VMware® Infrastructure 3

Advanced Technical Design Guide

~and~

Advanced Operations Guide

Two books in one!

Ron Oglesby
Scott Herold
Mike Laverick

Chapter 4 - VirtualCenter and Cluster Design

In this chapter, we will look at ESX design options from a managed cluster perspective in addition to VirtualCenter (as a service) and its alternatives for implementation. Also, we will wander around the VMware licensing model and some of its design considerations.

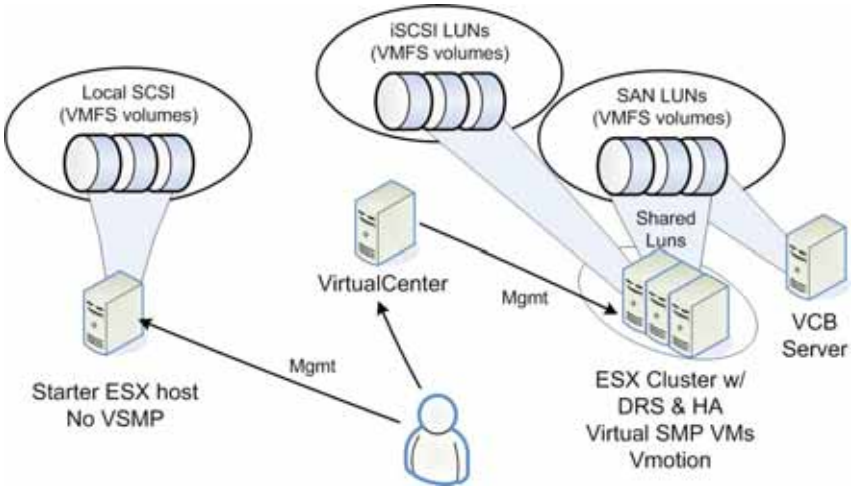
Licensing

While no one really likes to talk licensing, it is a must. VMware (like any other software vendor) wants to get paid for their intellectual property, and thus we need to license the servers. Let's look at what components need to be licensed and how they are licensed, then we'll review the licensing architecture used in VI3.

Licensed Components

One thing about being in IT (that no one ever mentioned when I got into it), is that licensing, not to mention the legalities of it, is one of the biggest pain in the butt issues I deal with all the time. Every time a vendor comes out with a feature, you get a new price and a new license model. Or, you have to ask how things are licensed (like per server vs. per processor, and does server mean a virtual or physical instance, etc…?). While all the license models in the world are a little out of scope here, and quite frankly, impossible to write about since MS and other vendors change their stance about once a quarter, we will try to walk you through an ESX environment's licensed components and what they mean to you. Below we have an ESX server environment using almost every possible licensed feature for ESX.

Figure 4- 1: VirtualCenter Licensing



The setup in the previous image shows two different ESX environments. The ESX host on the left of the image is a standalone host, showing a basic "Starter Edition" configuration. You get a basic ESX host, the ability to create VM's and use local VMFS volumes and that is about it. These servers can have up to 4 processors (you pay per socket) and 8 GBs of RAM. There is no limit on the number of VM's, but you are limited in reality by the amount of RAM to about 16 VM's. In addition management of these servers is not centralized. They are each managed individually, with no centralized VirtualCenter Server.

Moving left to right in the image we then see a VirtualCenter server managing an environment of multiple ESX servers. When moving up to Standard Edition for the ESX host you get additional storage options like iSCSI and Fiber Channel based SAN. In addition, your VM's can now use Virtual SMP (dual and quad processor), and the limitations on the number of processors and amount of memory for the host has been removed. This is a fully functional host, but not a fully functional environment, yet.

You'll then notice that the "Standard" servers are configured in a cluster using advanced features of VirtualCenter like Distributed Resource Scheduling (DRS), High Availability Services (HA), and VMotion. Each of these is an "add-in" license to enable a specific functionality. These licenses are in addition to the basic VirtualCenter agent license, which allows the Standard Edition ESX server to be managed by a VirtualCenter agent. Confused yet? Don't be, while you can buy these piece meal, often times you will either purchase a Starter, a Standard, or VI 3 Enterprise (what used to be call an Infrastructure Node license) that

includes all of the optional components, even if you don't need them all. The reason to get the infrastructure node is simple, price. If you need one or two of the add-ins, it is often about the same cost as getting the node license and much simpler from a quoting and purchasing/upgrading perspective.

Finally, on the right hand side you will see a VMware Consolidated Backup (VCB). This is another feature (that you can get with the VI 3 Enterprise license) that is licensed Per ESX server and allows for enhanced backups of VM's.

So which one do you need? 90% of you who are reading this book will need the infrastructure node licenses. But for a short who's who in the zoo, we break it down like this:

Starter Edition: This is used by one or two server shops that do not need central management, work load balancing of VM's, or high availability. Generally used when testing the waters.

Standard Edition: This is used by almost everyone and managed in almost every environment by a VirtualCenter Server. Used by those that may or may not need the DRS or HA services but do need a central mgmt console.

VI 3 Enterprise (Formerly Infrastructure node): This is used by a majority of environments with anything more than a couple of servers. These license packs allow you to take advantage of automatic VM recovery (HA services), give you the VirtualCenter mgmt license, along with DRS and VCB.

How licensed components are licensed…

ESX Host licenses are simple; they are licensed by processor socket (NOT CORE). As of this writing the ESX two processor licenses is good for a dual processor quad core server, and if you wish to support a 4 processor server (regardless of cores per socket) then you need to purchase two dual processor licenses. The amount of processor throughput you are getting for your dollar in multi-core systems is great. I expect at some point for this to change, but right now you should get it while the getting is good…

For other components it's a little trickier. The VirtualCenter Agent component along with VMotion, DRS, and HA are per host features, but generally work out to a per processor cost. What I mean by this is that an Infrastructure node license for a dual processor system is maybe $5,000 US. But a quad processor system with the same feature is $10,000 US. So while the HA, or DRS features are pretty much processor agnostic and really are VM or Host-based, their pricing is very dependent on the number of processors (and therefore the number of VM's) you have. VCB, on the other hand is licensed by host (unless you roll it into the infrastructure node). In either case, it's still a per host license; only, it's included with the inf. node pack so we get back to the paying by the proc concept.

VirtualCenter is essentially licensed by the number of agents (read ESX servers) you have. These again tie back to the number of processors as a Quad is a different cost than a dual proc.

The basics of licensing with ESX is that everything is moving to a per processor model. So the sweet point now is that multi-core processors are offering more processing time per dollar since VMware is still charging for licenses by the number of sockets and not cores. Best bet is for you to determine the number of processors you are going to use, then get quotes based on that. Too often people get mixed up in the "I have 6 dual, dual cores, or 4 quad proc dual cores, and confuse themselves and the sales guys. Just count up proc sockets, and tell them what features you want.

Choosing a Licensing Model

Licensing in ESX 3.0 is essentially a file-based license. In the ESX 2.x era, administrators would key in a long product license code into the web interface for each ESX server they managed. Then, if the server were managed by VirtualCenter, and you wanted to use features like VMotion, you keyed in licenses (basically from a text file) to enable VC management and advanced features.

The file-based license architecture used in ESX 3.0 changed all of this. ESX licensing now has two modes: 'Host based' licenses (a file that resides on the ESX server) and Server-based licenses (a file that is hosted on a separate Windows server running the VMware license services). The concept is this, you (as a VMware customer that has purchased licenses) can go to their website and

generate license files. The format of these files is the basic format used by FlexNet (formerly FlexLM) licensing. These license files can contain all of your licensed functionality (VMotion, DRS, how many processors, etc…) and determine where the license files are stored, on the host or on a centralized Windows Server running the FlexNet licensing services. It should be noted here, that when you generate the license file you must choose between the host based and server based, they can not be interchanged.

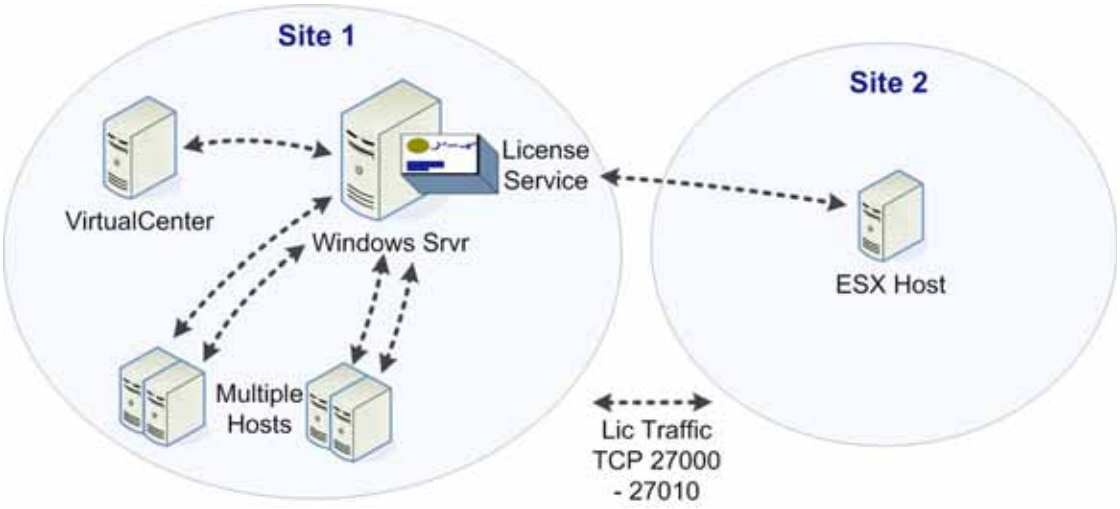
Essentially you have two options when it comes to how to maintain / host your licenses. You have the ability to host them on a central server for all of your hosts, or you can have an individual license file for each and every host in your environment.

VMware (in their documentation) allows for a 3rd model where you would store the VirtualCenter license file on the license server then also have each individual ESX server host its own license. We see no need for this except in cases of upgrades (like during the transition from a single server environment with no VirtualCenter) or some weird political issue within your organization that X business unit owns a server license, but corporate IT owns the VMotion and Virtual Center license etc… If the latter is the case then cut through the politics, if it is the former, then you are moving into a centralized model anyway.

Server Based Licenses (Centralized)

With that out of the way, let's talk about centralization. The centralized model requires a Windows server running the VMware licensing services. These services are really the FlexNet/FlexLM services, and I would suggest you install this first (prior to VirtualCenter) as it is pretty much required anyway. The server-based (Centralized) model is the most common in use in ESX environments. It allows for you to add your licenses for base ESX Functionality (licensed by processor socket) and allows for licensing for additional functionality like VMotion, DRS, HA services, Virtual SMP etc.

Figure 4- 2: Centralized Licensing



The nice thing about this model is that you can generate your license file for your entire environment (let's say 100 CPU's as an example); then, dole it out as your ESX servers come online regardless of the number of CPU's they have. Essentially you can have multiple hardware form factors (duals, quads, eight-ways, etc.) all pulling from the same server, and as long as you have available licenses, they will be good to go. Additionally, this centralized model makes licensing much easier to manage. You have one place to add and remove licenses and can restrict access to that server/application as needed.

In a multiple site model, you can opt to host the licenses at the corporate site and have the remotely located ESX servers connect to that server. If the WAN link between the remote site and central office is down it won't kill the remote ESX hosts. The VM's already running will continue to do so, but you cannot make configuration changes (more about that later in the chapter). If you feel that this is too much of a risk, you can put a license server at the remote site. Often license services are located on the VirtualCenter server, which makes sense, since that is the central configuration point and the tools you would most likely use to make the changes you want. VirtualCenter server placement is much more important than the license server placement, and you should finish reading the next few sections before jumping into a decision about license server placement.

The draw back to this model is that it is centralized and doesn't really allow you to earmark licenses for specific servers or business units. If in your environment

your business units own the licenses for ESX, then they may have reservations about centralizing them. Another drawback to this model is that it is a single point of failure, not a massive failure, mind you, but not easily made redundant. See, the license server parameters for each host is set on a host by host basis. Meaning, at time of install, you point your server to a license server, give it the port number and it knows where to get licenses from. The issue with this is that if the license server (or link to it) is down (and FlexLM isn't really clusterable) the ESX servers can't get their licenses. Now this is not the end of the world since ESX servers will continue to operate and you are not even in violation of the license for 14 days! In addition the already running VM's will continue to run, but no new hosts can be added to the clusters. We'll talk about some licensing options for high availability environments in a moment, for now just understand the limitations.

Advantages of the Server-based (Centralized) License Model

- Single management point for all licenses
- The only way to go (because of previous advantage) in large ESX environments

Disadvantages of the Server-based (Centralized) License Model

- Requires an additional Windows Server to host the services (though license services are often co-located with VirtualCenter services)
- A single server outage or down WAN link can be a single point of failure (though it does not affect already running VM's or Hosts)

If licensing model seems familiar to you it is probably because Citrix uses the same platform, only in their use of this model they tie their license file to the host name of the server on which the license service resides. VMware does not do this as it would be a royal pain in the butt when using Host-based licensing. Hopefully, they will never go to that model.

Host Based Licenses (Per Server)

Host based licenses are often used in small one or two server environments. They have the advantage of not requiring a separate server for hosting the license file but require that you install (and manage) the licenses on a per host basis. While this works (and is used) in many small environments, it is not very scalable and can become a management nightmare when licenses need to be

upgraded or changed for some reason. In addition if you are going to run VirtualCenter to manage these one or two hosts, then you need a license service running on a Windows server for the VirtualCenter server. Since most environments beyond one or two servers use VirtualCenter (and therefore have the license services running on windows) it is easiest just to centralize the licenses on that server.

Advantages of the Host-based License Model

- Does not require a Windows Server for licensing (unless you also use VirtualCenter)
- VM's and hosts can still be started and restarted if a license server is down

Disadvantages of the Host-based License Model

- Decentralized model requiring host licenses be managed individually

License Server High Availability and Common Implementations

In most cases (in every case I have been a part of) the license services reside on the same Windows server that is hosting the VirtualCenter services. The license service itself is a lightweight, almost no resources used, type of service, and could easily reside on any server in the environment. It just makes a lot of sense to match it up with the VirtualCenter server. The other thing about a down license server is that running VM's and hosts are not affected for 14 days. Basically, you have 2 weeks to get the license server back up and running (from a legality perspective). Of course you can't add any new hosts the environment, but VM's continue to operate. Personally I think VMware has went out of their way to make this easy to live with, but maybe that's just me.

At this point it is important to understand that neither VirtualCenter nor the VMware License Services are supported in a clustered configuration. The VirtualCenter database may be hosted on a cluster, but the services themselves are not cluster aware. It is also important to note that VirtualCenter uses a 'Heartbeat' at 5 minute intervals to determine if the license service is still available and if there have been any changes to the licenses (remember these are just files managed by other interfaces). If the licenses have changed (or the license server goes away, same applies to the VirtualCenter server) it basically notes that the

licenses affected are now in an "Unlicensed Use" state. The idea here is that if the license server returns to service (or the single license) things will go back to normal, but it may take 5 minutes because of the polling interval.

Now, let's get back to the common implementations part of this.

- VMware recommends that you: Install the License Service on the VirtualCenter server and;
- States that you can make this server a Virtual Machine and place that Virtual Machine in an HA cluster to provide redundancy.

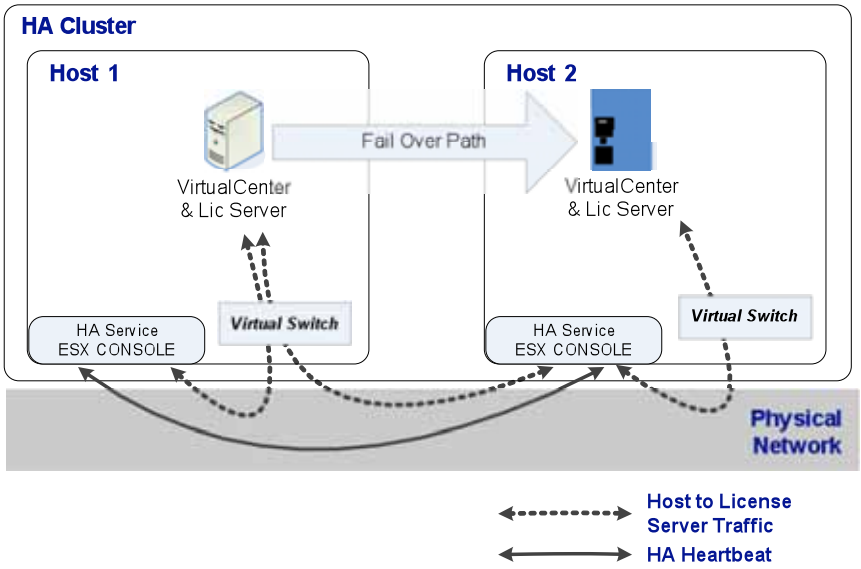
We need to look at each of these individually and explore a couple of other options. The first recommendation makes complete sense in that if you already have a Windows server hosting VirtualCenter why not use it for the Licensing services. The second needs to be explored a little more.

License Service on a VM

The second suggestion is a little more interesting, and we should explore a bit without getting too much into HA and its internals just yet. Reviewing the next image we will notice a few interesting attributes:

- The HA heartbeat traffic for the host is done via the console interface over the physical network.
- TCP traffic for HA uses ports 2050 thru 5000 and 8042 thru 8045
- HA services run on the host itself and are NOT part of VirtualCenter; they are just configured using VirtualCenter.
- Using the image below ESX Hosts will look for licenses via their console interface, out over the physical network, back to the VirtualSwitch interface, then to a Virtual Switch, and ultimately the License Server (When the license server is a VM).
- If Host 1 fails (let's assume a server shutdown of some type) the Virtual Machine restarts on Host 2, generally within a few minutes.
- The remaining hosts contact the license server, as before, once network connectivity is re-established by the VM

Figure 4- 3: License Service on a VM



This seems to work fairly well but ignores some basic limitations of HA and hard shutdown Windows servers. If the host completely fails the VM will restart on another host (good for us). And if this VM hosts just the license services for VMware there are really going to be no problems since HA kicked in and the license service is not heavily transactional.

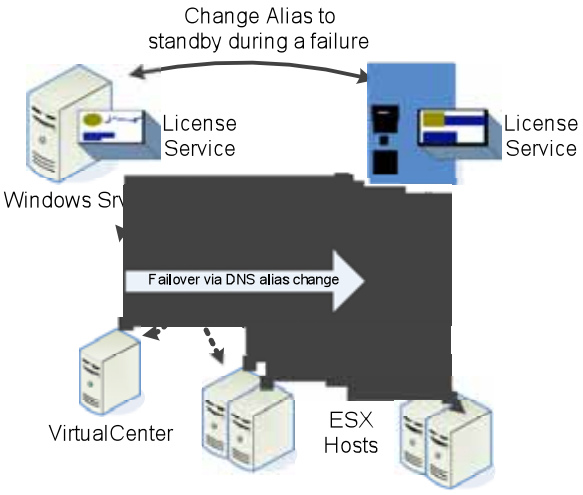
But if Host 1 does not completely shutdown, let's assume that you only loose connectivity to storage on Host 1, then the VM will eventually blue screen (due to loss of connectivity to its disk), but the VM and host are still running. If this happens HA failover never occurs, yet you have lost your licensing service.

HA services (the good and the bad) are gone over in more detail at the end of the chapter, but for now it's important to understand that it protects against a hardware failure on the host and RESTARTS the VM as if it had had its power cord pulled. It does not VMotion VM's off as that assumes the host is still up and running, which if HA has kicked in, it's not. Also, it is important to note that you need a host failure, and not something like a loss of storage connectivity, to make HA activate.

License Service on a Physical Machine

Another option is to place your License server on a physical machine. In this scenario you are doing this for one of two reasons. The first is that your VC server is physical, and therefore you have decided to place your license services on the same physical box. Or, you have an existing FlexNet server (for maybe a Citrix environment) and are going to go ahead and use that for VMware, too.

Figure 4- 4: License Service on a physical machine



The trick with this is that you still need some type of redundancy. In most cases, we will use DNS names (read alias) for the license service (like vmwarelicense.company.com). In a physical server model you could have a standby server, with the license servers loaded and none of the hosts pointed at it. Use the DNS name (as the figure is shown below) during configuration, and in the event of a server failure, simply install the licenses on the license server and change the DNS alias. This of course would require somewhat short TTL's on the DNS side, but it would ensure that within a few minutes your licenses are back up and available.

Using Software-Based Replication for Redundancy

A final option for your redundant license server configurations (whether virtual or physical), is to use a software replication package like Co-Standby Server or NSI's Doubletake. These types of packages essentially do software base replica-

tion between two servers with a named primary and standby. In the event of one server going down, the other server is notified (via a heartbeat) and takes over the services of the primary. In some cases, these packages even take over the original host name and IP address.

This type of solution will work either physical or virtual and can even allow for you to run a physical server and a virtual standby. The drawback to using these is cost. While the level of redundancy is great, you have to shell out some dollars for these packages, and the better they are the more they hit your pocket book.

VirtualCenter Server and Services

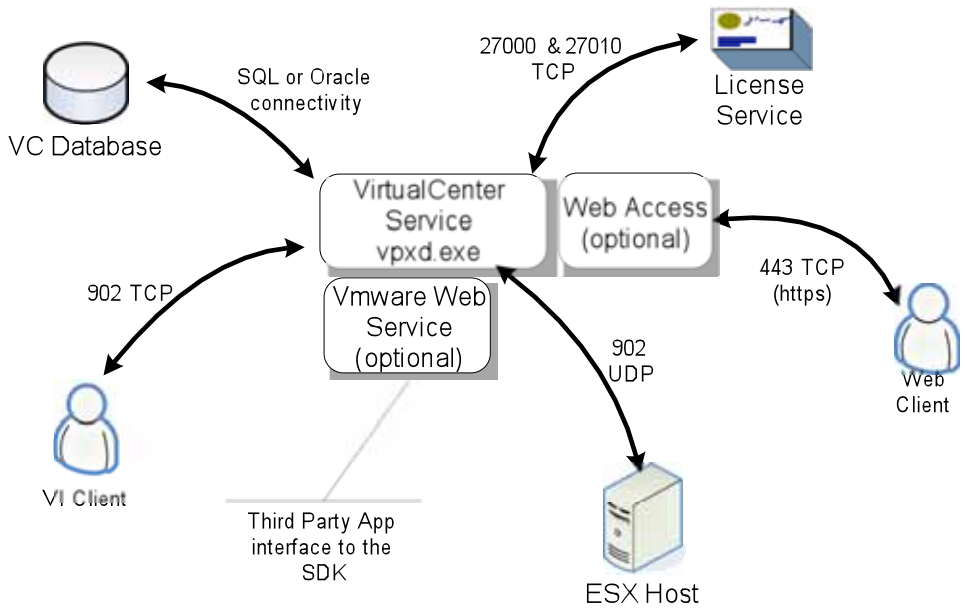
While VirtualCenter is sometimes talked about as an add-on product for ESX, it is really central to the entire system, and needed for almost all of the advanced functionality. VirtualCenter will (in most environments) be the daily tool used to monitor, manage, and configure their ESX environment. It provides several crucial functions in each ESX environment including:

- Configuration of the ESX servers
- Resource management for the hosts and virtual machines
- Console access for the Virtual machines
- Performance reporting for VM's and Hosts
- Inventory views into Hosts and virtual machine
- Ability to logically group these objects and provide role based security
- Alerting on performance thresholds for individual VM's or Hosts.
- Interfacing for third party products in the environment
- Configuration of ESX Clusters for both HA and DRS
- Configuration of resource pools to manage VM resource utilization

So regardless of what else goes on in your environment, you will more than likely want a VirtualCenter server, and quite frankly, we see it as a must beyond

one or two servers. So, to better understand what VirtualCenter does, let's look at its components, then dive into how to design the system.

Figure 4- 5: VirtualCenter Components



The first two components to note are the "optional" ones: the VMware Web Service and the VirtualCenter Web Access. These two should not intermingle as they are really two separate entities. The VMware Web Service is the SDK and interface for many third-party tools. If you use something like Provision Network's VDI broker or certain products from HP, these tools ask for the VirtualCenter server name and will want to really connect to the SDK. The Web Access component is a web page/s for Virtual machine management. It has limited functionality but will allow you to do some things. So while these are "optional" you might as well install them and save yourself the hassle later when some new fancy tool wants to use them.

As you can see, the rest of the components circle around the vpxd or VirtualCenter service. This service is what your VirtualCenter client (or Virtual Infrastructure Client) will connect to when managing the environment. The VPXD service also talks to a database (where it stores its information) and the License Services (to make sure you are compliant on the licensing side).

Often times in small environments all of these components (short of the ESX host) are located on the same physical machine. Administrators will install MS SQL on their Windows server and then install the VirtualCenter package including the optional services, the License Services, and the VI client. It all works and is even fairly scalable, but let's look at each of these components individually.

VirtualCenter Service (VPDX) hosting

The VC server itself needs little in the way of resources. Thinking about the diagram, and about some of the rules we are about to go over; you can tell that there will not be a lot of connectivity to the VC server. It will communicate with hosts in the environment (maybe even a lot of them), possibly have several users connected with the VI client, and maybe even a third party tool using the web service. But even with this number of connections, we may be talking only on the high side of 100 or so… not a lot of load. In addition, VMware recommends that a VirtualCenter 2 server not have more than 2000 VM's or 100 ESX hosts to maintain its performance. While most environments won't start off with anywhere near this many, it is easy to look ahead (in this era of blades) and see an environment with 100 Hosts, at about 8-12 VM's per blade and pushing that limit. Our general recommendation for sizing is as follows:

- 1 processor for up to 25 hosts, add a second processor if you plan on scaling above 25
- 512 MB base memory, then about 1 MB of memory per every VM managed and 2 MB of memory per managed host.

Following those recommendations, let's look at two sample environments and come up with their VirtualCenter server configs (remember these exclude the SQL or Oracle DB if they are running locally). If I had a 10 Host environment with 200 Virtual Machines, I would need 512+ (10*2) + (200*1) = 732 or 768 (rounded up to next 256 increment). But if the environment needed to scale to 60 hosts and 1200 VM's we would need 512 + (60*2) + (1200*1) = 1832 or 2048 rounded up to the next increment.

Granted, the equations are simplified, but as you can tell a dual processor, 2GB system will pretty much handle the load for one of their bigger supported environments. You may also notice we start out at 512. That allows us to assume there is overhead from the OS, Monitoring agents, anti-virus, etc. But in any

case, you can see this doesn't have to be a powerful system. If you want to be a little more conservative, take the MB per VM and per host and increase them by 50% or 1.5MB per VM and 3 MB per host.

Physical vs. Virtual Machine VirtualCenter

Running the VirtualCenter server as a Virtual Machine is not an issue. A number of organizations do it and do it successfully. Though, in some shops, the question of running your management service, which allows you to manage and view and possibly troubleshoot your virtual infrastructure, becomes a contentious one; so let's look at some of the benefits and drawbacks.

Advantages of using a VM to host VirtualCenter

- Reduced number of physical servers
- Hardware failures covered by HA services
- Load can be shifted and the VM easily "upgraded"
- VM can be snapshot before upgrades and patches
- No need for a standby server, HA will often recover faster than you can configure and start the standby server

Disadvantages of using a VM to host VirtualCenter

- HA Services do not cover all failures (such as a storage failure)
- Internal resistance due to managing a VM environment with a VM

Running it as a VM is not a bad idea as long as you do not host the VC database on that same server. Here we are explicitly talking about the VC server and not it's DB. The databases used (SQL and Oracle) are transactional databases that do not like to have their availability based on a power off and power on. Loss of transactions can occur and in really bad cases, database corruption. We suggest that if running this as a VM, separate the Database from the VirtualCenter Server. Then the DB can be hosted on a cluster (VM or Physical) and be guaranteed higher availability easily.

If you elect to run the VirtualCenter as a Physical server then you have a number of options. Obviously, you lose some of the HA features when not running it as a VM. So, you need to plan accordingly. One of your options is to use a

physical machine and a standby. The standby can even have VirtualCenter installed; then during a failure start it up, point it to the Database, and off you go. You should (on the standby) change the VirtualCenter unique ID (located in the VI Client –Administration – VirtualCenter Server Configuration –Runtime Settings) to match that of the production server, but it is a quickly recoverable solution.

You also have the option of using something like an NSI Doubletake to replicate the VC server to a standby. Again, this can be physical to physical or physical to Virtual. Thinking about it some, it can even be virtual to virtual if you so choose.

While VMware doesn't officially state that VC is cluster aware* they do have some articles on it (from the VC 1.x days), and I am sure they are planning on making VC a clusterable application though it's not available yet. In some cases, Users in the forums have posted links to cluster VC steps…

Just before going to print VMware posted an article on clustering the VirtualCenter service for 2.0. We will cover this in detail the next edition, once we have had time to test it. The article can be found at: http://www.vmware.com/pdf/VC_MSCS.pdf

VirtualCenter Database Sizing

Whenever designing the VC solution and talking to the DBA's, they always want to know how big this database is going to get. From a utilization standpoint (processor and memory) this DB has very few clients (Uh, your VC server) and probably will never have more than 10 connections to the DB from even your VC server. So, with low utilization it always comes down to a question of space. At the time of this writing, VMware has a spreadsheet they float around to help estimate size of the DB. But, it is important to note that its sizing does change between versions as they change counters and objects. So, sizing described here is based on VirtualCenter 2.0.1 Patch 2 and later.

The biggest factors impacting the size of your VC database are the number of Virtual Machines and the logging level set in VirtualCenter. The default logging level in VC is Level 1 of 4 possible levels. Each time you crank up to the next

level you increase the number of *counters/metrics* being collected on each sample, thus increasing the amount of data. In addition, the default smallest collection interval is 5 minutes. If you change this and increase the frequency from the defaults, these numbers can also be skewed.

Anyway, let's look at some sizing numbers below based on the most important items with regards to the database sizes at different logging levels for VirtualCenter:

	Objects by logging level (in MB)			
	VM	Host	Cluster	Res Pools
Logging Level 1	3.52	3.01	3.52	0
Logging Level 2	10.05	10.05	9.55	7.03
Logging Level 3	26.65	49.75	9.55	7.03
Logging Level 4	37.19	71.36	21.6	196.3

These numbers are based on common configurations for hosts, and VM's, meaning the Host numbers assume a 4 processor host (or dual proc dual core) with several network interfaces and numerous disk devices etc. The VM numbers assume the average VM is a single processor with 1 disk and 1 network interface. The Cluster and Resource Pools are pretty steady numbers, but if your VM's all have 2 disks instead of 1, or if your Hosts have 8 processors instead of 4, you can fudge these numbers up by 15% for logging levels 3 and 4. At levels 1 and 2, it really doesn't matter much.

In addition to the raw database size, which you can get by multiplying your expected number of each object times the MB at the used logging level, you should really allow the DB size *2 to allow for the temp database that will also use disk space. Logging level selection is looked at in the next section, but for now let's runs through some examples.

Remember VMware does have a spreadsheet calculator on line, though it is sometimes hard to find, at the time of this writing you could download it here:

www.vmware.com/support/vi3/doc/vc_db_calculator.xls

Sample Environment 1: 10 Hosts, 150 VM's, 1 Cluster, and 2 Resource pools, level 1

Here, we do some simple arithmetic to determine the potential DB size after a year of data. Year-old data is purged so the DB will remain about the same size over time.

	# of Objects	MB/Per	Total @ 1 yr
VM's	150	3.52	528
Hosts	10	3.01	30.1
Clusters	1	3.52	3.52
Res Pools	2	0	0
		total:	561.62 MB

As you can see, we are running about 15 VM's per host and wind up with about a 560 MB database. Of that 528MB is from the Virtual Machines. Now, let's change the number of hosts (by using smaller hosts but host the same number of Virtual Machines.

Sample Environment 2: 20 Hosts, 150 VM's, 2 Clusters, and 4 Resource pools, level 1

Here we have basically doubled the size of the environment from a host perspective, creating two clusters, compared to one, but running the same number of virtual machines.

	# of Objects	MB/Per	Total @ 1 yr
VM's	150	3.52	528
Hosts	20	3.01	60.2
Clusters	2	3.52	7.04
Res Pools	4	0	0
		total:	595.24 MB

As you can see, the additional hosts are really just 10 or 15 new logical objects being monitored and barely impact the size of the DB. The greatest number of Objects (with unique metrics being monitored for each) is the VM's; therefore, they are most important in determining the size of the DB.

Sample Environment 3: 10 Hosts, 150 VM's, 1 Cluster, and 2 Resource pools, level 4

In this sample we take environment 1, which previously was at logging level 1(estimated at 560MB), and crank the logging level up to level 4. Now, most environments are never going to run at level 4 but this shows how the logging level increases the DB size dramatically:

	# of Objects	MB/Per	Total @ 1 yr
VM's	150	37.19	5578.5
Hosts	10	71.36	713.6

Clusters	1	21.6	21.6
Res Pools	2	196.3	392.6
		total:	6706.3 MB

Notice that the environment that did have a 560 MB db is now more than 10 times that size at about 6.7GB. Of course you may be higher or lower than this (use a 10-15% plus or minus) but the trick is to realize that the logging level has a major impact on DB size by increasing the number of metrics being monitored for each item.

VirtualCenter Stat Collection and Logging Levels

Number of processors per host and VM and number of network interfaces and disk devices for hosts or VM's, all have an impact on the size of the DB, but their impact is fairly minimal when compared to the logging level and sheer number of metrics that are added as you increase the logging level. Because of this, let's look at the different logging levels, what they provide you, and why you would use them.

Beware: Changing your logging level in VC removes all of your previous VC logged data. I believe they change the fields/tables when they change the logging level; so, when you change this in VirtualCenter it removes all of your previous data.

Logging Level 1

This level provides the basic metrics for VM's and hosts, including CPU, Disk, Memory and network usage. Uptime metrics are counted along with DRS metrics. Statistical information for individual devices is not collected in this logging level.

Logging Level 2

This level grabs all of the metrics for the core four (CPU, disk, memory, and network) and device statistics that were not included in the level 1. As an exam-

ple, an average quad processor, ESX server will have 6 metrics collected at level 1 during a sample interval, while level 2 collects a total of about 20 (+/- a few based on the number of devices in the host).

This level is used most often in environments that do capacity planning and charge back on VM's. It allows you a pretty granular look at the information about the core four without grabbing level 3 counters and which is a big jump in the amount of metrics monitored.

Logging Level 3

This level collects all metrics for all counter groups. The increase in from level 2 (20 metrics each sample interval) is almost 500%. The total metrics captured here is 131. This level is often used for troubleshooting or environments in ASP/Hosting models.

Logging Level 4

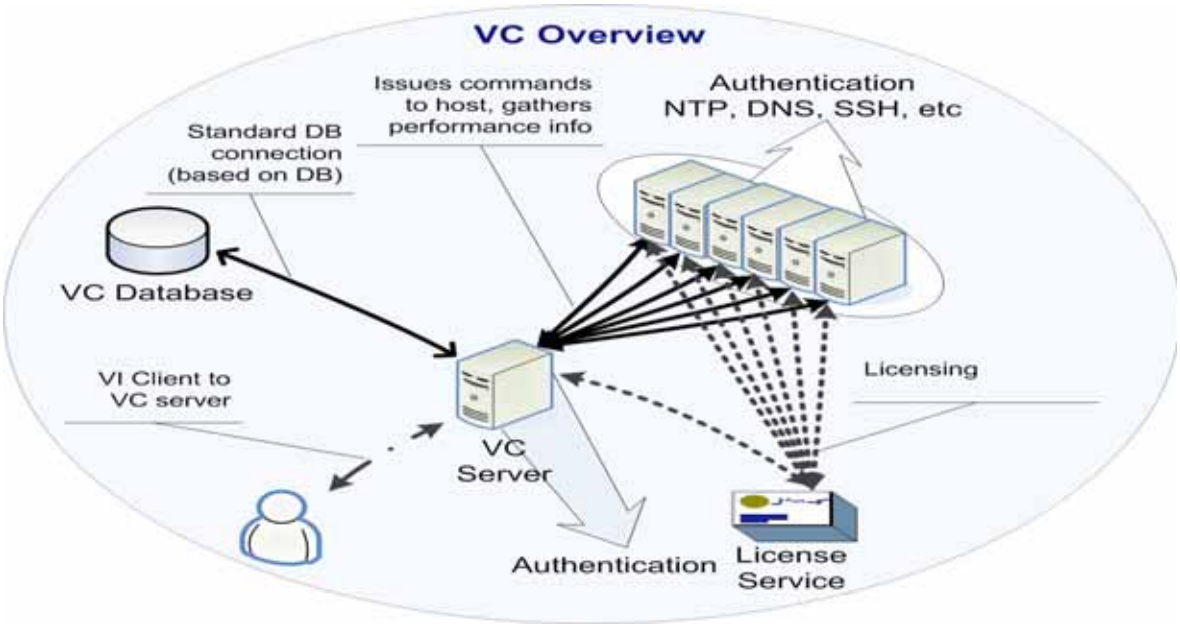
Level 4 is the highest logging level and collects samples for any metric supported by virtual center. Total metrics collected for a single quad processor host of average config is a small jump from level 3 to 174.

The size of the VC database, while taking up 3 pages here, is really not all that important to your overall infrastructure design. We simply wanted to provide you with some info on sizing, so when your DBA explains that you need 2 controllers, with 5 logical partitions to support your VC db, and he needs his sizing info, you can show him some simple math, and move on to your next part of the design… Like VirtualCenter system design.

VirtualCenter System Design

In an ESX 3.0 environment there are a number of communications paths you need to worry about when designing the environment. The first, and most obvious, is VC to ESX host communications. These are the "no brainers" with which most people are familiar. Essentially, the VC server sends commands to ESX hosts for execution (such as starting a virtual machine), and performance data is shipped to the VC server for each host being managed by that server.

Figure 4- 6: VirtualCenter Overview



These are the communications that most people worry about, but the reality is that host servers will also communicate to DNS (really dependent on this if using HA services in the cluster), NTP, and Authentication for shell logins etc. In addition, you need to think about the connection from each host to the License server (when using a centralized license server) and for the VC server to the VC database. Finally, the VC server will also communicate with Active Directory (or possibly Windows local groups) for authorization to log in to the VC client and perform operations/issue commands within VC.

To show a simple example let's assume you want to VMotion a VM from one host server to another. In doing so the following steps (simplified) will show how almost all the communication paths are used:

1. User launches VI Client and connects to the VC server and inputs credentials.
2. VC server contacts the database.
3. VC server checks account against DB for this user's rights to log on and matches to roles in VC (Windows auth done at this point)

4. VI Client still in communication with VC server begins to load inventory in the GUI from the VC server that is reading it from the DB.
5. User selects a VM and issues a VMotion command.
6. Licensing is checked on those hosts along with prerequisite checks for VMotion compatibility.
7. VMotion begins (which we won't get into here).
8. Progress is reported to VC DB, errors/logs are written, and normal VMotion traffic happens between host servers.
9. Change is reflected is in GUI once VMotion is completed, and also VM location info is written into the DB

As you can see, there is a lot of chatter going on in this environment. Not heavy traffic, mind you, but enough to be sure that if one of the links is broken it can make for a fun night of troubleshooting.

VC Server Locations in the Enterprise

Now that we have a basic understanding of the communication paths in ESX environments we have to look at the decisions that need to be made about where to place the VC Server or Servers on your network. If you have a single datacenter or single central point that ESX will be used, then your decision is already made. But, if you will have ESX in multiple locations you will need to decide if you will have a VC server in each location with an ESX server, a VC server in only large sites, or a VC server in only a central site.

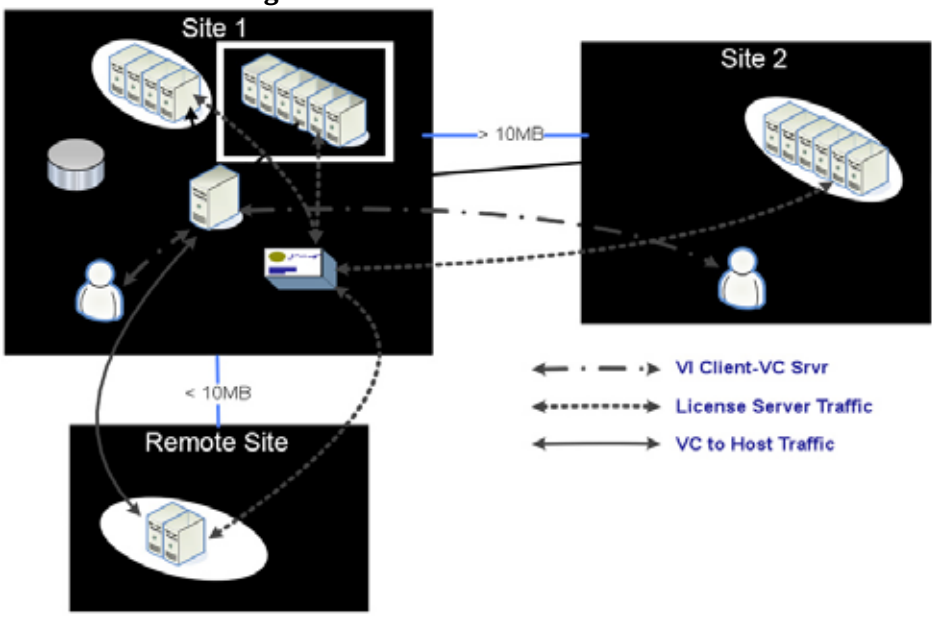
This decision should be made with the following items taken into account:

- Bandwidth available between sites containing ESX servers
- Number of ESX servers in a location
- Location of administrators for the ESX servers in any given location
- DR/BC Plans

Centralized VirtualCenter

First let's look at a centralized VC Model with both high-speed and low-speed connected sites. In this scenario, we assume that ESX servers are located at both of the remote sites and that the administrator has decided to manage them all via a single VC server and DB.

Figure 4- 7: Centralized VirtualCenter



Advantages of the Centralized VirtualCenter Model

- Single management point for all clusters/hosts
- Single place to security for your whole ESX environment
- Single server to manage, backup, patch, update, etc…
- Administrators from remote sites can still access the VC client if they have been given rights

Disadvantages of the Centralized VirtualCenter Model

- Almost unusable, if not completely unusable, over low bandwidth links
- May require changes in how templates are deployed, updated, and managed
- Not the best for DR/Hot site configurations

The Centralized VC model has a number of benefits. You have one point of management for every server, you have single place to set security in the entire virtual environment, and only one server and database to manage, update, patch, and monitor. In this configuration, remote administrators (like from Site 2 in this example) can still access the VC server from their site and manage their servers if they have been given proper permissions.

The drawback to this design is that low speed/low bandwidth links can cause sporadic issues in VirtualCenter, such as disconnected servers, slow downs in getting performance information, and sluggish response when issuing commands to be executed on the host. If a site has a small connection to the central location (like a single T-1) VC may still work, but timeouts and other issues when communicating over a T-1 used for other traffic (as seen in some environments) makes VC almost unusable.

A quick note here; recently I was at a VMware client with a 10+ Mb ATM link that was lightly used. Their centralized VC model works fairly well with just a few things/workarounds to make template deployment and updates easy to use. But, just a few weeks earlier I stopped at another client, using almost the same configuration with a 35 Mb link that was totally saturated and has had nothing but issues with the centralized model. So it's not just the link size, but its available bandwidth that you need to consider.

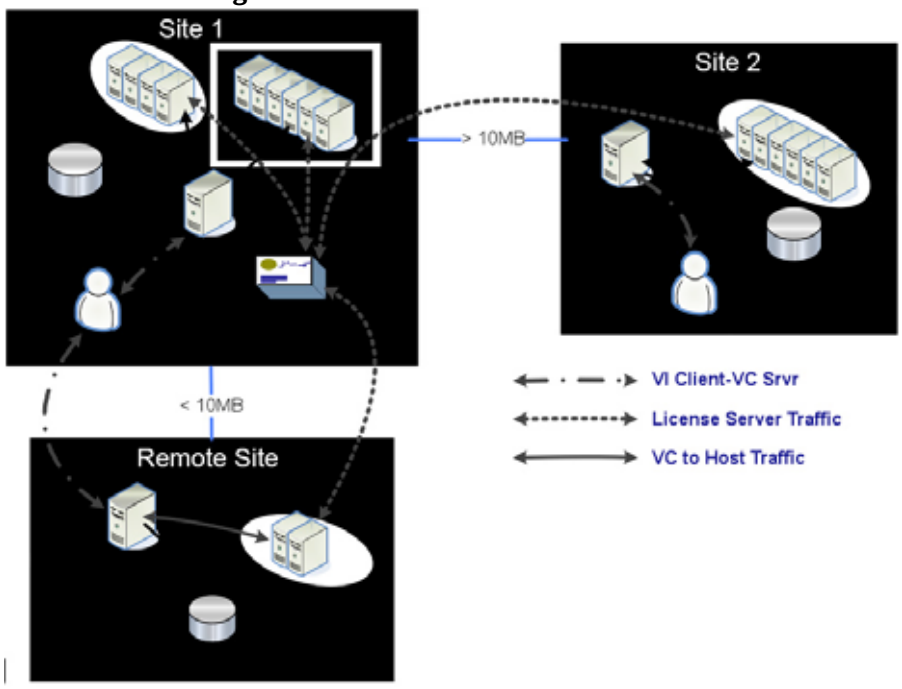
Finally, with centralized models your template deployments can become an issue. Generally template deployment is very bandwidth intensive process. If done over the network (like templates being centralized then copied to remote VMFS volumes) you could wind up copying gigs and gigs of data. Of course this often fails right out of the gate in low speed connections. So centralizing like this requires that you configure templates at each location anyway, to speed template deployment. In some cases, administrators will resort to NOT using the template functionality in VC and instead leaving a VM or VM's on a remote server and VMFS volume that acts as a template, but they simply clone it or copy it using shell commands.

Decentralized VirtualCenter Servers

In this design option, we use a VC server and database at each location. We assume here that we have centralized administrators for Site 1 and possibly some

offshoot remote locations. Site 2 is a second datacenter with its own set of administrators and also has its own VirtualCenter server. You will notice that we show the License server hosted centrally. License services were discussed previously in this chapter, and here we have decided that the centralized licensing will be used since all sites are sharing the same licenses, instead of a bunch of individual purchases managed by site.

Figure 4- 8: Decentralized VirtualCenter



You will also notice in this image that Admin in Site 1 is connecting to the VirtualCenter server in the Remote site. If the link has low latency and plenty of bandwidth, this often works fine using the VI client. This configuration is slower than a LAN, obviously, but still usable. If this link is saturated or extremely slow Administrators will often load the VI client right on the VC server, then just use MS Terminal Services to access the server and run the VI client for managing that remote environment.

Advantages of the Decentralized VirtualCenter Model

- Works well over limited bandwidth links
- Ability to decentralize security for different environment owners
- Works best for DR/Hot Site configurations

- Templates are localized in each site and can be used "traditionally"

Disadvantages of the Decentralized VirtualCenter Model

- Multiple servers to manage, update, patch, backup, etc…
- Decentralized security points, must configured in multiple locations
- Disjointed virtual infrastructure creating datacenter based silos

The obvious benefit of the decentralized model is that it is simple. You bring up an ESX 3.x cluster in a new site, and you bring up its VirtualCenter management components and configure them. Replicate your design and procedures from place to place, and you are done. The biggest disadvantage is that the management is decentralized now, but is this really a big deal? If you think about other management tools in your environment like HP SIM, or HP Openview or IBM Director or Desktop management tools you have, are they completely centralized? Often not. In a lot of cases these are decentralized since the staff in that location handles / uses the management systems. So, while decentralizing can have its drawbacks, it is a model used more often than not in larger, multi-site environments.

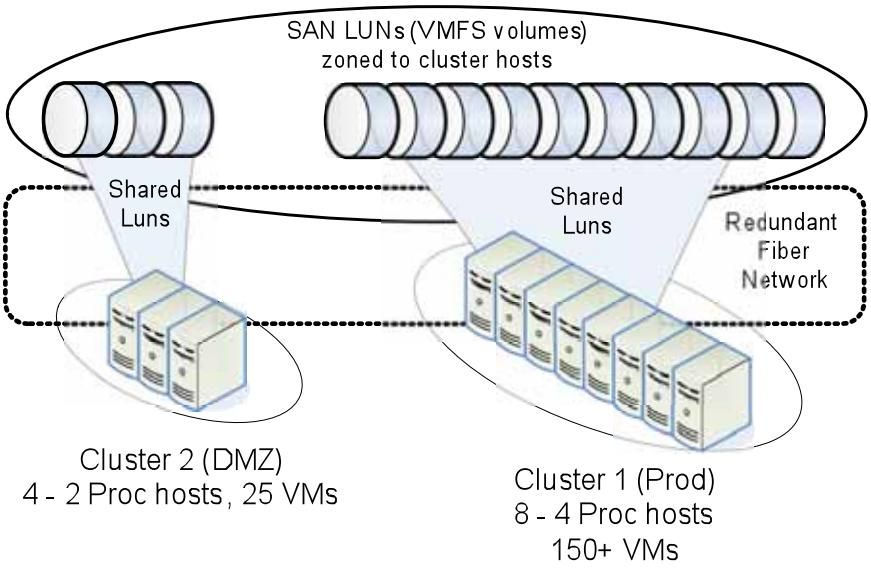
An important note here about remote sites and design alternatives- from a VirtualCenter perspective, a 50 or 100Mb WAN link (non-saturated) is as good as a LAN connection. Most of the information going back and forth between hosts and the VC server is fairly light, and a centralized model for VC works just fine (barring any potential template issues previously stated) with this type of available bandwidth. The scenarios we are describing is sub 35Mb connections and specifically things like T-1s, E-1's or aggregated links of the like. In addition, it is important to note that this does not mean you can simply span an ESX cluster across sites, but that VC (as discussed here) can be used to manage clusters in different sites. Clusters and their design considerations are discussed later in this chapter.

Cluster Overview

Clusters in ESX 3.X are really simple to understand once you break it down to the basic functions of a cluster and the required components to make these functions work. From a functional stand point, a cluster is a set of like-configured ESX Servers, in the same site/datacenter, that share common storage and networks and have VMotion enabled across the hosts. There are some

important configurations and requirements to make our simple definition a reality, but the basics concepts are detailed below along with a drawing of a simple 2 cluster environment.

Figure 4- 9: Cluster



Let's take a look at these two clusters and review each of the decisions that went into making these two different configurations. The first cluster (prod cluster) is a set of 8 quad processor systems, hosting 150+ Virtual Machines. As you can see, all 8 servers share the same VMFS volumes (a total of 10 VMFS volumes in this case). VM's in this cluster only reside on those 10 LUN's. The second cluster is a smaller cluster using 3 shared LUN's and 3 dual processor hosts hosting 25 Virtual Machines.

The first item to notice is the size of the servers, Dual processor vs. Quad processor configurations. When doing cluster design the first question is how many VM's do you want to host? In this case they have decided to separate the DMZ VM's from the Prod VM's. We can assume that this is due to physical network connectivity and/or security policies in the organization to keep DMZ separate from Prod. That creates two distinct pools of Virtual Machines to be hosted and therefore two distinct clusters. You see, since the configurations have to be split (from a network perspective) the configuration of the hosts, not to mention the physical connectivity to the network will be different, therefore VM's cannot be VMotioned / mixed between these hosts.

Once the physical limitation/configuration was decided on then we have to run the number of expected VM's (25 in the DMZ and 150+ in the Prod) through a simple exercise taking into account the following:

1. ESX 3.x clusters are limited to 16 servers per cluster (based on HA limits).
2. If not using HA and only using DRS, clusters are limited to 32 nodes.
3. Reality is that a common best practice is to limit cluster size to between 10 and 12 hosts.
4. Estimated average VM to Processor ratio (in this case we will assume 6)
5. The amount of redundancy we want to have in the farm from a host level (Newhart?) In this case we assume at least N+1.
6. Storage will be allocated as needed; all volumes will be VMFS and shared amongst hosts.

Hardware selections (dual, quads, 8-ways, multi-cores etc…) are discussed in Chapter 3. Here we are assuming a single core for simplicity, for a "core" discussion please jump back to chapter 3. Assuming that we will host 6 VM's per processor we know that the DMZ will need at least 4 (and a fraction) to host 25 Virtual Machines. In the prod environment we want to host about 150+ virtual machines. Using that same 6:1 ratio we can estimate that we need about 25 processors not including redundancy. Knowing the number of processors required for each environment we can then decide on the form factor based on costs and amount of redundancy.

For the DMZ we need about 4 processors to host the expected VM's. In this case we can buy 2 quad processor systems getting us 8 total processors, or we can purchase 3 dual processor systems for a total of 6 processors. In the quad processor case we have almost 4 processors completely unused. Yes, I know ESX will still use them, just not fully, but that is not the point. The point is we have just over purchased by purchasing way more machine that we need just to satisfy the N+1 need. In the dual processor configuration the active VM's can basically be hosted on 2 dual processor servers requiring a 3rd to act as the N+1 server which (in almost every case) 3 dual processor servers are much less expensive than 2 quad processor servers.

On the Production cluster we have to host 150+ VM's or about 24 or 25 processors. Again this can be done with quad processors for a total of 6 or 7 hosts. Or it can be done in a dual processor configuration requiring 13 hosts. When using dual processor configs with this cluster we are doubling the number of fiber and network connections, and their cost may come into play In addition, the concept of managing twice as many servers and being close to the 16 server limit in the cluster is an issue.

The decision in this example was made to go with a quad processor system (needing 7 hosts) plus an extra to match the N+1 requirement, while still having room to expand the cluster without starting a new one. Now that we have figured out the hardware needs, we will focus on the Cluster options in the design.

Cluster Design Options

In chapter 3 we looked at mixing workloads (like test and dev with production etc.) and how that can affect your network connectivity requirements or change the hardware you are about to purchase. Assuming that you are past the point of deciding what is going to be hosted on these servers, let's review cluster requirements then look at the design alternatives:

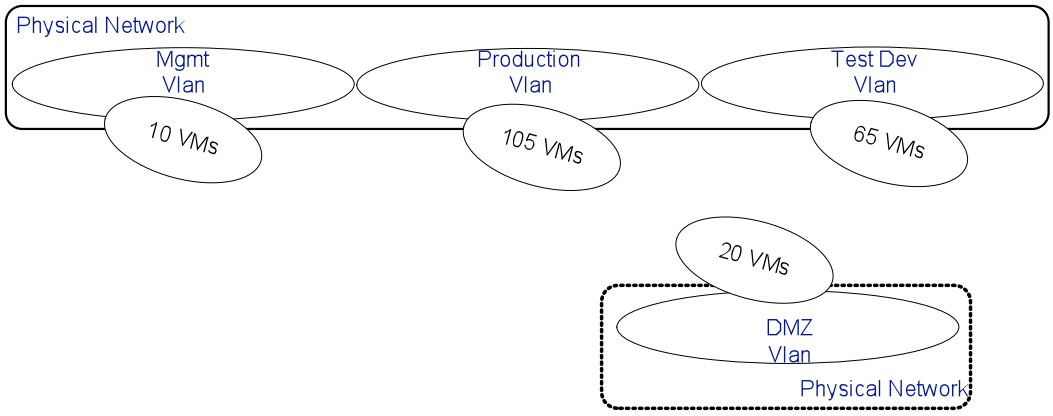
- All servers in a cluster will need to be zoned to the same VMFS storage.
- Servers will need to be pretty much identically configured:
- Processors will need to be similar if not identical (just do a search on VMware's site for Processors and VMotion compatibility, there's a list).
- Like major software version (3.x) – try to keep the patch level the same to minimize issues.
- Licensing will need to be identical (to allow for VMotion, DRS, HA, or any other options you wish to use).
- Network connectivity will need to be the same (physical connections to the same network or logical networks when using VLAN tagging) Virtual Switch names and configurations should be identical.

- Should all be in the same site as cross site VMotion is not the reality for any site connected via a WAN link.

Knowing these requirements we are still left with a lot of alternatives. Let's create a sample environment containing the following:

- 2 Logical networks (VLANs) test and prod on the internal network
- 1 Separate physical network for the DMZ
- 1 Mgmt network (VLAN) used for out of band mgmt (like ILO, DRAC etc), and other mgmt tools
- VM's will be required in each of the environments (Mgmt, DMZ, Test and Prod)
- 200 VM's in total with 10 VM's, 20 VM's, 65 VM's and 105 VM's respectively

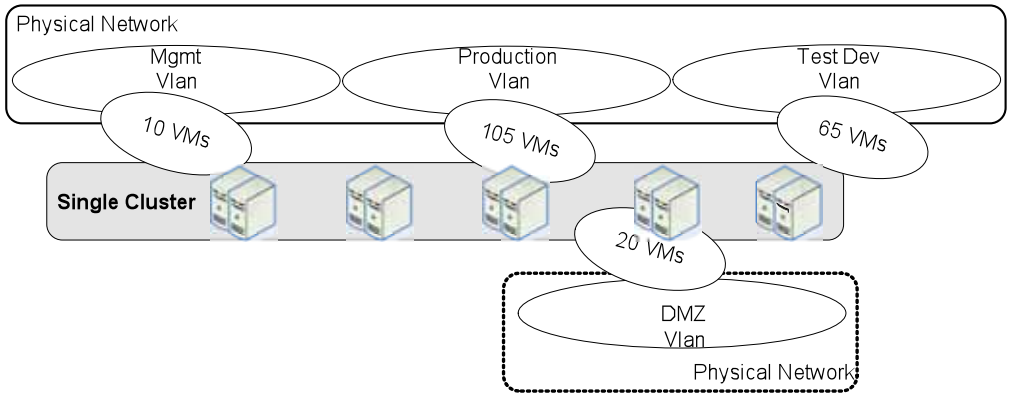
Figure 4- 10: VM Environment



Single Cluster Design

The first option is to create a single cluster that will host VM's for all environments on the network. This cluster will require about 10 ESX servers assuming quad processor hosts with an average load of 24 VM's per server maximum and N+1 redundancy.

Figure 4- 11: Single Cluster



The big benefit of this design is that you have a single cluster for all VM's. This allows you to leverage a single set of servers hosting different workloads for better performance and allowing you the ultimate virtual infrastructure; any host can host any VM for any network segment. In addition, you reduce the number of hosts required and eliminate unique silos and configurations of ESX in your environment.

While this environment simplifies the mgmt of the virtual infrastructure, in the long run it does increase the complexity of the system and hosts. In our example the host is going to have physical network connections to both the internal network and the DMZ. In addition, the Virtual Switches on the internal network will have multiple port groups configured for the 3 logical VLAN's. Each time a VM is added the proper Virtual Switch will have to be selected to ensure proper network connectivity of the VM. While this sounds a little daunting it is not that much to handle, but it increases the network complexity from a trunking/vlaning perspective and may get push back from your security team for hosting DMZ VM's on the "inside" of the network.

Advantages of the Single Cluster Model

- Single type of server/configuration to manage
- Ability to optimally balance work loads
- No siloing/underutilization do to siloing
- Reduced number of hosts because capacity used for redundancy is used across all environments

- Easier to manage in the long run
- Lowest capital cost upfront

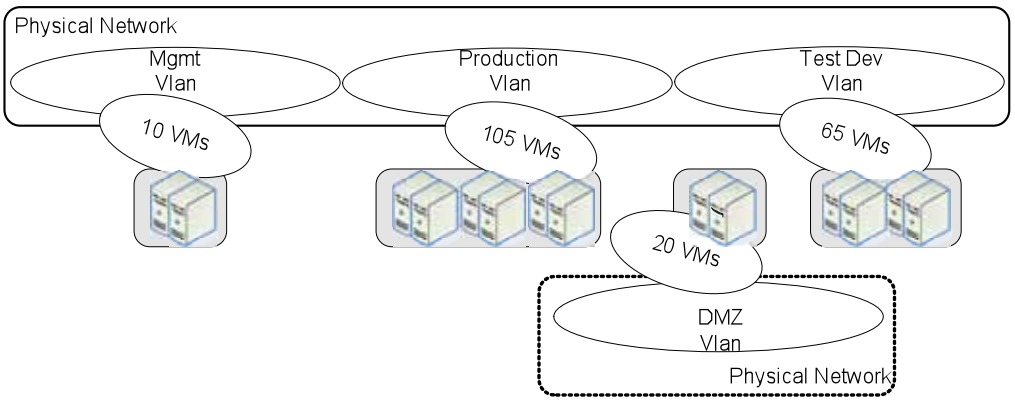
Disadvantages of the Single Cluster Model

- More complex from a host networking perspective
- Security teams will often say no to this just out of principal
- Requires more management of resources and resource pools due to the large number of environments being hosted

Silo'd Cluster Design

In this alternative a cluster is created for each of the unique network environments. It should be noted that each ESX host in the environment will still have a network connection (or two) to the mgmt or production network for its service console, but here we are just talking about VM connectivity. As you can see, we now have 4 clusters for this environment. The mgmt network will have 2 ESX hosts, the production environment will require 6 hosts, the DMZ will have 2 hosts, and the test and dev cluster will have 4 hosts. Essentially the environment is silo'd by the network to which the VM's will connect. This eventually results in 4 clusters and 14 hosts.

Figure 4- 12: Silo' Cluster



In addition to network separation, the cluster design will take into account zoning of the VMFS storage. Each environment will have its own LUN's zoned to it and should not be crossed.

The major benefit of this design is that it is the easiest to implement when you first get into ESX Server. The major drawback is that once implemented it is a real pain to get out of for both technology and political reasons. Selling this type of cluster design in an environment is easy to do; migrating to a more consolidated model later on can be cumbersome when you start talking about changing the LUN zoning, reconfiguring the switch ports servers are connected to, migrating VM's etc. In addition, when you silo servers that are alike, such as the large number of web servers in the DMZ, tool servers in the mgmt area, or unused test servers in the test environment you will not be able to optimally mix differing work load types, meaning you will run into similar bottlenecks on hosts running similar types of VM's.

Advantages of the Silo'd Cluster Model

- Simplest option from an individual host networking perspective
- Increased number of hosts because capacity used for redundancy is not leveraged across all environments
- Easiest to sell to internal teams during an initial implementation
- Requires very little with regards to resource management since A: there is more available hosts and B: no VM's from differing environments

Disadvantages of the Silo'd Cluster Model

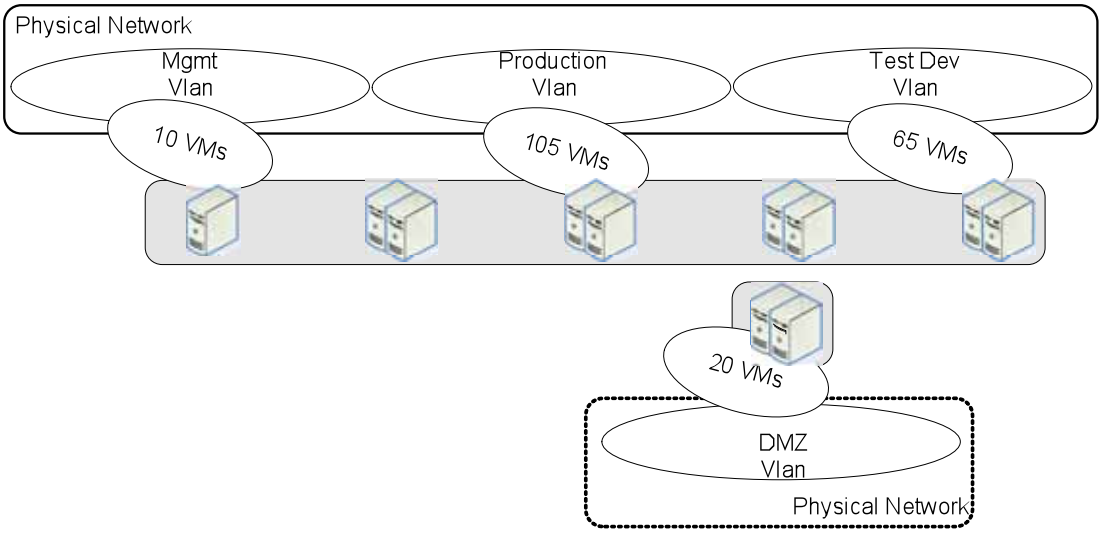
- Multiple configurations and clusters to manage
- Not an optimal configuration for balancing work loads
- Increased under utilization due to more redundant capacity
- Highest capital cost upfront due to the increase in hosts

Middle Ground- Some Silo'd Clusters

Often a good design comprise is to only silo off some of the clusters. One that is used often is the separation of DMZ clusters from internal clusters. Using the same fictional network this option would segregate the DMZ from the Internal VM's, creating 2 clusters with 9 hosts in one and two in the other. In some ESX

environments they only have internal ESX hosts and may just separate the Dev and Test VM's from prod VM's. In any of these cases, the trick is to find the economies of scale, if you have an environment that is significantly large enough to warrant its own cluster (lets say 8, 10 or 12 hosts) then the cost for redundancy is minimal and it wouldn't really hurt anything to remove it. Using our example and only siloing off the DMZ VM's we wind up with just 11 hosts (1 more than a single cluster design) and still keep the security guys happy, but we could have just as easily folded the DMZ into the Prod cluster and had the Test/Dev cluster rolled out to its own.

Figure 4- 13: Mixed Cluster Design



The advantage here is that it allows for minimal siloing, but still keeps cost as low as possible by leveraging the redundancy capacity across numerous environments. It also is a good balance in that it keeps different types of workloads running on the hosts to get close to optimal configuration for resource usage.

Advantages of having some Silo'd Clusters

- Relatively easy to sell to internal teams
- Reduced number of hosts when compared to the completely silo'd model

- Reduces the number of silos to manage when compared to the previous model
- Good balance between the two primary models

Disadvantages of having some Silo'd Clusters

- Some teams/app owners may feel they need their own cluster
- Just as complex from a networking perspective as the first option
- Still requires resource management in any environments mixed (in our example, prod, Test, and Mgmt).
- Siloing Test and Dev or DMZ type workloads decreases resource utilization because of similar workload characteristics in these environments

Distributed Resource Services (DRS) Enabled Clusters

When creating clusters in ESX 3, one of the options is to enable the cluster for DRS. DRS is essentially a load leveling tool that uses resource utilization from the hosts and VM's to create recommendations for Virtual Machine placement. DRS clusters are limited to 32 nodes maximum, but that number is essentially artificial as most clusters that use DRS will also use HA services (discussed next) which are limited to 16 nodes maximum. Why the conflict? Well DRS is a product written by one group and HA is a product licensed from another ISV then modified for VMware. The long and short is that if you plan on using DRS and HA that 32 node limit goes out the window, and in most cases we even stay a little below 16 nodes.

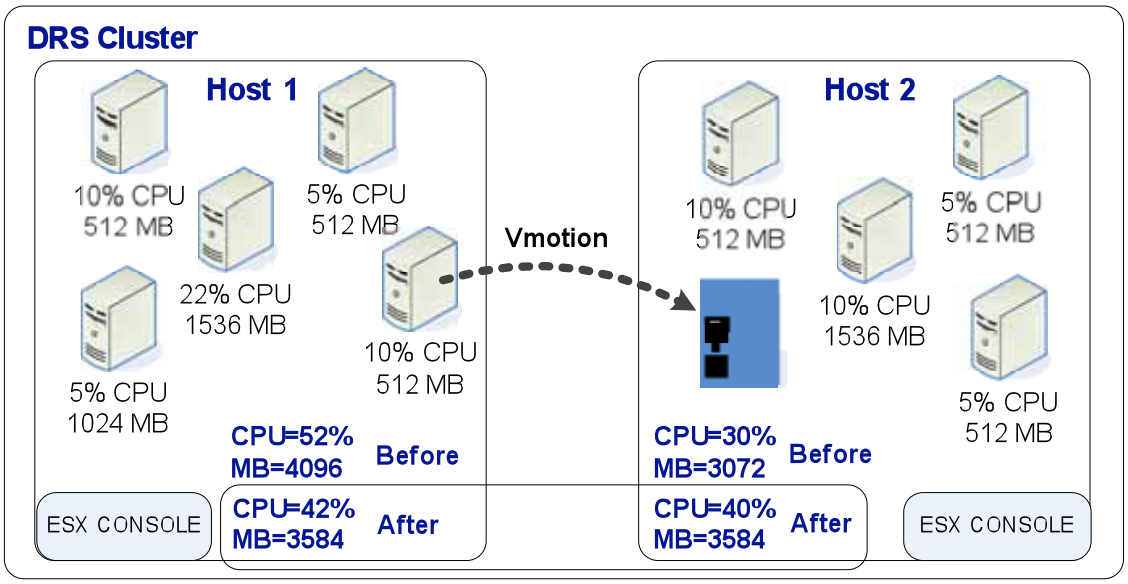
Anyway, DRS creates recommendations based on load that can be used as simple recommendations (seen in the VirtualCenter client) and acted on manually, or they can be automated at different levels and allowed to automatically balance VM load across the environment.

DRS is controlled, managed and executed by VirtualCenter, unlike HA that is managed and configured by VirtualCenter but runs independently (as an agent) on each host. DRS's essential functionality is to balance CPU and Memory load on the hosts by moving VM's from high-utilized hosts to less utilized hosts. DRS does not take into account network or disk utilization (throughput or IO).

Right now DRS's focus is on processor and memory resources which happen to be the two major bottlenecks in 99% of ESX servers.

To understand the advanced features of DRS, we first should look at how it works from the basic recommendations perspective.

Figure 4- 14: DRS Cluster



In this example we have two nodes in a DRS cluster. Host 1 has 52% CPU utilization and about 4GB of ram in use, while Host 2 has 30% CPU in use and about 3GB. VirtualCenter (and specifically the DRS components) sees this differential in utilization and attempts to determine which VM moves would balance the load. Here (a simplistic model mind you) the 10% CPU and 512MB VM gets moved to the less utilized server. This results in the two servers almost having the same load. The system does not move a VM like the 22% utilized VM, since it would just create an imbalance where Host 2 is more utilized and more moves would be needed to balance the load.

Obviously that is a simplistic look at how DRS works, but the important items to note (at this point) is that this entire process is controlled by VirtualCenter working off of performance information it collects from the ESX hosts. It is also important to note that you have some control about the aggressiveness of these moves and can control whether this process is fully automated, partially

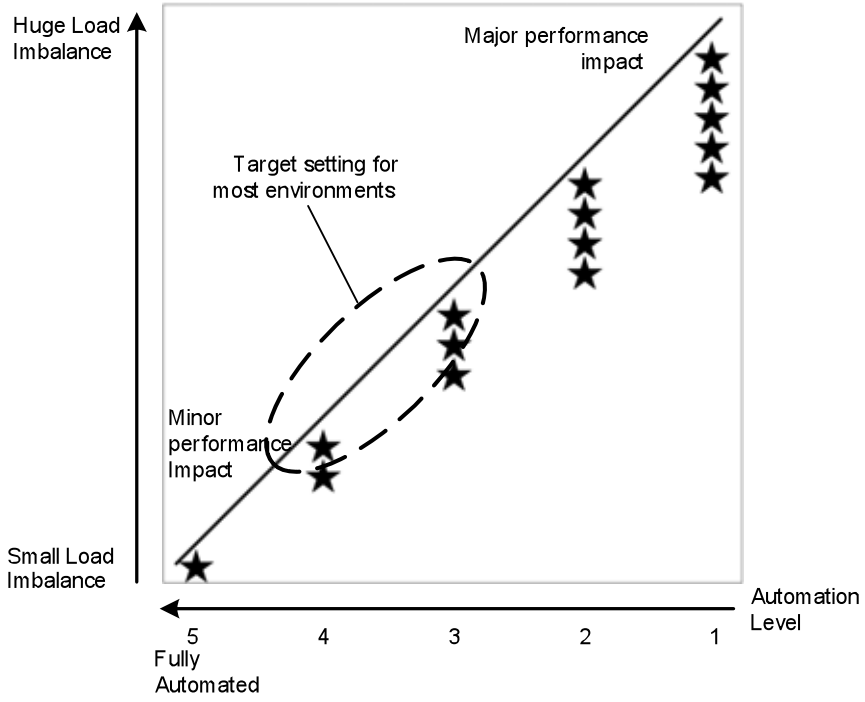
automated, or in completely manual mode, which basically just sends recommendations to the admin and he decides whether to apply the recommendation or not.

DRS Recommendations

VirtualCenter constantly reevaluates load in a DRS cluster. Basically, DRS is driven by a default interval of 5 minutes or when a host is added or removed from the cluster thereby affecting the load / capacity of the environment significantly. So in any given hour DRS looks at load in a cluster 12 times, then based on this creates a series of prioritized recommendations to level load across the cluster.

These recommendations are prioritized by DRS and shown to the administrators using a Star ranking. These "stars' also correspond (inversely) to the aggressiveness of the DRS automation mode. So, star rankings can be seen like this:

Figure 4- 15: DRS Recommendations



Looking at the image you can see that a recommendation with a "5 Star" rating means that there is a huge load imbalance and making this change (moving this one VM) will solve the imbalance. As you move down the ratings to a 1 star rating, you have a very minor imbalance from which you will see almost no performance impact. The automation levels are exactly the inverse of this. Meaning at level 1 automation (the most conservative) it will only automatically apply recommendations with 5 stars, while the level 5 automation will apply any recommendation (1 star or greater).

The reality is that these recommendations are movement recommendations for single VM's. In any production environment you are very unlikely to see a 5 star rating. In most cases, a 5 star rating means you have moved a VM (manually) in a DRS environment violating some type of affinity rule (discussed later) but for straight utilization it is almost impossible to have a single VM cause that much of an imbalance in your environment, so level 1 automation is almost useless and can mess up a manual change you made for a reason. If you want to be fully auto-mated, level 4 is just as good as any other.

In any case, you should gain a level of comfort with DRS automation. Maybe start at manual, look at the recommendations, apply them, then start to move the automation level up and gain a level of comfort with each automation setting.

Rules for Load Balanced or "Special" VM's

When hosting load balanced VM's running on an ESX cluster (two VM's maybe using Windows load balancing) or running multiple VM's that have dependencies on each other (like an application server that needs its database server up to be useful), you may need to set some special rules to ensure VM's are placed on proper hosts.

DRS Affinity Rules

Affinity Rules, when talking about DRS affinity, should not be confused with processor affinity in at the VM or host level. VM level processor affinity assigns a VM to a specific processor on a host. DRS affinity allows you to set rules so that multiple VM's are always on the same host, or always kept on separate hosts.

If you select the DRS rule to "Keep Virtual Machines Together" the rule wizard allows you to add multiple VM's to a list affected by this rule. Essentially, DRS will take this into consideration when making recommendations and assume that these VM's need to be on the same host, and when moved, they are always moved together. This is most often used when VM's are part of an application set such as a front end and backend server, and you want to keep the traffic between the servers local to the host on the virtual switch.

The opposite side of that rule is to "Separate Virtual Machines". This will ensure that two or more VM's are never placed on the same host by DRS. This is most often used when you have a pair of load balanced VM's to provide redundancy for an application. If DRS were to move the VM's onto the same host, and there was a host failure, you basically lose your hardware redundancy and both VM's can go down at the same time. By configuring this DRS properly you will ensure that it never moves the selected VM's onto the same host.

DRS Automation Rules for Specific VM's

DRS automation settings can be overridden on a per VM basis. Essentially, allowing you to automate the entire environment but exclude certain VM's from being moved around by DRS. This is often done for "sensitive" VM's in an environment. A perfect example of this is Virtual Machines in a pharmaceutical environment. Some of these VM's are running pharmacy apps that are governed by laws saying that the server (VM) its network configs, disk, and paths to disk etc., can be audited at almost any time. In these environments they may still want to use DRS to level load, but will need to override the DRS automation for this VM and never move it just to level load.

To set these rules you can edit the cluster settings and edit the Virtual Machine options. This then allows you to select the specific VM's to override DRS settings and set to one of 5 automation levels:

- Manual: Recommendations are created but have to be applied by a human; this is often used to allow for change control on the move for auditing purposes.
- Partially Automated: This is essentially like a level 1 or level automation for this VM, only move it if it corrects a large imbalance in cluster load.

- Fully automated: Which is essentially apply level 5 automation to this VM
- Default: Which just accepts the Cluster's DRS settings and applies them to the VM.
- Disabled: Which ignores DRS for these VM's. The VM's load is calculated as part of the host load, but no recommendations are ever made for this VM and other VM's are used to level host load.

These rules are great and allow you some granularity, but they come with a caveat. They should be used sparingly. If you have a DRS cluster with 20 VM's and have DRS automated in the cluster but have disabled it for 17 VM's or even 10 VM's, you have effectively limited the balancing ability greatly. These settings should be used judiciously and in probably no more than 25-35% of the environment at most. Greater than that and you are limiting your options greatly and incurring unneeded overhead.

Maintenance Mode

Not that this is a huge deal, but something we want to note here is a function/state of a cluster node known as maintenance mode. You can set a cluster member to maintenance mode which essentially removes its resources from the cluster's resource pools and from the available capacity in the DRS pool. This then forces DRS/VirtualCenter to migrate (VMotion) all of the VM's off of the affected server and keep VM's off of it until it is removed from maintenance mode.

This is essentially a nice way to migrate VM's, allow you to work on, upgrade, bounce etc., the Host while not impacting the VM's in the environment.

High Availability (HA) Services

High Availability services is a service provided within VirtualCenter managed clusters that provides redundancy for hardware failures within the Cluster. This service should really be called "Faster Recovery Services," but I am sure some marketing department put the smack down on that one. The reason for our alternate name recommendation is that HA does not really provide true High Availability. In IT we consider clusters (where a process fails over from one

host to another seamlessly) as true High Availability. VMware's HA does not do this. VMware's HA offers a fast/automated recovery for VM's running on a host that has failed or been isolated from the network.

The HA service (as it sits today) is really a software package from Legato called the Automated Availability Manager or AAM for short. AAM is a pretty interesting product if you get to reading about it, but from a VMware perspective you see about 1/10th of the full AAM functionality. In the VMware world, VirtualCenter, instead of the AAM console, is used to configure HA services for HA enabled Clusters. Unlike DRS, VirtualCenter does not initiate VM moves but instead configures the AAM agents running on the hosts and allows these agents to manage themselves and look for host failures.

This is probably a good time to explain to you (if you don't already know) how HA failures affect Virtual Machines. Some people in the industry assume that having HA configured during a host failure means that the VM's are never offline and that when a host fails the VMotion functionality is invoked and the VM's seamlessly migrate to the other host machines. This is completely untrue. VMotion requires that the source and target host machines be up and running. And since HA only takes action AFTER a host failure, it is impossible to VMotion any VM's to another host. In addition to this, the VM's are essentially in a powered off state when HA kicks in and were more than likely powered down hard as their host has just crashed… Getting a warm fuzzy yet?

Not to worry. It takes about 15 seconds for HA services to realize a failure and begin restarting VM's on other hosts in the cluster. The really cool thing about this is that the recovery time for VM's (have them back up and running on good hardware) can generally be counted in minutes. Tests we ran in the early 3.0 days show 20+ VM's restarting from a failed host in less than 5 minutes. Try that on a physical piece of hardware with a bad motherboard.

HA Key Architecture Notables

So let's review some of the key attributes of HA services:

- Clusters using HA can contain up to 16 nodes.

- HA configurations allow you to specify the number of host failures for your Cluster (1-4).
- HA offers what it calls admission controls to protect against over utilization.
- HA has isolation detection to allow a host to determine when it has lost network connectivity.
- Key files and logs for HA can be found in the /opt/LGTOaam512/ directory (see, Legato…).
- HA does not need VirtualCenter running to restart VM's.
- HA is highly dependent on DNS, meaning all host names in the cluster must be able to be resolved via DNS. (Most HA issues are really DNS related.)
- HA does not detect (at this point) SAN storage losses or Network connectivity losses that are not Console NIC's.

When configuring/designing your HA cluster you have a few key decisions to make. The first is to determine the number of host failures you feel the system can absorb. This comes back to your decisions on N+x in your environment. As an example let's assume you have enough VM's to pretty much load up a 10 Host cluster. Your environment requires that you supply N+1 redundancy. So you build an 11 host cluster to meet the requirement. When configuring HA services you will be asked to specify the number of Host failures for which you want to guarantee VM failover. In this example your setting would be 1.

HA allows you to configure a cluster for anywhere from 1 to 4 host failures. The limitation of resides in a limit built in to provide for only 4 "Primaries" or Primary Nodes. These nodes in Legato terms are where resources can be moved. ESX incurs an overhead for each primary added, and the acceptable overhead (per VMware) falls at a maximum of 4 hosts for this purpose in a 16 node cluster.

While the number of failures allowed is an interesting number, don't just go and set it to the highest number yet. The reality is you need to take into account the day to day load of your environment. Let's assume you built the cluster we used in the previous example. You have enough VM load to run 10 hosts at 80% utilization. You then design and built the cluster with 11 hosts to provide N+1.

Here, your setting would seem to be 1 for the maximum number of failures, but let's do some math and see if you can go any higher.

Maximum number of acceptable host failures

While it is easy in the physical world to say "N+1" is required, in the virtual world it is not a one to one ratio. Your 11 server cluster (using DRS and HA) will have VM's running on all hosts to utilize their resources when available. But if we do some quick math around your 11 host environment built to handle 10 hosts at 80% we find some interesting things:

- We assume that each server has 100 units of capacity (converting % of utilization to a number).
- 10 servers have 1000 total capacity units available. 80% of this available capacity is 800 units.
- Your environment (example above) will use 800 units of capacity with 11 hosts that have a total of 1100 units available.
- 11 hosts splitting up those 800 units results in about 72.7 units each or 72.7% utilization.
- If one of these 11 servers were to fail, its 72.7 units are then split amongst 10 hosts (taking us back to 80% util).
- But don't we still have 200 units available? -more than enough to take another host failure.

The idea here is to determine if 80% is truly the maximum or, if during a failure, you can run at 87 or 90 or even 100%+ utilization… Right now our example environment has enough capacity before each server reaches 100% utilization to handle at least 2 more server failures.

The trick with determining the acceptable % is first dictating a policy in the environment for operations during a failure (IE: is it ok to run at 100% or close to it or do you have to stop at 80%, 90% whatever?). Once you do that, you can do the math yourself, determine the bottleneck in the farm as it grows (generally memory but sometimes proc), and then set the maximum number of failures.

Why is this maximum number of failures important? Well, not just because it could save your job, but because of a second option in the HA configurations, Admission Control. Admission Control allows you to dictate what happens when you reach the number of failures configured in the HA environment.

The first option is "Do not power on virtual machines if they violate availability constraints." That is really a big mouthful for "after we hit the limit number of configured failure, and another host fails, don't restart the next failures VM's." This setting essentially will force the system to obey your design decision (the acceptable amount of utilization during a failure) and keep from over utilizing the remaining hosts in the cluster during a massive failure. When this option is selected it also keeps you from reverting snapshots, migrating VM's into the cluster, and reconfiguring any VM's for more CPU or Memory.

The second option for Admission Control (the default) is to "allow Virtual Machines to be powered on even if they violate availability constraints". Basically ignoring your setting, and continue to start VM's even if your farm has sustained the number of expected failures or more.

Detection of a failure

HA is a pretty cool animal once you think about it. The VirtualCenter server allows you to configure your settings, and it takes off from there monitoring all the hosts and looking for failures. But how does it work? The bottom end of HA is an agent that runs on each host in this cluster. This agent stores information (in memory) about the nodes in the cluster and the controller in the cluster. The first node in the cluster acts as the controller. This controller interprets HA rules, can initiate failover actions, and has redundancy provided for its operations by other hosts in the cluster. This controller has a redundant setup in which another node will take over for the controller in the event of a controller failure (one of the reasons for storing HA info on each node in the cluster).

Failures of a node in the cluster are determined by a heartbeat that utilizes the Console network interface. It is HIGHLY advised that you provide redundant NICs for the console to limit failovers in the event of a single Console NIC failure. The heartbeat for HA is monitored by all hosts. Each host monitors the other hosts' heartbeats and in the event of a lost heartbeat a failure event will be triggered. The heartbeat threshold for failures is configured for 15 seconds.

Meaning a loss of the heartbeat will not be considered a "failure" for the first 15 seconds. This allows for a minor network disruption or a loss of a heartbeat packet without kicking off recovery actions.

Once a failure is detected VM's are restarted on the Hosts with the most available capacity. Load may not be evenly distributed initially, but the VM's will restart and DRS at that point can begin to load level the environment. It is at this point that some people notice that HA is "not working." In one environment they successfully configured HA, and then had a project that moved their DNS servers to new IP addresses. The ESX hosts themselves were never reconfigured, and during a failure, no VM's were restarted… I cannot state enough times how important it is to ensure your ESX servers have proper DNS settings and are able to resolve the names of all the other hosts in the cluster.

DNS, DNS, DNS

Did I emphasize it enough? Have proper DNS configurations? If not, here is another piece of information. Prior to ESX 3.01 and VC 2.1 (thankfully this now fixed) the DNS names for ESX hosts using HA have a limit of 29 characters. If the FQDN for your host is not 29 characters or less you will have problems with HA. As an example, this FQDN works just fine "esx1.ronoglesby.com" -19 characters. While this one: "prodesx12.internal.ronoglesby.com" -33 characters, does not. Beware; this has bitten more than one ESX admin. Hopefully you are not running this older version of ESX, but one never knows...

Node isolation

One interesting thing that comes from using a heartbeat for this process is that node isolation can cause some big issues. Imagine if someone (inadvertently of course) disconnected all the nics or even just the console nics on one of your ESX hosts. The cluster node at this point stops receiving heartbeats from the other nodes. When this happens, the node first needs to determine if it has been isolated. It does this by pinging the console NIC's default gateway. If no response comes back after 12 seconds it determines it has been isolated and can take action at that point to shut down VM's (this is configurable and discussed next).

Specific Virtual Machine Settings

When VM's are added to the cluster there are some default settings and behaviors you should understand. The first is that within the HA cluster you can configure VM Restart Priorities and their Isolation Response. Let's look at restart priorities first.

Restart Priority

Restart priorities are set for each VM (default is Medium) and are a relative starting priority for VM's after a failure. VM's with higher starting priorities (maybe more critical VM's) are started before medium, then medium priority VM's are started, and so on. Setting these all too high does nothing other than give equal weight to the VM's for restarts, so use judiciously.

Restart priorities are really important when you have configured the admission control to allow for more than your planned amount of failures. Assuming you have only planned for 1 or 2 failures and set the configuration accordingly, then you have 3 hosts fail, the restart priority will ensure that the more important VM's are started before you potentially run out of resources.

Isolation Response

Isolation Response is a setting that dictates what to do with the Virtual Machine when a node detects it has been isolated from the network. The default for this type of event is to power off the virtual machine. The reason behind this default is fairly well conceived. HA recovery is based on the assumption that the host has failed and the VM's need to be started up again. In the case of a node isolation, the VM's may still be running, and the host will still have a lock on the files for the VM's. By initiating a power off of the VM, the lock will be released, and the HA mechanisms can kick in and restart the VM on another host.

One interesting thing about this is to understand the timing involved and the effect on the VM. If a node has been isolated for 12 seconds it declares itself isolated then begins to follow the isolation responses for the VM's. Power Off responses are just that, like hitting the power button on a server and not doing a clean shutdown. At 15 seconds, the other hosts begin their restart steps on the VM's that just powered down. Now, not all applications behave well after a

hard power down, and therefore, they have allowed for another option "Leave power on."

'Leave power on' leaves the VM running on the isolated host. Other hosts trying to restart this VM will not be able to but it may be isolated from the network (or not) and basically just sits on this host hoping everything is ok. Of course the VM with this setting may also have lost its disk connectivity (if using iSCSI over the console nic) and could be dead in the water anyway. Personally, I don't see much of a use for this unless the data inside the VM is extremely susceptible to corruption during a hard power off.

Conclusion

As you can see there are a number of small decisions that need to be made to go into your cluster design. It is imperative that you design your clusters correctly, unlike the logical groupings in VirtualCenter, a cluster is a physical object that requires physical connectivity to the network and storage systems. So decisions about the cluster are more important (and not as easily changed) than the logical Virtual Machine groups in VirtualCenter. Take your time and think out your cluster design, because after hardware selection it is the most important design aspect you will make.