Chapter 6

Effects on Operations

ESX creates a myriad of problems for administrators, specifically problems having to do with the scheduling of various operations around the use of normal tools and other everyday activities such as deployments, VMotion to balance nodes, and backups. Most, if not all, the limitations revolve around issues related to performance gathering and the data stores upon which VMs are placed, whether SCSI, including iSCSI, or non-VMDK files accessed from NFS shared off a NAS or some other system.

The performance-gathering issues dictate which tools to use to gather performance data and how to use the tools that gather this data. A certain level of understanding is required to interpret the results, and this knowledge will assist in balancing the VMs across multiple ESX Servers.

The data store limitations consist of bandwidth issues; each has a limited pipe between the ESX Server and the remote storage and reservation or locking issues. These two issues dictate quite a bit how ESX should be managed. As discussed in Chapter 5, "Storage with ESX," SCSI reservations will occur whenever the metadata of the VMFS is changed and the reservation happens for the whole LUN and not an extent of the LUN. This also dictates the layout of VMFS on each LUN; specifically, a VMFS should take up a whole LUN and not a part of the LUN.

This chapter covers data store performance or bandwidth issues, SCSI-2 reservation issues, and performance-gathering agents, and then finishes with some other issues and a discussion of the impact of Sarbanes-Oxley. Note that some of the solutions discussed within this chapter are utopian and not easy to implement within large-scale ESX environments. These are documented for completeness and to give information that will aid in debugging these common problems.

Data Store Performance or Bandwidth Issues

Because bandwidth is an issue, it is important to make sure that all your data stores have as much bandwidth as possible and to use this bandwidth sparingly for each data store. Normal operational behavior of a VM often includes such things as full disk virus scans, backups, spyware scans, and other items that are extremely disk-intensive activities. Although none of these activities will require any form of locking of the data store on which the VMDK resides, they all take a serious amount of bandwidth to accomplish. The bandwidth requirements for a single VM are not very large compared to an ESX Server with more VMs. Staggering the activities in time will greatly reduce the strain on the storage environment, but remember that staggering across ESX Servers is a good idea as long as different data stores are in use on each ESX Server. For example, it would cause locking issues for VMs that reside on the same LUN but different ESX Servers to be backed up at the same time. This should be avoided. However, virus scans will not cause many issues when done from multiple VMs on the same LUN from multiple ESX Servers, because operations on the VMDK do not cause locks at the LUN level. It is possible that running of disk-intensive tools within a VM could cause results similar to those that occur with SCSI Reservations, but are not reservations. Instead, they are load issues that cause the SAN or NAS to be overworked and therefore present failures similar to SCSI-2 Reservations.

Best Practice for Internal VM Operations
Stagger all disk-intensive operations internal to the VM over time and ESX hosts to reduce strain on the storage network.

SCSI-2 Reservation Issues

With the possibility of drastic failures during crucial operations, we need to understand how we can alleviate the possibility of SCSI Reservation conflicts. We can eliminate these issues by changing our operational behaviors to cover the possibility of failure. Although these practices are generally simple, they are nonetheless fairly difficult to implement unless all the operators and administrators know how to tell whether an operation is occurring, and whether the new operation would cause a SCSI Reservation conflict if it were implemented. This is where monitoring tools make the biggest impact. SCSI Reservation conflicts will be avoided if a

simple rule is followed: Verify that any other operation has first completed on a given file, LUN, or set of LUNs before proceeding with the next operation.

Best Practice
Verify that any other operation has first completed on a given file, LUN, or set of LUNs before proceeding with the next operation.

To verify that any operation is in use, we need to perform all these operations using a similar management interface so that there is only one place to check. The use of Virtual Center or HPSIM with VMM will assist because it gives you one place to check for any operation that could cause a conflict. Verify in your management tool that all operations upon a given LUN or set of LUNs have been completed before proceeding with the next operation. In other words, serialize your actions per LUN or set of LUNs. In addition to checking your management tools, check the state of your backups and whether any current open service console operations have also completed. If a VMDK backup is running, let that take precedence and proceed with the next operation after the backup has completed. To check whether a backup is running, you can quickly look at the VMFS via the MUI or VIC to determine whether there are any REDO files out on the LUNs in question. If there are, either a backup is running or a backup has left a REDO file on the file system, which means the backup most likely failed for some reason. To check to see whether service console operations that could affect a LUN or set of LUNS have completed, judicious use of sudo is recommended. Sudo can log all your operations to a file that you can peruse and then you, as the administrator, can check the process lists for all servers. No user interface combines backups, VMotion, and service console actions.

As an example, let's look at a system of three ESX Servers with five identical LUNs presented to the servers via XP12000 storage. Because each of the three servers shares each LUN we need, we should limit our LUN activity to one operation per LUN at any given time. In this case, we could perform five operations simultaneously as long as those operations were LUN specific. Once LUN boundaries are crossed, the number of simultaneous operations drops. To illustrate the second case, consider a VM with two disk files, one for the C: drive and one for the D: drive. Normally in ESX, we would place the C: and D: drives on separate LUNs to improve performance, among other things. In this case, because the

C: and D: drives live on separate LUNs, manipulation of this VM, say with VMotion, counts as four simultaneous VM operations. This count is due to one operation affecting two LUNs. Therefore, five LUN operations could equate to fewer VM operations. This is the most careful of methods. However, instead of LUN, we can use FILE in many of these suggestions that follow, except where we are changing the metadata.

Using the preceding example as a basis, the suggested operational behaviors are as follows:

- Simplify deployments so that a VM does not span more than one LUN. In this way, operations on a VM are operations on a single LUN.
- Determine whether any operation is happening on the LUN you want to operate on. If your VM spans multiple LUNs, check the full set of LUNs by visiting the management tools in use and making sure that no other operation is happening on the LUN in question.
- Verify that there is no current backup operation happening and that the VM is not in REDO mode.
- Choose one ESX Server as your deployment server. In this way, it is easy to limit deployment operations, imports, or template creations to only one host and therefore one LUN at a time.
- Use a naming convention for VMs that also tells what LUN or LUNs are in use for the VM. This way it is easy to tell what LUN could be affected by VM operation. This is an idealistic solution to a problem, but at least label VMs as spanning LUNs.
- Inside VC or any other management tool, limit access to the administrative operations so that only those who know the process can actually enact an operation. In the case of VC only the administrative users should have any form of administrative privileges. All others should only have VM user or read-only privileges.
- Administrators should only be allowed to power on or off a VM. For reboots required by patch application, schedule each reboot so that there is only one reboot per LUN at any given time. A power-off and power-on are considered separate operations. However, there are more than just SCSI Reservation concerns with this case. For example, if you have 80 VMs across 4 hosts, rebooting all 80 at the same time would create a performance issue, and some of the VMs could fail to boot. The standard boot process for an ESX Server is to boot only the next VM after the VMware Tools are started,

guaranteeing that there is no initial performance issue. The necessary time of the lock for a power-on or -off operation is less than 7 microseconds, so many can be done in the span of a minute. However, this is not recommended because the increase in load on ESX could adversely affect your other VMs. Limiting this is a wise move from a performance viewpoint.

- Use care when scheduling VMDK-level backups. It is best to have one host schedule all backups and to have one script to start backups on all other hosts. In this way, backups can be serialized per LUN. For ESX version 3, this problem is solved by using the VMware Consolidated Backup tool. However, for ESX versions 2.5.x and earlier, use the built-in ESX tool, vmsnap_all, to start a backup, or use the vmsnap_all tool to serialize all activities per VM and LUN. This is discussed further in Chapter 12, "Disaster Recovery and Backup." Using the following pseudo-code may assist with backups where ssh-keygen was employed to make SSH not require a password be entered. By having one host and only one ESX Server run the following, you are guaranteed a serialized action for each backup regardless of the number of LUNs in use. In addition, third-party tools such as ESXRanger can serialize backups:

```
for x in $hosts
  do
    for y in $vms
    do
      ssh $x vmsnap.pl $y &
    done
done
```

This pseudo-code demonstrates for ESX version 2.5.x and earlier releases that we can also change the behavior so more than one backup can occur simultaneously as long each VM in the list of VMs in `$vms` has all its disks on a single separate LUN. If you go with this approach, it is better for performance reasons to have each ESX Server doing backups on a different LUN at any given time. For example, our three machines can each do a backup using a separate LUN. Even so, the activity is still controlled by only *one* host so that there is no mix up or issue with timing. Let the backup process limit and tell you what it is doing. Find tools that will:

- Never start another backup on a LUN while another is still running.
- Signal the administrators that backups have finished either via e-mail, message board, or pager(s). This way there is less to check per operation.

- Limit VMotion (hot migrations), fast migrates, and cold migrations to one per LUN. If you must do a huge number of VMotion migrations at the same time, limit this to one per LUN. With our example there are five LUNs, so there would be the possibility of five simultaneous VMotions, each on its own LUN, at any time. This assumes the VMs do not cross LUN boundaries. VMotion needs to be fast, and the more you attempt to do VMotions at the same time, the slower all will become. There is a chance that the OS inside the VM will start to complain if this time lag is too great. Using VMotion on ten VMs at the same time could be a serious issue for the performance and health of the VM regardless of SCSI Reservations. Make sure the VM has no REDO logs before invoking VMotion.
- Only use the persistent VM disk modes. The other modes create lots of files on the LUNs that will require locking. In ESX version 3, persistent disk modes lead to not being able to perform snapshots and use the consolidated backup tools. These limitations make this item a lower priority from an operational point of view.
- Do not suspend VMs as this also creates a file and therefore requires a SCSI Reservation.
- Do not run vm-support requests unless all other operations have completed.
- Do not use the vdf tool when any other modification operation is being performed.
- Do not rescan storage subsystems unless all other operations have completed.
- Limit use of `vmkmultipath`, `vmkfstools`, and other VMware-specific COS commands until all other operations have completed.
- Create, modify, or delete a VMFS only when all other operations have completed.
- Be sure no third-party agents are accessing your storage subsystem via vdf, or direct access to the `/vmfs` directory. Although vdf does not normally force a reservation, it could experience one if another host, due to a metadata modification, locked the LUN.
- Do not run scripts that modify VMFS ownership, permissions, access times, or modification times from more than one host. Localize such scripts to a single host. It is suggested that you use the deployment server as the host for such scripts.
- Stagger any scripts or agents that affect a LUN so that they run from a management node that can control when actions can occur.

- Stagger the running of disk-intensive tools within a VM such as virus scan. The extra load on your SAN could cause results similar to those that occur with SCSI Reservations but which are not reservations errors but are instead queue-full or unavailable-target errors.
- Use only one file system per LUN.
- Do not mix file systems on the same LUN.
- Do not store a VM's VMX configuration files on shared ext3 partitions on a SAN LUN. In ESX 3.0, you can place a VMX configuration of virtual machines on VMFS volumes (locally or on the SAN).

What this all boils down to is ensuring that any possible operation that could somehow affect a LUN is limited to only one operation per LUN at any given time. The biggest hitters of this are automated power operations, backups, VMotion, and deployments. A little careful monitoring and changes to operational procedures can limit the possibility of SCSI Reservation conflicts and failures to various operations. A case in point follows. One company under review due to constant, debilitating SCSI Reservation conflicts reviewed the list of 23 items and fixed one or two possible items but missed the most critical item. This customer had an automated tool that ran simultaneously on all hosts at the same time to modify the owner and group of every file on every VMFS attached to the host. The resultant metadata updates caused hundreds of SCSI-2 reservations to occur. The solution was to run this script from a single ESX Server for all LUNs. By limiting the run of the script to a single host, all the reservations disappeared, because no two hosts were attempting to manipulate the file systems at the same time, and the single host, in effect, serialized the actions.

Hot and cold migrations of VMs can change the behavior of automatic boot methodologies. Setting a dependency on one VM or a time for a boot to occur deals with a single ESX Server where you can start VMs at boot of ESX, after VMware Tools start in the previous VM, after a certain amount of time, or not at all. This gets much more difficult with more than one ESX Server, so a new method has to be used. Although starting a VM after a certain amount of time is extremely useful, what happens when three VMs start almost simultaneously on the same LUN? Remember we want to limit operations to just one per LUN at any time. We have a few options:

- Stagger the boot or reboot of your ESX Server and ensure that your VMs only start after the previous VMs' VMware Tools start, to ensure that all the disk activity associated with the boot sequence finishes before the next VM boots, thereby helping with boot performance and eliminating conflicts. VM boots are naturally staggered by ESX when it reboots anyway.

- Similar to doing backups, have one ESX Server that controls the boot of all VMs, guaranteeing that you can boot multiple VMs but only one VM per LUN at any time. So, if you have multiple ESX Servers, more than one VM can start at any time on each LUN. In essence, we use the VMware PERL API to gather information about each VM from each ESX Server and correlate the VMs to a LUN and create a list of VMs that can start simultaneously; that is, each VM is to start on a separate LUN. Then we wait a bit of time before starting the next batch of VMs.

All the listed operational changes will limit the amount of SCSI subsystem errors that will be experienced. Although it is possible to implement more than one operation per LUN at any given time, we cannot guarantee success with more than one operation. This depends on the type of operation, the SAN, settings, and most of all, timings for operations.

There are several other considerations, too. Most people want to perform multiple operations simultaneously, and this is possible as long as the operations are on separate LUNs. To increase the number of simultaneous operations, increase the number of LUNs available. Table 6.1 shows the maximum number of operations allowed per number of hosts connected to the LUN for various arrays. The table is broken into categories of risk based on the number of operations per LUN and the SCSI conflict retry count. In this table, gaps exist between the number of hosts per LUN and the number of operations per LUN; assume that if you are above the listed number, you are in the next-highest category.

Table 6.1

Risk Associated with Number of Operations per LUN

Array Type	# of Host(s)	Low Risk (0% to 10% failure)	Medium Risk (30% to 60% failure)	High Risk (> 60% failure)	SCSI Conflict Retry Count
Entry level - MSA	1	4	8	10	20
	2	2	4	5	20
	4	2	3	4	20
	8	1	2	3	20
Enterprise – EVA, Symmetrics	1	8	12	16	8
	2	2	4	6	8
	4	2	3	4	8
	8	1	2	3	8

Array Type	# of Host(s)	Low Risk (0% to 10% failure)	Medium Risk (30% to 60% failure)	High Risk (> 60% failure)	SCSI Conflict Retry Count
Hitachi/HDS	1	6	10	12	20
	2	2	4	6	20
	4	2	3	4	20
	8	1	2	3	20

Note that no more than eight hosts should be attached to any one given LUN at a time. Also note that as firmware is modified, these values can change to be higher or lower.

Performance-Gathering Agents

Performance and other monitoring is an important issue from an operational point of view. Many customers monitor the health of their hardware and servers by monitoring hardware and performance agents. Although hardware agents monitor the health of the ESX Server, they should not monitor the health of a VM, because the virtual hardware is truly dependent on the physical hardware. In addition, most agents are talking to specific chips, and these do not exist inside a VM. So using hardware agents will often slow down your VM.

Best Practice for Hardware Agents
Do *not* install hardware agents into a VM; they will cause noticeable performance issues.

Because these agents will adversely affect performance, measuring performance now is a very important tool for the Virtual Infrastructure and will tell you when to invest in a new ESX Server and to how to balance load among the ESX Servers. Although there are automated ways to balance the load among ESX Servers (they are covered in Chapter 11, "Dynamic Resource Load Balancing"), most if not all balancing of VM load across hosts is performed by hand, because there are more than just a few markers to review when moving VMs from host to host.

The first item to understand is that the addition of a VM to a host will impact the performance of the ESX Server; sometimes in small ways, and sometimes in

other ways that are more noticeable. The second item to understand is that the performance tools that run within a VM depend on real clock cycles to determine the performance of the VM and that a VM is not always given full clock cycles. Because there are often more VMs than CPUs or cores, a VM will share a CPU with others, and as more VMs are added the slice of time the VM gets to run on a CPU is reduced even further. Therefore, there is a greater time lag between each usage of the CPU and thus a longer CPU cycle. Because performance tools use the CPU cycle to measure performance and to keep time, the data received is relatively inaccurate. However, the experimental descheduler VMware tool can be used to counteract this effect. When the system is loaded to the desired level, a set of baseline data should be discovered.

Once a set of baseline data is available, the internal to the VM performance tools can determine whether a change in performance has occurred, but it cannot give you raw numbers, just a ratio of change from the baseline. For example, if the baseline for CPU utilization is roughly 20% measured from within the VM and suddenly shows 40%, we know that there was a 2x change from the original value. The original value is not really 20%, but some other number. However, even though this shows 2x more CPU utilization for the VM, it does not imply a 2x change to the actual server utilization. Therefore, other tools need to be used to gain performance data for a VM that do not run from within the VM. Although useful for baselines, they are not useful overall. In this case, VC, a third-party VM manager, ESXCharter, and the use of `esxtop` from the command line are the tools to use. These all measure the VM and ESX Server performance from outside the VM, to gain a clearer picture of the entire server. The key item to realize is that when there is a sustained over 80% utilization of an ESX Server as measured by VC or one of the tools, a new ESX Server is warranted and the load on the ESX Server needs to be rebalanced.

Balancing of ESX Servers can happen daily or even periodically during the day by using the VMotion technology to migrate running VMs from host to host with zero downtime. Although this can be dynamic (covered in Chapter 11, "Dynamic Resource Load Balancing"), using VMotion by hand can give a better view of the system and the ability to rebalance as necessary. For example, if an ESX Server's CPU utilization goes to 95%, the VM that is the culprit needs to be found using one of the tools; once found, the VM can be moved to an unused or lightly used ESX Server using VMotion. If this movement becomes a normal behavior, it might be best to place the VM on a lesser-used machine permanently. This is often the major reason an N+1 host configuration is recommended.

Another item that can increase CPU utilization is the deployment of VMs. Deployment is discussed in detail in a later chapter, but the recommendation is to create a deployment server that can see all LUNs. This server would be responsible for deploying any new VM, which allows the VM to be tested on the deployment server until it is ready to be migrated to a true production server using VMotion.

For example, a customer desired to measure the performance of all VMs to determine how loaded the ESX Server could become with their current networking configuration. To do so, we explained the CPU cycle issues and developed a plan of action. We employed three tools in this example, VC, `vmkusage` (now rolled into VC 2.0), and `esxtop` running from the service console. We found that the granularity of VC was less than `vmkusage`, which was less than `esxtop`. For performance-problem resolution, `esxtop` is the best tool to use, but it spits out reams of data for later graphing. VC averages things over 30-minute or larger increments unless the graph is exported then the raw data can be manipulated in Excel. `vmkusage` averages all data over 5-minute distributions. `esxtop` uses real and not averaged data. The plan was to measure performance using each tool as each VM was running its application. Performance of ESX truly depends on the application within each VM. It is extremely important to realize this, and when discussing performance issues to not localize to just a single VM, but to look at the host as a whole. This is why VMware generally does not allow performance numbers to be published, because without a common set of applications running within each VM, there is no way to compare all the apples to oranges. It is best to do your own analysis using your applications, because one company's virtualized application suite has nothing to do with another companies, and therefore will have major implications on performance.

Other Operational Issues

Because ESX makes extensive use of memory, there are operational concerns regarding the use of memory, too. The main issue with memory is to prevent the overcommitment of memory during runtime of the VMs. The runtime covers the memory actually used, and not always what is allocated. A VM may have 16GB of memory allocated to it. If this much memory is allocated to all the VMs, on a 32GB ESX Server, only one VM could be created unless the memory is overcommitted as the OS takes some memory. If the goal is to run ten VMs, there is a memory requirement of 160GB, which is quite a bit over the 64GB server memory limit inherent in ESX. Which means that if all the 16GB of memory is actually used by a VM, the ESX Server will need to start swapping (or paging) memory out

in large chunks to accommodate the running of another VM. However, if in reality only 1GB of each VM is used, there is only 10GB of the available 32GB of memory in use at any time, allowing more VMs to be created and used without swapping memory, even though there is potential for up to 160GB of memory to be used. In this case, it is best to assign memory sparingly and to only give a VM what it needs to run. This way memory management will allow a denser population of VMs to run. Note it is not possible to create a VM that will exceed the physical memory of the machine.

Consider the following thought: with ESX we are now back in time to the realm of limited resources. There are no longer gobs of memory and disk available for any particular machine, but a realm where memory and disk can be vast; but as more VMs are added, more resources are used. The goal is now to preserve memory. As an example, consider programming the old Commodore 64, where no more than 360K would fit on a single floppy; to go past this, more than one floppy had to be used. Everyone programmed to the 360K limit of the floppy so that code would fit on a single disk. Once another floppy was in use, the applications usage went downhill, performance suffered, and wait time increased. With ESX, we are back in this realm where we need to be cognizant of the limitations of the host, which is trying to do much, much more with less than ever before.

Best Practice for ESX
The mindset for ESX is to only give out the necessary resources to each VM rather than give out all the resources.

All VMs affect the resource limits of the host and therefore resource management becomes a huge issue (as covered in another chapter). Note, however, that changes to the way resources are used, assigned, and managed can inadvertently affect all VMs on a host or in a farm.

Limiting memory assignment to VMs can allow more VMs to run in a single ESX host without impacting memory or performance limits.

Sarbanes-Oxley

Sarbanes-Oxley can also impact performance of an ESX Server, because the gathering up of log files at incorrect times will inadvertently cause a load on the

system. With the advent of ESX version 3, it is now possible to store VM log files with the VMDK, and access to the file system holding these logs has been throttled so that copies to and from do not adversely impact performance. What logs are necessary? Those logs that tell who did what and when are ultimately required, but many system administrators are streaming *all* logs to tape for later retrieval. For ESX, this will be all the logs from each VM running on the host and all the VM log files and service console log files in `/var/log`. Once more, a single host is best to use to gather all this information so that no one LUN or server is overloaded. Because the ESX service console is Linux, the network features of syslog can be used to create a logging server so that all data is logged to one host from where it can be conveniently archived. The key is to make use of syslog to send the logs to the remote host and the local host. To do that, modify the `/etc/syslog` file as follows: Duplicate each uncommented line and add to the end instead of the local filename the `@ipaddress` option to force log files to not only be stored locally but remotely to the IP address of the syslog server.

Conclusion

By paying careful attention to operational issues, it is possible to successfully manage ESX and remove some of the most common issues related to poor operational use of ESX. ESX is designed to be centrally managed and care should be taken to do so. It is also important to realize that each implementation of ESX has different operational concerns.