintaining flexibility around utilized/free space control and manageability. Without actually breaking out the Advantages/Disadvantages of this solution, assume it takes the best of both configurations and provides a solid solution that is being used at an increasing rate across many organizations.

Use RDMs Wherever Possible

There are some organizations that want to have the absolute maximum amount of flexibility at the expense of creating a management nightmare for their VMware administrators. In this scenario a few large LUNs are provided for storing mapping files and virtual machine snapshot files. The rest of the LUNs are customized on a per virtual machine basis and assigned as RDMs inside ESX. There is no wasted space, as each LUN is customized for the virtual machine it is assigned to and is configured with a native operating system partition. Regardless of which type of RDM is being used, VMotion, DRS and HA functionality will still be supported. For systems that do not require low level disk access from the virtual machine, it is also possible to add snapshots for backup and checkpoint purposes.

A MAJOR consideration when configuring this type of environment lies in the amount of LUNs that must be assigned to each host in the cluster. Let's take our traditional example with one small hitch that was previously irrelevant. We have our 10 host ESX cluster of 4-way dual core systems, and we can assume we can safely run 30 virtual machines per host. Our added hitch here is that

each virtual machine has an operating system LUN and a data LUN to maximize data throughput down multiple paths for the single VM. Doing some quick math we determine that 300 VMs X 2 LUNs per VM = 600 LUNs. If we go back to the very beginning of this chapter we remember that we can only assign 256 LUNs to any given ESX host. In this specific example we either need to drastically reduce the number of virtual machines per host (which is a bad idea to leave idle cycles in a consolidation environment), or reduce the size of our cluster, which increases management and limits our failover and HA options for those hosts and virtual machines.

One thing that drives us absolutely crazy is seeing people assign one LUN per virtual machine logical hard drive and configuring it with VMFS and placing a VMDK file that fills up the entire LUN. It simply makes no sense. If you are going to do this you eliminate your capability to create a snapshot and are much better off using an RDM.

Needless to say, using RDMs should be intermixed in with a hybrid disk sizing solution and only used if there is a specific need. Remember, RDMs do not provide any noticeable performance increase in a properly configured environment.

Hosts per LUN

VMware has a published maximum of 32 paths to any given volume. If you take a look at their recommended maximum of 32 hosts in a cluster, you had better be using single path LUNs, which is not recommend as it provides no redundancy for the most critical aspect of your virtual infrastructure. We assume every ESX host connecting to fiber based or iSCSI SANs will have at least two paths, making the absolute maximum number of hosts in a cluster 16. We even go a few hosts lower in our practical configurations from Chapter 4 where we recommend no more than 10-12 hosts in a single cluster. At most, this would provide a very safe value of 20-24 paths communicating to a single LUN at any given point in time.

LUNs per host

VMware has a hard limit of 255 LUNs that can be assigned to a host at any given time. Using our sizing suggestions from above you should be hard

pressed to have to manage any more than 40-45 LUNs on any given host of a cluster in a worst case scenario. This is still a fairly manageable number and provides some good opportunities for properly balancing LUNs down multiple paths to the storage infrastructure. We find that having 20-25 LUNs assigned to a single host is actually the norm.

Template Locations

Virtual machine template storage is a unique scenario in which it is worth building and configuring a lightweight NFS server simply for that purpose. Every Linux distribution under the sun has NFS included, and Windows has even started distributing an NFS server in Windows 2003 R2 or other versions of Windows with their Windows Services for UNIX download. With the amount of usage this particular volume will receive, it does not need to be a costly or high performance solution, as long as it stays running so virtual machines can be properly deployed.

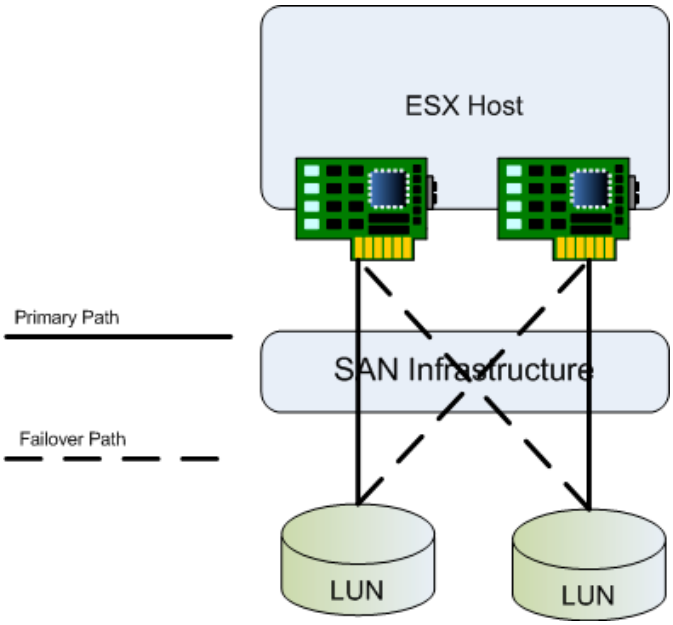
Load Balancing and Failover

The various storage configurations within ESX each require unique failover considerations. Carefully following the guidelines laid out in this section will ensure the critical backend storage infrastructure is as redundant as possible for your virtual infrastructure.

Fiber SAN and iSCSI Hardware Initiator

For redundancy purposes it is recommended that at least two fiber HBAs be installed in an ESX host. While the VMkernel itself does not provide any load balancing down multiple storage paths, through creative and proper planning we can achieve manual load balancing on a per LUN basis. Fortunately, the VMkernel does provide efficient failover capability if the same LUN is seen down multiple paths.

Figure 5- 7: Multipathing



As you can see in the figure we have a simple configuration consisting of a single host with two HBAs communicating with two LUNs. By default ESX has configured a different HBA as the primary path for each LUN. This provides our load balancing method for our storage infrastructure. While this is not the most efficient load balancing in the world, it is the best we can currently get in our virtual infrastructure. If either primary path were to fail anywhere in the SAN infrastructure, the failover path would kick in within about 45-60 seconds and take over storage communication for the required LUN.

VMware provides two different internal methods to provide load balancing down multiple paths of the storage Infrastructure: Fixed/Preferred Path and Most Recently Used (MRU).

The MRU option is the default configuration that is used with Active/Passive storage devices, such as small arrays that have dual controllers in an active passive configuration. These are devices that maintain a single active path to the exposed disk and failover in the event of a component failure. The Fixed/Preferred Path option is the default for Storage that is configured for Active/Active accessibility. Generally higher-end SANs will support an active/active configuration. You will need to verify the configuration your SAN

supports by reviewing the VMware documentation or working with your SAN vendor on the proper setting.

In an MRU configuration, the HBA being used to access a specific LUN will be used until it is unavailable. In the event of a failure, either in the card, the connection, or the storage processor level, ESX will automatically failover to the other HBA and begin using it for connectivity to the LUN. If the first card/path comes back online, the path currently being used will not change. As the name states, ESX will continue to use the path that it has failed over to since it is the 'Most Recently Used'. The storage array itself can often be configured to specify which of the available paths between the ESX host and the SAN controller is the "preferred path", but this functionality is not available in ESX when using the MRU option.

Advantages of using MRU

- Automatically configured to an available path
- Very little configuration required

Disadvantages of using MRU

- Possibility that all VMs will run over a single HBA
- Since the first HBA in the system is scanned first, this path is likely to be found and used first by the system for all LUNs

Contrast this to the Fixed/Preferred Path configuration option. In this configuration, the Preferred Path is used whenever it is available. So if a connection is lost, and the ESX Server fails over to another HBA/Path, it will only use that path until the preferred path becomes available again. This is used in Active/Active storage arrays in which any of the available paths can be used for storage communication at any point in time.

In an Active/Active SAN it is possible to assign alternate preferred paths on a per LUN basis. It is worth the extra time and configuration it takes to continuously analyze the balance of communication down the multiple paths of an ESX host and adjust it when necessary by changing the preferred path of key LUNs to equalize the path utilization. In Active/Active configurations this rebalancing is done through the Virtual Infrastructure Client and does not impact the availability or performance of the virtual infrastructure. Naturally, it should be documented and processed through a change control just in case you fat finger

something and end up disabling a path and crippling your disk I/O by jamming it all down a single path.

Advantages of using Fixed Multipathing

- Allows you to 'manually' balance your LUNs between HBAs.
- System will automatically return to your original configuration when the path is returned to an operational state.

Disadvantages of using Fixed Multipathing

- Requires a little manual setup initially to ensure LUNs are split evenly between HBAs.

So what happens during a failure?

When an active path to a SAN disk is lost, the I/O from the virtual machines to their VMDK files will freeze for approximately 30-45 seconds. This is the approximate amount of time it will take for the SAN driver to determine that the link is down and initiate failover. During this time, the virtual machines using the SAN may seem to freeze, and any operations on the /vmfs directory may appear to hang. Once the failover occurs, I/O requests that have queued up will then be processed and the virtual machines will begin to function normally.

If all connections to the storage device are not working (assume a disastrous loss or a single path configuration), then the VM's will begin to encounter I/O errors on their virtual disks.

iSCSI Software Initiator and NFS

Load balancing iSCSI software initiated LUNs and NFS volumes is completely controlled by the standard network high availability capabilities of the VMkernel. The process in which load balancing and failover are performed for these storage types are entirely controlled by the configuration of the virtual switch responsible for managing the communication to the storage systems. You will have an immediate understanding of this process upon completion of the next chapter, in which we discuss the networking aspects of the virtual infrastructure.

Note on Windows Guests

When using Windows Server OS's in your VMs, you may want to adjust the standard disk time out for disk access. During the failover of an HBA, the default timeout in a Windows guest may cause issues within the guest OS that is used to having extremely responsive disks. For the Windows 2000 and Windows Server 2003 guest operating systems, you should increase the disk timeout value in the registry so that Windows will not be extensively disrupted during failover. The registry value can be found at the following location:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Disk

The entry you wish to change, or add if it does not exist, is the TimeOutValue. Set the data to x03c hex or 60 decimal. This will configure the Windows Operating system to wait at least 60 seconds for disk operations to finish before reporting errors, which will likely consist of a wonderful blue screen.

Conclusion

As you can see, there are a lot of critical considerations and configurations that are required for a proper storage infrastructure for your virtual environment. With the introduction of network based storage platforms the entire process is complicated by the fact that we are not simply dealing with a centralized storage infrastructure, but we need to consider the importance and impact of the network infrastructure as well. We did not dig too deeply into these network configurations, but not to worry, our next chapter does plenty of that for us.