Best Practices for Running VMware vSphere® on Network-Attached Storage (NAS)

TECHNICAL MARKETING DOCUMENTATION
V 2.0/JANUARY 2013

vmware®

Table of Contents

Introduction

The significant presence of network-attached storage (NAS) in the datacenter today has enabled many people to deploy virtualization environments with NAS-based shared storage resources. For clarity, both Network File Systems (NFS) and NAS refer to the same type of storage protocol and are used interchangeably throughout this paper.

The capabilities of VMware vSphere® on NFS are very similar to those of vSphere on block-based storage. VMware offers support for all vSphere features and functions on NFS, as it does for vSphere on block storage. Running vSphere on NFS is a viable option for many virtualization deployments, because it offers strong performance and stability when configured correctly.

This paper provides an overview of the considerations and best practices for deployment of vSphere on NFS-based storage.

Background

VMware has supported IP-based storage for a number of years now. NFS storage was introduced as a storage resource that can be shared across a cluster of VMware vSphere ESXi™ hosts. The addition of new choices has led to the question, "What is the best storage protocol choice on which to deploy a virtualization project?" There seems to be no single correct answer.

The considerations for this choice tend to hinge on the issue of cost, performance, availability and ease of manageability. However, an additional factor might be the legacy environment and the storage administrator's familiarity with one protocol versus the other based on what is already installed.

A better question than which storage protocol to deploy virtualization on is the question of which virtualization solution enables one to leverage multiple storage protocols for their virtualization environment. Another consideration is which solution provides the best way to move virtual machines from one storage pool to another, regardless of what storage protocol it uses, without downtime or application disruption. After those questions are considered, the clear answer is vSphere.

NFS provides suitable I/O performance for many virtual environments. The comparison of storage protocols has been the subject of many papers and projects. One such paper that compares and contrasts storage protocols in a vSphere environment is posted on the VMware Technical Resource Center as *Storage Protocol Comparison*.

The general conclusion reached in a performance comparison paper, *Comparison of Storage Protocol Performance in VMware vSphere 4*, is that for most workloads, the performance is similar between many protocols, but that there is a slight increase in ESXi-host CPU overhead per transaction for NFS. For most virtualization environments, the end user cannot detect the performance delta when comparing one virtual machine running on NFS-based storage to another virtual machine on a different storage protocol.

The more important consideration that often leads customers to choose NFS storage for their virtualization environment is the ease of provisioning and maintaining NFS shared storage pools. NFS storage is often less costly than other storage protocols to set up and maintain. For this reason, many businesses that are deploying virtualization choose NFS. It is also the choice when considering deployment of virtual desktop infrastructures.

NFS Datastore Provisioning Steps

Before a vSphere host can utilize NFS storage, the following configuration steps must be taken:

1. Create a new VMkernel port group for IP storage on an already existing virtual switch (vSwitch) or on a new vSwitch when it is configured. The vSwitch can be a vSphere Standard Switch (VSS) or a vSphere Distributed Switch (VDS).
2. Ensure that the NFS client on the vSphere host(s) is enabled.
3. Ensure that the NFS storage is configured to export a mount point accessible to the vSphere hosts on a trusted network.

Regarding item 1, the user must create a new VMkernel port group to configure the vSwitch for IP storage access. The user must also populate the network access information.

Regarding item 2, the user must open the firewall port for the NFS client on all hosts to configure the NFS client on the vSphere host. This is automatically opened in the more recent releases of vSphere. To check if the port is open, go to the ESXi host manage tab in VMware® vCenter™, select **Settings** and then select **Security Profile.** Click **Edit** and scroll down to the NFS client, shown in Figure 1.

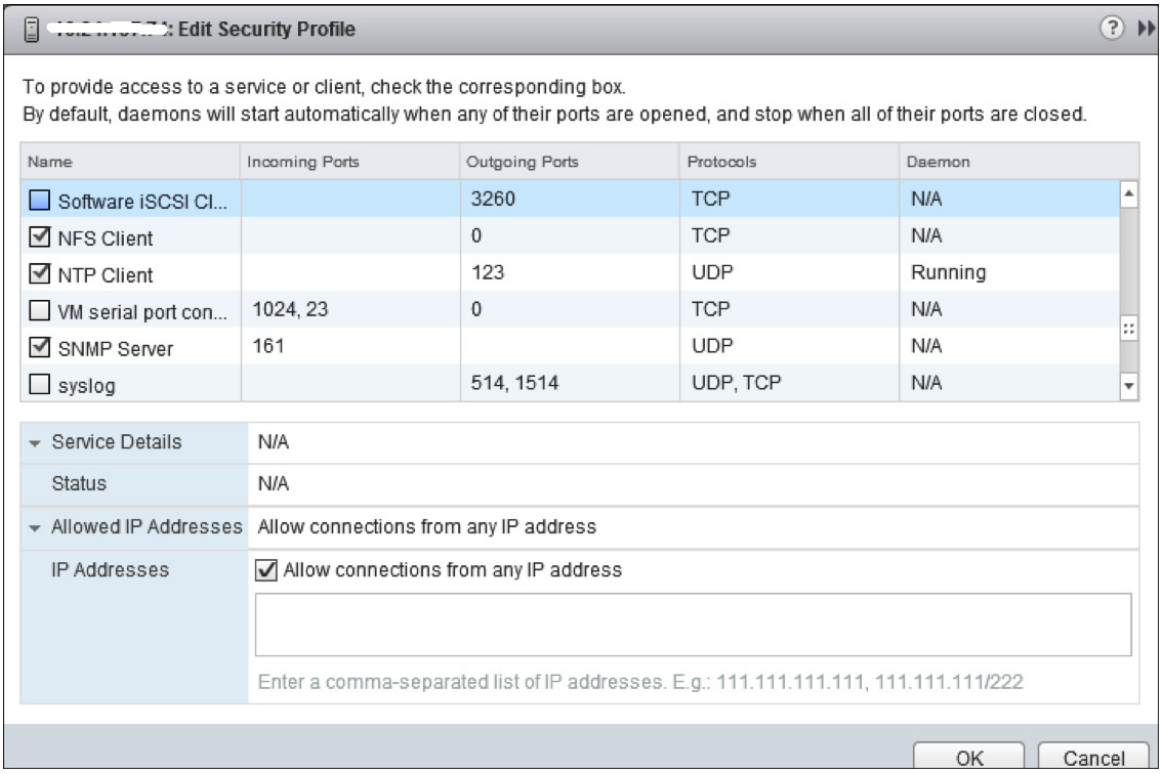

Figure 1. ESXi Host Security Profile—NFS Client

When addressing item 3, the user must ensure that they are using NFS over TCP rather than UDP. For example, if UDP is used, vSphere provides a descriptive message indicating that this is not possible. Similarly, when mounting a share using a version of NFS that is not version 3, vSphere also provides a descriptive message stating that this version of NFS is not supported. VMware currently supports NFS version 3 over TCP only.

You must also ensure that the vSphere host has root access to the datastore. This is typically done on NFS servers using the **no_root_squash** option in the /etc/exports file. Different storage vendors have different methods of enabling this functionality. If you do not grant root access, you might still be able to mount the NFS datastore on the vSphere host. However, you will not be able to create any virtual machines on the datastore. It will fail with an "unable to access file" error message. For more details on NFS storage options and setup, consult the best practices for VMware provided by your storage vendor.

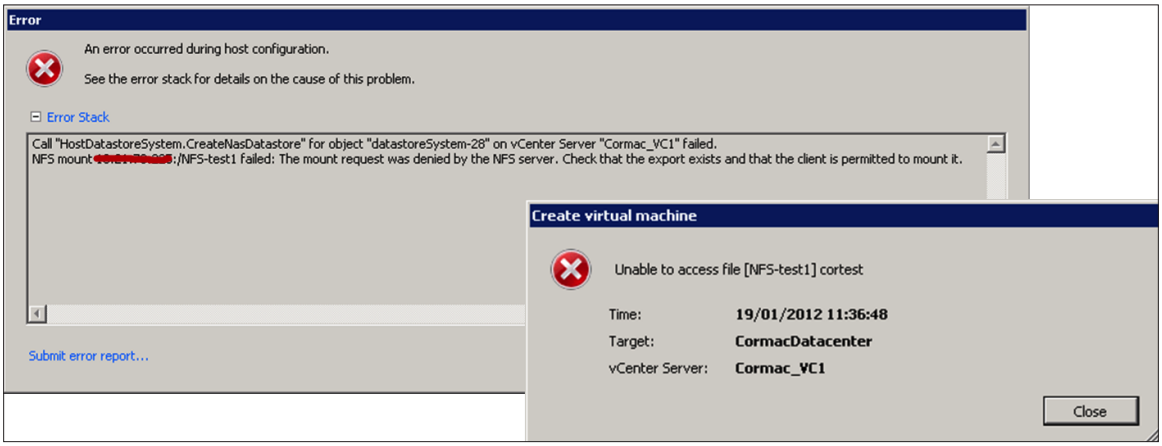

Figure 2. Errors When NFS Storage Is Not Correctly Exported to ESXi Host

With these items addressed, an NFS datastore can now be added to the ESXi host.

Networking Settings

At the time of this writing, VMware supports only NFS version 3 over TCP/IP. Therefore there are still some limits to the multipathing and load-balancing approaches that can be taken. Although there are two channels opened when an NFS datastore is mounted onto a vSphere host (one channel for control and the other for data), there is still only a single TCP connection for I/O.

It is important to understand that there is only one active connection between the vSphere host and a single NFS datastore target. This means that although there might be alternate connections available for failover, the bandwidth for a single datastore and the underlying storage is limited to what a single connection can provide.

To leverage more available bandwidth, a vSphere host might have multiple connections to the storage targets. One would be required to configure multiple datastores, with each datastore using separate connections between the server and the storage. This is where one often runs into the distinction between load balancing and load sharing. Load sharing is the configuration of traffic spread across two or more datastores configured on separate connections between the vSphere host and the storage array.

From a networking perspective, NFS requires multiple subnets to use multiple uplinks. In a situation where an EtherChannel is properly utilized on a single subnet, multiple subnets are not required to utilize multiple uplinks. However, even with EtherChannel and multiple storage targets, planning is still required to utilize more than one uplink.

Throughput Options

Let's look at some of the options available and how you might be able to improve performance, keeping in mind that you have a single connection between host and storage.

1. 10GbE. This is an obvious option to begin with. If you can provide a larger pipe, the likelihood is that you will achieve greater throughput. Of course, if there is not enough I/O to fill a 1GbE connection, then a larger connection isn't going to help you. But let's assume that there are enough virtual machines and enough datastores for 10GbE to be beneficial.

2. Jumbo frames. Although this feature can deliver additional throughput by increasing the size of the payload in each frame from a default MTU of 1,500 to an MTU of 9,000, great care and consideration must be used if you decide to implement it. All devices sitting in the I/O path must be able to implement jumbo frames for this to provide the full benefits (array controller, physical switches, network interface cards and VMkernel ports). For example, if the MTU is not correctly set on the switches, the datastores might mount but I/O will fail. A common issue with jumbo-frame configurations is that the MTU value on the switch isn't set correctly. In most cases, this must be higher than the hosts and storage that are typically set to 9,000. Switches must be set higher, for example, to 9,198 or 9,216, to account for IP overhead. Refer to switch-vendor documentation, as well as storage-vendor documentation, before attempting to configure jumbo frames.

3. Load sharing using IP hash. One of the configuration options mentioned above is to use multiple connections from the ESXi host to the storage targets. This configuration might be implemented with a single VMkernel port, but the host must have two or more uplinks connected to the storage. When a user employs the "Route based on IP hash" algorithm, different NFS datastore traffic might route through different uplinks. There is an important point to make about using this algorithm, however. Since the hashing algorithm uses both the source and destination IP addresses to determine which uplink to use for traffic, there is no guarantee that an evenly distributed flow of traffic across all uplinks is achieved. If you have a very small subnet with only a few targets, the hashing algorithm might still send all traffic via a single uplink. KB article 1007371 explains how the hash is calculated. If this is a load-sharing method that you wish to leverage, you should read the KB article to determine how best to achieve a load-balance distribution across uplinks in your environment.

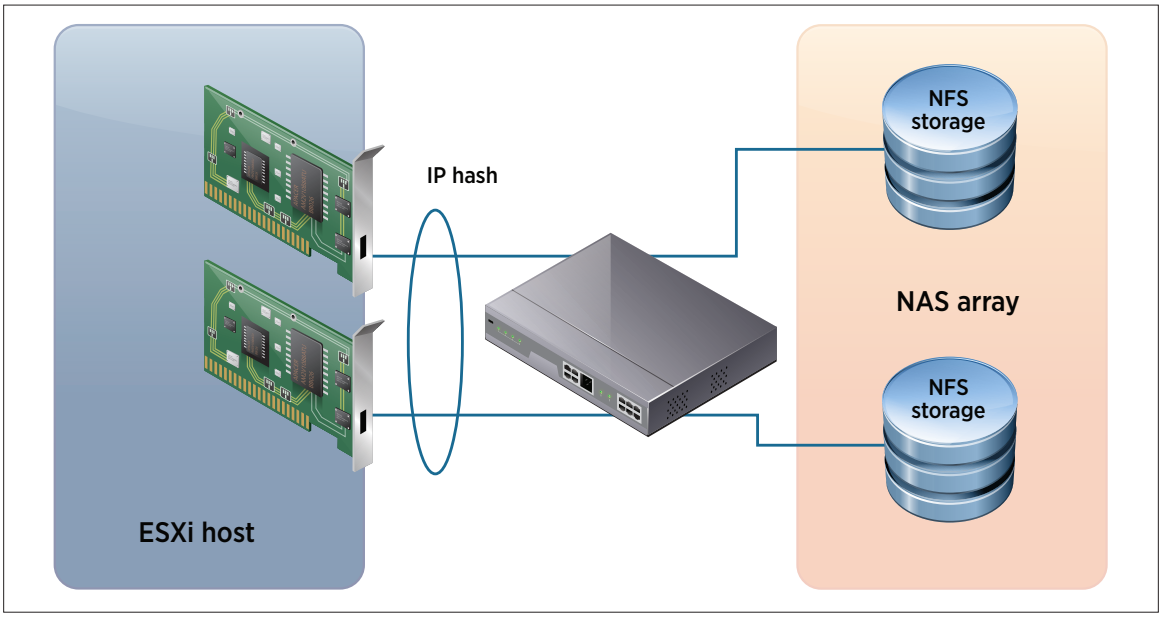

Figure 3. Load Sharing Using IP Hash

4. Load sharing using subnets. In this configuration, two network interface cards are required on the ESXi host. One network interface card is used for mounting a datastore via one array controller and the other network interface card is used for mounting a datastore via the other controller. Therefore, two VMkernel ports are also required. These connections must be on separate subnets. Of course, this is a very static setup, and if the datastores presented via one controller are much busier than the datastores presented via the other controller, then throughput might not be maximized. Knowing which of your virtual machines are busy, and balancing them appropriately across datastores, can assist here. The more connections that can be made between the ESXi host and the storage-array controllers, the more options you have for load balancing/load sharing.

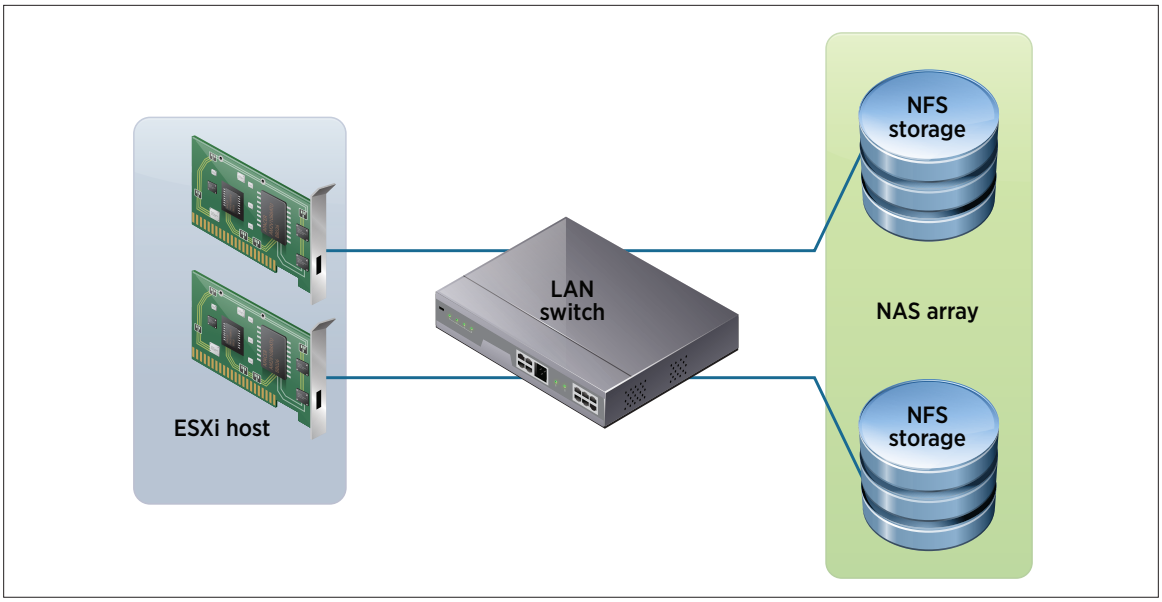

Figure 4. Load Sharing Using Subnets

To implement this, one must have access to each datastore using separate connections between the host and the storage, in other words, NFS shares presented on different IP addresses and different subnets, as shown in Figure 4.

5. Link aggregation. Another possible way to increase throughput is via the use of link aggregation. This isn't always guaranteed to deliver additional performance, but link aggregation will be discussed in the context of availability later on in this paper.

Minimizing Latency

Because NFS on VMware uses TCP/IP to transfer I/O, latency can be a concern. To decrease latency, one should always try to minimize the number of hops between the storage and the ESXi host.

Ideally, one would not route between the ESXi host and the storage array if they were both on the same subnet. In fact, prior to ESXi 5.0 U1, routing between the ESXi host and the storage array was not supported. Some restrictions were lifted to support the routing of NFS traffic in vSphere release 5.0 U1. There are still quite a number of restrictions with routing NFS and the release notes should be examined in detail to determine whether it is suitable in your environment. However, although routing NFS traffic is supported in ESXi 5.0 U1 and later, it is recommended not to route the traffic to minimize latency.

Availability

To achieve high availability, the LAN on which the NFS traffic will run must be designed with availability, downtime avoidance, isolation and no single point of failure (SPOF) in mind. Multiple administrators must be involved in designing for high availability. These are the virtualization administrator and the network administrator. This section outlines these steps and investigates several options, which can be utilized to make your NFS datastores highly available.

1. Network interface card teaming at the host level. For additional availability, two or more network interface cards can be combined together to form a team. In the event of one network adaptor failing, connections will failover to another network interface card on the team.

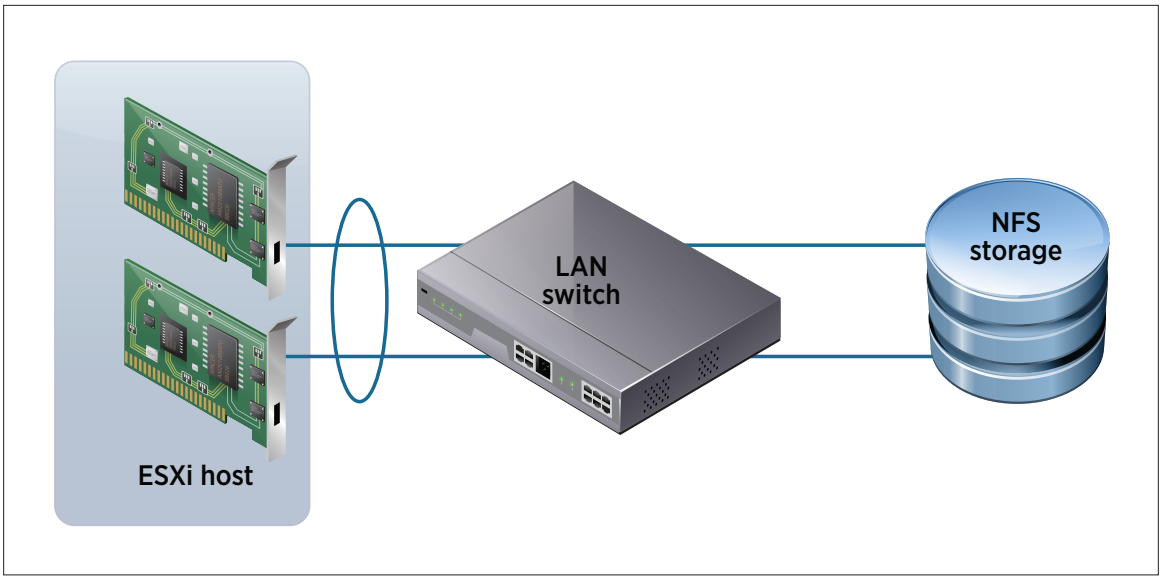

Figure 5. Network Interface Card Teaming for Availability

A common practice is to set the network interface card teaming failback option to **No**. The reason for this is to avoid a flapping network interface card if there is some intermittent issue on the network.

There are a number of load-balancing configuration options available for the team. The default is **Route based on originating virtual port.** This means that a hashing algorithm is applied to the originating virtual port id (either a virtual machine or a VMkernel connection), and based on the result, a particular network interface card will be chosen. If there is only one VMkernel port for NFS traffic on the ESXi host, then this algorithm will use the same uplink. If there are multiple VMkernel ports on different subnets, the uplink used depends on the route to the target IP address. Therefore, there is no guarantee that connections will balance successfully with this policy. It is possible that all NFS connections might be placed on the same uplink, even though they are using unique VMkernel ports on the switch.

Another option is to choose **Route based on IP hash**, but this has some restrictions.

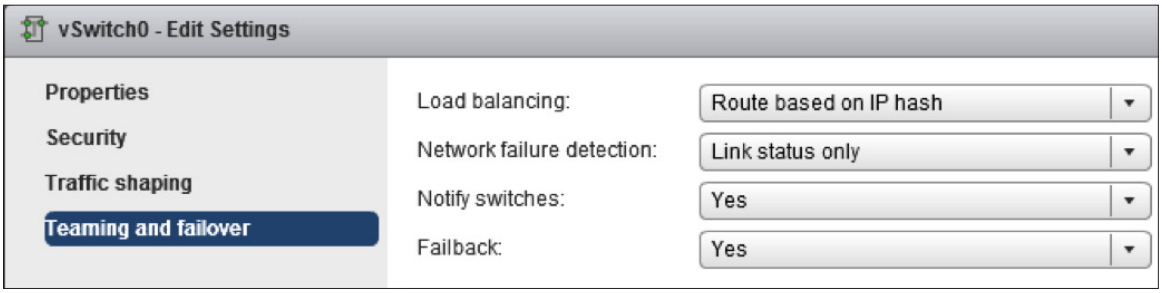

Figure 6. Network Interface Card Teaming Configuration—Route Based on IP Hash

This policy can be used only if all physical switch ports connected to the active uplinks are placed in the same team or channel for link aggregation. This is a physical switch setting. Conversely, all port groups using the same set of uplinks should have the IP hash load-balancing policy set. We will talk about link aggregation in the next section. Further details regarding the requirements for link aggregation and "Route based on IP hash" can be found in the KB article 1001938. It is important to follow these requirements because failure to do so can disrupt network connectivity.

The design in Figure 5 is somewhat simplified. There are still issues with the physical LAN switch being an SPOF. To avoid this, a common design is to use network interface card teaming in a configuration that has two physical switches. With this configuration, there are four network interface cards in the ESXi host, and these are configured in two pairs with IP hash failover. Each pair is configured as a team at their respective LAN switch.

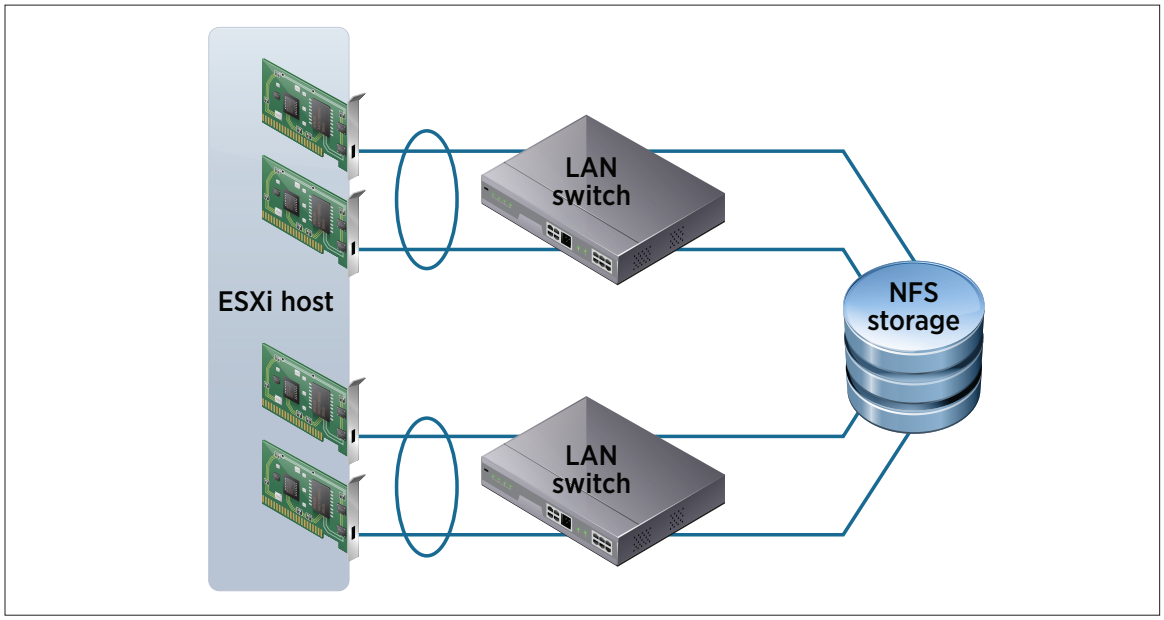

Figure 7. Network Interface Card Teaming with Redundant Switch for Availability

Another option available with VMware vSphere Enterprise Plus Edition™ licensing is the **Route based on physical NIC load** load-balancing policy, otherwise known as **Load-Based Teaming** (LBT). This policy was introduced in vSphere 4.1 and is available only on the VDS. This means that you cannot use LBT if you are not licensed for VMware vSphere Enterprise Plus Edition. The operation of this policy implies that when an uplink reaches 75 percent utilization for 30 seconds, LBT tries to move workloads to other, nonsaturated uplinks.

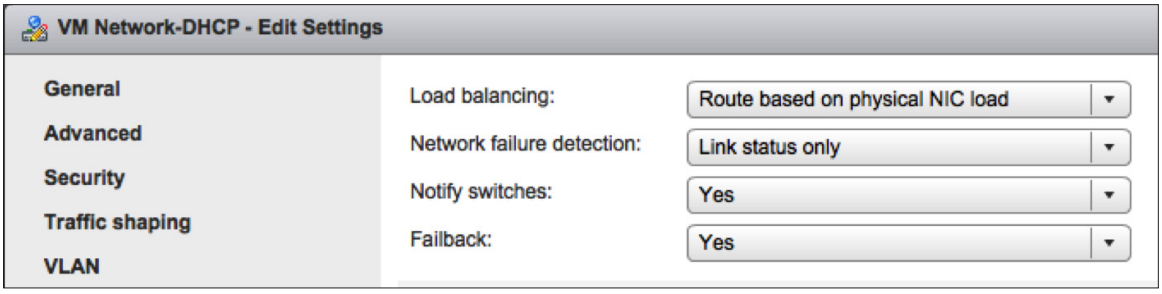

Figure 8. Network Interface Card Teaming Configuration—Route Based on Physical NIC Load

2. Link Aggregation Control Protocol (LACP) at the array level. LACP enables you to combine multiple physical interfaces into a single logical interface. Now it is debatable whether this can improve throughput or performance because we are still limited to a single connection with NFS version 3, but it does enable protection against path failures. Many NFS-array vendors support this feature at the storage controller port level. Most storage vendors will support some form of link aggregation, although not all configurations might conform to the generally accepted IEEE 802.3ad standard. The best option would be to check with your storage vendor. One of the features of LACP is its ability to respond to events on the network and detect which ports should be part of the logical interface. Many failover algorithms respond only to link down events. This means that a switch might inform the array that an alternate path must be chosen, rather than having the array rely on a port failure.

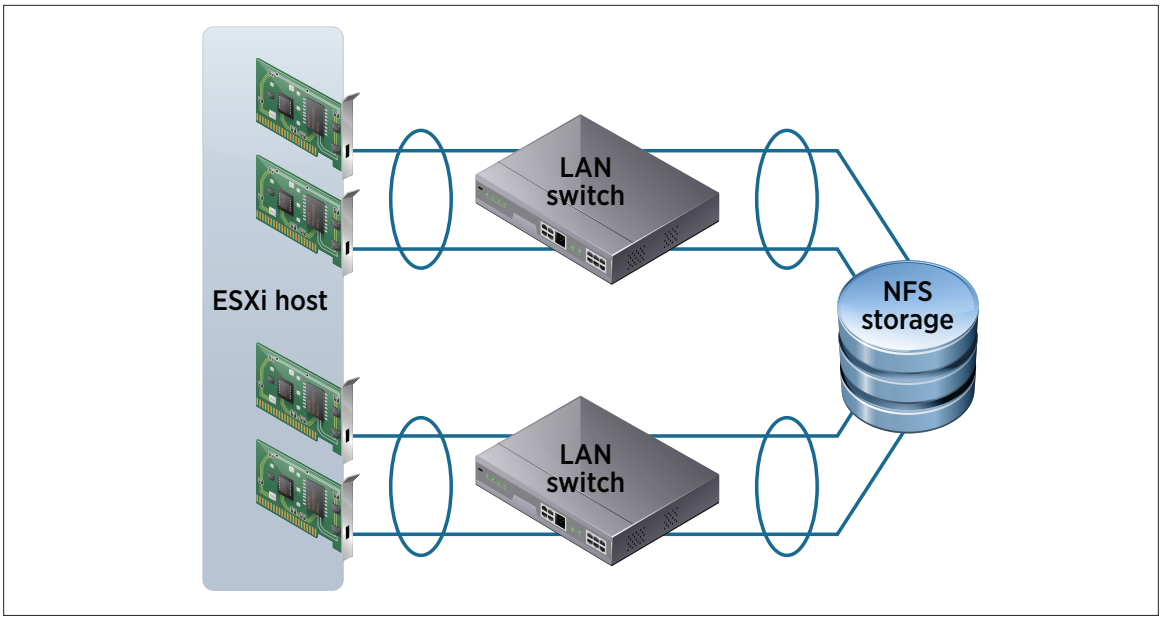

Figure 9. Network Interface Card Teaming with Redundant Switch and Link Aggregation for Availability

With LACP, you are required to use "Route based on IP hash" load balancing. There are some technical requirements for using the IP hash load-balancing policy over the default "Route based on originating virtual port" load-balancing policy. The VMware KB article 1004048 has a good overview of some sample configurations of link aggregation with network interface card teaming and LACP. In addition to the use of IP hash, the other requirement is that you must ensure that all uplinks in the team are in an active state. This is because the physical switch sends data to the virtual machines via any of the physical network interface cards. If one of those network interface cards is not active, the virtual machine might never receive the packet because the virtual switch drops it.

3. Link Aggregation Control Protocol (LACP) at the ESXi host level. This is a new feature that VMware introduced in vSphere 5.1 and requires a 5.1 version of the VDS. This feature might provide additional availability in the same way as it might provide additional availability for arrays, in so far as a failover to an alternate network interface card can now occur based on feedback from the physical switch as opposed to just relying on a link failure event. At the time of this writing, there are some restrictions with the LACP support at the ESXi host side. These include the ability to have only one LACP group supported on a VDS.

As you can see, there are quite a number of options available in terms of availability. It might be easier to implement some form of network interface card teaming, but other more complex options are available with LACP. Considering the number of restrictions related to the use of LACP, both at the array and at the host level, much planning is required up front before implementation. Refer to vendor documentation for guidance.

Miscellaneous Network Features

Flow Control

This section highlights a few other recommendations from our storage partners. The first of these is flow control. Flow control manages the rate of data flow between the ESXi host and storage array. Depending on the interconnect (1GbE or 10GbE), some array vendors recommend turning flow control off and enabling management of congestion higher up the stack. One should always refer to the storage-array vendor's best practices for guidelines.

Spanning Tree Protocol

The second recommendation relates to the use of switch ports when Spanning Tree Protocol (STP) is used in an environment. STP ensures that there are no network loops in a bridged network. This is accomplished by disabling network links and ensuring that there is only a single active path between any two network nodes. If there are loops, it will have a severe performance impact on your network with unnecessary forwarding of packets, eventually leading to a saturated network.

A switch port can exist in various states under STP while the algorithms determine if there are any loops in the network. For example, switch port states can be blocked, listening or forward. Transitioning between the various states can take some time and this can impact applications that are network dependent. Some storage array vendors recommend setting those switch ports to which their array ports connect as either RSTP edge ports or Cisco PortFast. This means that the switch ports will immediately change their forwarding state to active, to enable the port to send and receive data. Refer to your storage-array best practices for advice on this setting, to determine if it is appropriate for your storage array.

In addition to using this setting on switch ports, it is also the recommended setting for the vSphere host's connections in the same environment.

Security Considerations

Private Network

vSphere implementation of NFS supports NFS version 3 in TCP. Storage traffic is transmitted in an unencrypted format across the LAN. Therefore, it is considered best practice to use NFS storage on trusted networks only and to isolate the traffic on separate physical switches or to leverage a private VLAN. All NAS-array vendors agree that it is good practice to isolate NFS traffic for security reasons. This would mean isolating the NFS traffic on its own separate physical switches or leveraging a dedicated VLAN (IEEE 802.1Q).

Root Privileges

Another security concern is that the ESXi host mounts the NFS datastores using root privileges. This is done by having a **no_root_squash** option in the /etc/exports file or by using a vendor-specific root-enablement option. Because this is NFS version 3, none of the security features implemented in later versions of NFS are available. This raises some concerns about hackers getting access to the NFS server. To address the concern, it is a best practice to use either a dedicated LAN or a dedicated VLAN to provide protection and isolation.

Mount and Read/Write Privileges

Many NFS servers/arrays have some built-in security, which enables them to control the IP addresses that can mount its NFS exports. It is considered a best practice to use this feature to determine which ESXi hosts can mount and have read/write access to the volumes that are being exported. This prevents unapproved hosts from mounting the NFS datastores.

Interoperability Considerations

This section discusses features that are in some way related to storage, and NFS storage in particular. Most of the interoperability features are tried and tested with NFS, but areas that might be cause for additional consideration are highlighted.

Storage I/O Control

Storage I/O Control (SIOC) prevents a single virtual machine residing on one vSphere host from consuming more than its fair share of bandwidth on a datastore that it shares with other virtual machines residing on other vSphere hosts.

Historically, the disk shares feature can be set up on a per–vSphere host basis. This works well for all virtual machines residing on the same vSphere host sharing the same datastore built on a local disk. However, this cannot be used as a fairness mechanism for virtual machines from different vSphere hosts sharing the same datastore. This is what SIOC does. SIOC modifies the I/O queues on various vSphere hosts to ensure that virtual machines with a higher priority get more queue entries than those virtual machines with a lower priority, enabling these higher-priority virtual machines to send more I/O than their lower-priority counterparts.

SIOC is a congestion-driven feature. When latency remains below a specific latency value, SIOC is dormant. It is triggered only when the latency value on the datastore rises above a predefined threshold.

SIOC was first introduced for NFS datastores with vSphere 5.0. SIOC is recommended if you have a group of virtual machines sharing the same datastore spread across multiple vSphere hosts and you want to prevent the impact of a single virtual machine's I/O on the I/O (and thus performance) of other virtual machines. With SIOC you can set shares to reflect the priority of virtual machines, but you can also implement an IOPS limit per virtual machine. This means that you can limit the impact, in number of IOPS, that a single virtual machine can have on a shared datastore.

SIOC is available in the VMware vSphere Enterprise Plus Edition. More details on SIOC can be found in the *Storage IO Control Technical Overview & Considerations for Deployment* white paper.

Network I/O Control

The Network I/O Control (NIOC) feature ensures that when the same network interface cards are used for multiple traffic types, NFS traffic is not impacted by other traffic types. It works by setting priority and bandwidth using priority tags in TCP/IP packets. With 10GbE networks, this feature can be very useful, because there is one pipe that is shared with multiple other traffic types. With 1GbE networks, you have probably dedicated the pipe solely to NFS traffic. NIOC is congestion driven. If there is no congestion, any traffic type can consume as much bandwidth as it requires. NIOC takes effect only when there are different traffic types competing for bandwidth, and the performance of one traffic type is likely to be impacted.

Whereas SIOC assists in dealing with the noisy-neighbor problem from a datastore-sharing perspective, NIOC assists in dealing with the noisy-neighbor problem from a network perspective.

Using NIOC, one can also set the priority levels of different virtual machine traffic. If certain virtual machine traffic is important to you, these virtual machines can be grouped into one virtual machine port group whereas lower-priority virtual machines can be placed into another virtual machine port group. NIOC can now be used to prioritize virtual machine traffic and ensure that the high-priority virtual machines get more bandwidth when there is competition for bandwidth on the pipe.

SIOC and NIOC can coexist and in fact complement one another.

NIOC is available in the VMware vSphere Enterprise Plus Edition. More details on NIOC can be found in the *Network I/O Control Best Practices* white paper.

vSphere Storage DRS

VMware vSphere Storage DRS™, introduced with vSphere 5.0, fully supports NFS datastores. When you enable vSphere Storage DRS on a datastore cluster (group of datastores), it automatically configures balancing based on space usage. The threshold is set to 80 percent but can be modified. This means that if space on a particular datastore is utilized at 80 percent or more, vSphere Storage DRS will try to move virtual machines to other datastores in the datastore cluster using VMware vSphere Storage vMotion® to bring this usage value back down below 80 percent. The usage statistics of the datastores are checked on an ongoing basis.

If the cluster is set to the **automatic** mode of operation, vSphere Storage DRS uses Storage vMotion to automatically migrate virtual machines to other datastores in the datastore cluster if the threshold is exceeded. If the cluster is set to **manual**, the administrator is given a set of recommendations to apply. vSphere Storage DRS will provide the best recommendations to balance the space usage of the datastores. After you apply the recommendations, Storage vMotion, as seen before, moves one or more virtual machines between datastores in the same datastore cluster.

Another feature of vSphere Storage DRS is that it can balance virtual machines across datastores in the datastore cluster based on I/O metrics, and specifically latency.

vSphere Storage DRS uses SIOC to evaluate datastore capabilities and capture latency information regarding all the datastores in the datastore cluster. As mentioned earlier, the purpose of SIOC is to ensure that no single virtual machine uses all the bandwidth of a particular datastore. It achieves this by modifying the queue depth for the datastores on each vSphere host.

In vSphere Storage DRS, its implementation is different. SIOC (on behalf of vSphere Storage DRS) checks the capabilities of the datastores in a datastore cluster by injecting various I/O loads. After this information is normalized, vSphere Storage DRS can determine the types of workloads that a datastore can handle. This information is used in initial placement and load-balancing decisions.

vSphere Storage DRS continuously uses SIOC to monitor how long it takes an I/O to do a round trip. This is the latency. This information about the datastore is passed back to vSphere Storage DRS. If the latency value for a particular datastore is above the threshold value (the default is 15 milliseconds) for a significant percentage of time over an observation period (the default is 16 hours), vSphere Storage DRS will try to rebalance the virtual machines across the datastores in the datastore cluster so that the latency value returns below the threshold. This might involve one or more Storage vMotion operations. In fact, even if vSphere Storage DRS is unable to bring the latency below the defined threshold value, it might still move virtual machines between datastores to balance the latency.

When evaluating vSphere Storage DRS, VMware makes the same best practice recommendation made for vSphere Storage DRS initially. The recommendation is to run vSphere Storage DRS in manual mode first, monitoring the recommendations that vSphere Storage DRS is surfacing and ensuring that they make sense. After a period of time, if the recommendations make sense, and you build a comfort level using vSphere Storage DRS, consider switching it to automated mode.

There are a number of considerations when using vSphere Storage DRS with certain array features. Check your storage vendor's recommendation for using vSphere Storage DRS. There might be specific interaction with some advanced features on the array that you want to be aware of. VMware has already produced a very detailed white paper regarding the use of vSphere Storage DRS with array features such as tiered storage, thin provisioning and deduplication. More details regarding vSphere Storage DRS interoperability with storage-array features can be found in the *VMware vSphere Storage DRS Interoperability* white paper.

VAAI

Many NAS storage arrays now support a number of VMware vSphere Storage APIs–Array Integration (VAAI) primitives. This API enables the ESXi host to offload certain storage operations to the storage array rather than consuming resources on the vSphere host by doing the same operations.

Remember that a VAAI NAS plug-in is required from your respective storage array vendor for any of these primitives to work. The plug-in must be installed on each vSphere host that you require to leverage the VAAI NAS primitives.

The first primitive to discuss is Full File Clone, which enables the vSphere host to offload a cold clone operation or template deployments to the storage array. One important point to note is that this primitive does not support Storage vMotion. The primitive can be used only when the virtual machine is powered off. Storage vMotion on NFS datastores continue to use the VMkernel software data mover.

The next primitive is called Fast File Clone. With this, it is possible to offload the creation of linked clones to the array. With the release of VMware® View™ 5.1, this feature is supported as a tech preview. A future release of View (at the time of writing this) is required for full support of this primitive. With the release of vSphere 5.1 and with VMware vCloud® Director™ 5.1, this primitive is fully supported for VMware vCloud vApps when VAAI is enabled on the datastore and Fast Provisioning Using Linked Clones is selected.

Reserve Space is another VAAI NAS primitive. Without VAAI NAS, one can never preallocate or zero out space for Virtual Machine Disk formats (VMDKs) on NFS. Historically the only option available was to build thin VMDKs on NFS. With the introduction of Reserve Space, one can now create thick VMDKs on NFS datastores. However, VAAI NAS Reserve Space is not like Write Same for block. It does not get the array to do the zeroing on its behalf. When creating a VMDK on a VAAI NAS array, selecting **Flat** sends a Space Reserve NAS VAAI command to the array that guarantees that the space will be available. This is equivalent to VMware vSphere VMFS (Virtual Machine File System) lazyzeroedthick, and the blocks are zeroed on first write. However, selecting **Flat pre-initialized** also sends a Space Reserve NAS VAAI command, but it does ESXi-based zero writing to the VMDK. This is equivalent to a VMFS eagerzeroedthick. This means that it is a slow operation to initially create the VMDK, because all writes are sent to the datastore over the network. The writes are not offloaded. After the VMDK is initialized, however, it should have equivalent performance to other VMDK formats.

As an aside, we just said that VAAI NAS Reserve Space enables you to create virtual disks in Thick Provision Lazy Zeroed (lazyzeroedthick) or Thick Provision Eager Zeroed (eagerzeroedthick) formats on NFS datastores on arrays that support Reserve Space. However, when you check the disk type on the Virtual Machine Properties dialog box, the Disk Provisioning section always shows Thick Provision Eager Zeroed as the disk format no matter which format you selected during the disk creation. ESXi does not distinguish between lazy zeroed and eager zeroed virtual disks on NFS datastores.

The final feature is Extended Stats (NAS). This enables one to query how much space a VMDK actually consumed on an NFS datastore. For example, you might create a 100GB thin VMDK, but actually consume only 25GB of space on the array. This was an issue vSphere previously never addressed. This was not a necessary feature for VMFS, because vSphere understands VMFS very well. But we did need something like this for NFS.

Thin Provisioning (TP) primitives were introduced with vSphere 5.0. Features such as the raising of an alarm when a TP volume reached 75 percent of capacity at the back end, TP-Stun and, of course, the UNMAP primitive were introduced. However these TP primitives are for SCSI only. The VAAI space-threshold alarm is supported only on SCSI datastores. Similarly, the VAAI TP-Stun was introduced to detect out-of-space conditions on SCSI LUNs. However, for NAS datastores, NFS servers can already return an out-of-space error that should be propagated up the stack. This should induce a virtual machine stun similar to VAAI TP. This operation does not need the VAAI NAS plug-in, and should work on all NFS datastores, whether or not the hosts have VAAI enabled. vSphere Storage DRS also leverages this event. After the alarm is triggered, vSphere Storage DRS no longer considers those datastore as destinations for initial placement or ongoing load balancing of virtual machines. Finally, the UNMAP primitive is also for SCSI. The reclaiming of dead space is not an issue on NAS arrays. A detailed explanation of VAAI can be found in the white paper, *VMware vSphere APIs–Array Integration (VAAI)*.

Site Recovery Manager/vSphere Replication

VMware vCenter Site Recovery Manager™ (SRM) fully supports array-based replication on NFS datastores. vSphere Replication fully supports the replication of virtual machines that reside on NFS datastores.

It is a best practice to consider storing the swap in a different directory (or perhaps a different datastore) when using a replicated NFS datastore in SRM. The choice between different directory and different datastore depends on the granularity at which your storage array can replicate, but the idea is to avoid having it replicated. This enables SRM to reduce the amount of replicated content that gets recreated on a failover. It also saves on having to delete and recreate the .vswp files on the destination datastore after a failover. This caused unnecessary delays during failover in earlier versions of SRM because file handles on NFS needed to expire on the .vswp files before being deleted and recreated. These delays are reduced in version 5 of SRM.

Another consideration is the use of Fully Qualified Domain Names (FQDN) rather than IP addresses when mounting NFS datastores. Some storage-array vendors require you to use IP addresses when using their Storage Replication Adapter (SRA) with SRM. Discuss with your storage-array vendor the question of whether or not this is required.

Storage vMotion

Storage vMotion has gone through quite a few architectural changes over the years. The latest version in vSphere 5.x uses a mirror driver to split writes to the source and destination datastores after a migration is initiated. This means speedier migrations because there is only a single copy operation now required, unlike the recursive copy process used in previous versions that leveraged Change Block Tracking (CBT).

One consideration that has been called out already is that Storage vMotion operations of virtual machines between NFS datastores cannot be offloaded to the array with VAAI. The software data mover does all Storage vMotion operations on NFS datastores, even if the vSphere host has a VAAI NAS plug-in from the array vendor.

The only other considerations with Storage vMotion are relevant to both block and NAS. This is the configuration maximums. At the time of this writing, the maximum number of concurrent Storage vMotion operations per vSphere host was two and the maximum number of Storage vMotion operations per NFS datastore was eight. This is to prevent any single datastore from being unnecessarily impacted by Storage vMotion operations.

NOTE: A new enhancement in vSphere 5.1 enables up to four VMDKs belonging to the same virtual machine to be migrated in parallel, as long as the VMDKs reside on unique datastores.

Check your storage vendor's recommendation for using Storage vMotion. There might be specific interaction with some advanced features on the array that you must be aware of when moving a virtual machine from one datastore to another datastore.

Sizing Considerations

Recommended Volume Size

The following statement appears in the *VMware Configuration Maximums Guide*: "Contact your storage array vendor or NFS server vendor for information about the maximum NFS volume size." When creating this paper, we asked a number of our storage partners if there was a volume size that worked well. All partners said that there was no performance gain or degradation depending on the volume size and that customers might build NFS volumes of any size, so long as it was below the array vendor's supported maximum. This can be up in the hundreds of terabytes, but the consensus is that the majority of NFS datastores are in the tens of terabytes in terms of size. The datastores sizes vary greatly from customer to customer.

Sizing of volumes is typically proportional to the number of virtual machines you attempt to deploy, in addition to snapshots/changed blocks for backup purposes. Another consideration is that many arrays now have deduplication and compression features, which will also reduce capacity requirements. A final consideration is Recovery Point Objective (RPO) and Recovery Time Objective (RTO). These determine how fast you can restore your datastore with your current backup platform.

Recommended Block Size

This parameter is not tunable for the most part. Some vendors have it hard set to 4KB and others have it hard set to 8KB. Block sizes are typically a multiple of 4KB. These align nicely with the 4KB grain size used in the VMDK format of VMware. For those vendors who have it set to 8KB, the recommendation is to format the volumes in the guest operating system (OS) to a matching 8KB block size for optimal performance. One vendor did have a tunable block size, and they had different recommendations depending on the workload. These were 4KB for random workloads and 32KB for sequential workloads. Because having multiple virtual machines sharing an NFS datastore results in random workloads, this vendor would probably recommend that you use 4KB for vSphere infrastructures. In this area, it is best to speak to your storage-array vendor to get vendor-specific advice.

The questions one should ask are as follows:

1. What is the volume block size on the array?
2. Is it tunable?
3. If so, what should I tune it to? (Be sure to explain that the datastore will be used by virtual machines, so the workload will be random for the most part.)
4. Are there any considerations when formatting the volumes in the guest OS?

Rsize and Wsize

The rsize and wsize parameters, defined at mount time, define the I/O chunk transfer sizes between the host and the target. These sizes are not tunable and are hard set to 64KB. Feedback from VMware's storage partners suggests that the current size of 64KB meets their requirements. Although some storage arrays can now support much larger wsize and rsize values (up to 1MB), there is still no way to change these settings in vSphere at the time of this writing. It is included here for the sake of completeness, because the query has come up in the past.

Maximum Number of Virtual Machines per NFS Datastore

The number of virtual machines that can run on a single datastore is directly proportional to the infrastructure and the workloads running in the virtual machines. For example, one might be able to run many hundreds of low-I/O virtual machines but only a few very intensive I/O virtual machines on the same datastore. Network congestion is an important factor. Because VMware still supports only NFS version 3 over TCP/IP, one recommendation is to use more datastores presented over different interfaces so that traffic might be balanced over multiple TCP sessions.

The other major factor is related to the backup and recovery Service Level Agreement (SLA). If you have one datastore with many virtual machines, there is a question of how long you are willing to wait while service is restored in the event of a failure. This is becoming the major consideration in the debate over how many virtual machines per datastore is optimal.

Another consideration is the snapshot technology used by the backup product. There is a question of whether it uses array-based snapshots or virtual machine snapshots. Performance considerations are required if using virtual machine snapshots to concurrently capture point-in-time copies of virtual machines. In many cases, array-based snapshots have less impact on the datastores and are more scalable when it comes to backups. There might be some array-based limitations to take into consideration also. For instance, the number of snapshot copies of a virtual machine that a customer wants to maintain might exceed the number of snapshot copies an array can support. This varies from vendor to vendor. Check this configuration maximum with your storage-array vendor.

For a vSphere snapshot perspective, users can refer to this KB article 1015180 for further details regarding snapshots and their usage. As shown in KB article 1025279, virtual machines can support up to 32 snapshots in a chain, but VMware recommends that you use only 2–3 snapshots in a chain, and also that you use no single snapshot for more than 24–72 hours.

Advanced Settings

There are a number of tunable parameters available when using NFS datastores. Before drilling into these advanced settings in more detail, you should understand that the recommended values for some of these settings might (and probably will) vary from storage-array vendor to storage-array vendor. The objective here is to give you a clear and concise explanation of the tunable parameters and enable you to make your own decisions when tuning the values.

Maximum Number of NFS Mounts per ESXi Host

By default, the NFS.MaxVolumes value is 8. This means that 8 are the maximum number of NFS volumes that can be mounted to an ESXi host by default. This can be changed, because VMware supports a maximum of 256 NFS volumes mounted to an ESXi host. However, storage-array vendors might make their own recommendation around this value. The recommendations to change the NFS.MaxVolumes value vary from storage-array vendor to storage-array vendor. Check the appropriate documentation from the vendor. This is a per–ESXi host setting and must be done on all hosts.

TCP/IP Heap Size

Net.TcpIpHeapSize is the size of the memory (in megabytes) that is allocated up front by the VMkernel to TCP/IP heap. Net.TcpIpHeapMax is the maximum amount of memory that can be consumed by TCP/IP as heap. In vSphere 5.1, the default value for Net.TcpIpHeapSize is 0MB and the default value for Net.TcpIpHeapMax is 64MB. The maximum value for Net.TcpIpHeapMax is 128MB. When one changes the default NFS.MaxVolumes as discussed previously, one also must adjust the heap space settings for TCP/IP accordingly. Again, follow the advice from your storage-array vendor. The vendors make different recommendations for these values. These are per–ESXi host settings and, again, must be done on all hosts. Because the maximum value is only 128MB, it is advisable to set this to its maximum.

Heartbeats

The four heartbeat settings (NFS.HeartbeatFrequency, NFS.HeartbeatDelta, NFS.HeartbeatTimeout and NFS.HeartbeatMaxFailures) can be discussed together. Basically, they are used to check that an NFS datastore is operational. The NFS.HeartbeatFrequency is set to 12 seconds by default. This means that every 12 seconds, the ESXi host checks to see if it must request a heartbeat from an NFS datastore and ensure that the datastore is still accessible. To prevent unnecessary heartbeats the host requests a new heartbeat only from the datastore if it hasn't done any other operation to the datastore, confirming its availability, in the last interval defined by NFS.HeartbeatDelta (the default is 5 seconds).

So what happens if the datastore is not accessible? This is where NFS.HeartbeatTimeout and NFS.HeartbeatMaxFailures come in.

The NFS.HeartbeatTimeout value is set to 5 seconds. This is how long we wait for an outstanding heartbeat before we give up on it. NFS.HeartbeatMaxFailures is set to 10 by default. So if we have to give up on 10 consecutive heartbeats, we treat the NFS datastore as unreachable. If we work this out, it is 125 seconds (10 heartbeats at 12-second intervals + 5 seconds time-out for the last heartbeat) before we decide an NFS datastore is no longer operational and mark it as down.

vSphere hosts continue to make heartbeat requests, however, in the hope that the datastore does become available once again.

There are no specific vendor recommendations to change NFS.HeartbeatFrequency, NFS.HeartbeatTimeout or NFS.HeartbeatMaxFailures from the default. It is likely that the default values meet the requirements of most, if not all, vendors.

Locking

Similar to heartbeat, the lock-related settings (NFS.DiskFileLockUpdateFreq, NFS.LockUpdateTimeout and NFS.LockRenewMaxFailureNumber) can be discussed together.

To begin with, VMware isn't using the Network Lock Manager (NLM) protocol for NFS locking. Rather, VMware uses its own locking mechanism for NFS. VMware implements NFS locks by creating lock files named .lck-<file_id> on the NFS server.

To ensure consistency, I/O is issued to the file on an NFS datastore only when the client is the lock holder and the lock lease has not expired yet. After a lock file is created updates are sent to the lock file at intervals defined by NFS.DiskFileLockUpdateFreq (the default is 10 seconds). This informs the other ESXi hosts that the lock is still active.

Locks can be preempted. Consider a situation where VMware vSphere High Availability (HA) detects a host failure and attempts to start a virtual machine on another host in the cluster. In this case, another host must be able to take ownership of that virtual machine, so there must be a method to time out the previous lock. By default, a host will make three polling attempts (defined by NFS.LockRenewMaxFailureNumber) at 10-second intervals (defined by NFS.DiskFileLockUpdateFreq) to update the file lock. Each lock update attempt has a 5-second time-out (defined by NFS.LockUpdateTimeout).

In the worst case, when the last lock update attempt times out, it will take 3 * 10 + 5 = 35 seconds before the lock is marked expired on the lock holder client. Before the lock is marked expired, I/O will continue to be issued, even after failed lock update attempts.

Lock preemption on a competing client starts from the detection of a lock conflict. It then takes three polling attempts with 10-second intervals for the competing host to declare that the lock has expired and to break it. It then takes another 10 seconds for the host to establish its own lock. Lock preemption will be completed in 3 * 10 + 10 = 40 seconds before I/O will start to flow on the competing host.

There are no specific vendor recommendations to change these values from the default. Again, always double-check the vendor's documentation to make sure.

Finally, it is extremely important that any changes to the lock settings are reflected on all hosts sharing the datastore. If there are inconsistent lock settings across multiple hosts sharing the same NFS datastore, the results can be very undesirable.

TCP Connections per IP Address

The advanced setting SunRPC.MaxConnPerIP defines the maximum number of unique TCP connections that can be opened for a given IP address. This is of particular interest to users of NFS. If the number of mounts to an IP address is more than the number defined by SunRPC.MaxConnPerIP, then the existing connections are shared. Currently VMware supports as many as 128 unique TCP connections per vSphere host but also supports as many as 256 mounts per host. Therefore, it is required to share the existing TCP connections to mount 256 volumes.

Let's take the following example. If there are four IP addresses from which mounts are exposed from an NFS host or NAS array, and SunRPC.MaxConnPerIP is set to its default value of four (in vSphere 5.1), then this will give 4 * 4 = 16 unique TCP connections.

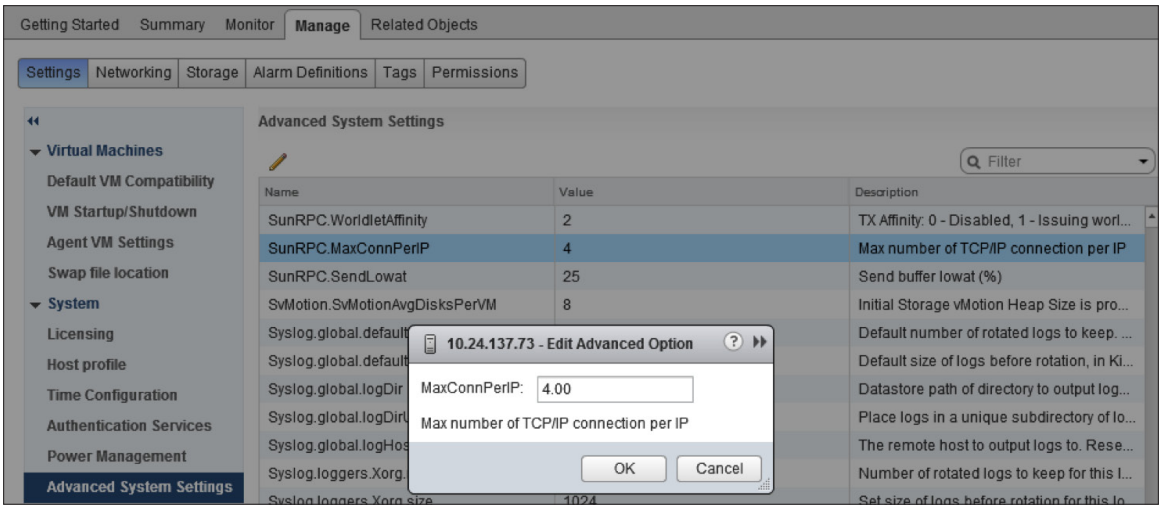

Figure 10. MaxConnPerIP Advanced Setting

By setting the value to 32, this will give 4 * 32 = 128 unique TCP connections. However, if each IP address is exporting 64 mounts, then the connection to these 64 volumes will be shared among the 32 TCP connections defined in SunRPC.MaxConnPerIP. With 256 volumes exported from four IP addresses, each connection will share two mounts in this scenario.

So why must you be concerned about shared connections? For the most part, you do not. However, if you currently have 128 unique TCP connections consumed by NFS mounts and you attempt to mount an NFS datastore from a new IP address, the operation will not be successful and will fail with the message "Error while adding NFS mount: NFS connection limit reached!" Mounts of NFS datastores from already existing IP addresses will be successful because they can share connections. VMware KB article 1007909 has further definitions on the advanced NFS options.

Additional Attributes of NFS Storage

There are several additional options to consider when using NFS as a shared storage pool for virtualization. Some additional considerations are thin provisioning, deduplication and the ease of backup and restore of virtual machines, virtual disks and even files on a virtual disk via array-based snapshots. The next section focuses on the options available and the criteria that might make one choice a better selection than the alternative.

Thin Provisioning

VMDKs created on NFS datastores are in thin-provisioned format by default. This capability offers better disk utilization of the underlying storage capacity in that it removes what is often considered wasted disk space. For the purposes of this paper, VMware will define wasted disk space as allocated but not used. The thin-provisioning technology removes a significant amount of wasted disk space.

On NFS datastores, the default virtual disk format is thin. This means that less allocation of volume storage space is needed for the same set of virtual disks provisioned as thick format. VMware vCenter now provides support for both the creation of thin virtual disks as well as the monitoring of datastores that might be overcommitted. As mentioned earlier, if the host has a VAAI NAS plug-in from the storage-array vendor, thick VMDKs can also be provisioned on NFS datastores.

Deduplication

Some NAS storage vendors offer data deduplication features that can greatly reduce the amount of storage space required. It is important to distinguish between in-place deduplication and deduplication for backup streams. Both offer significant savings in space requirements, but in-place deduplication seems to be far more significant for virtualization environments. Some customers can reduce their storage needs by up to 75 percent of their previous storage footprint with the use of in-place deduplication technology.

Backup and Restore Granularity

The backup of a VMware virtual environment can be done in several ways. Some common methods are to use VMware vSphere Data Protection (VDP), to use third-party products that leverage VMware vSphere Storage APIs–Data Protection (VADP), placing third-party backup agents in each guest OS, or to leverage array-based snapshot technology. However, the process of restoring the virtual machine becomes more complicated because there are often many levels of granularity that are requested for restore. Some recovery solutions might require the entire datastore to be restored whereas others might require a specific virtual machine or even a virtual disk for a virtual machine to be recovered. The most granular request is to have only a specific file on a virtual disk for a given virtual machine restored.

With NFS and array-based snapshots, one has the greatest ease and flexibility regarding what level of granularity can be restored. With an array-based snapshot of an NFS datastore, one can quickly mount a point-in-time copy of the entire NFS datastore, and then selectively extract any level of granularity they want. Although this does involve a security risk, NFS does provide one of the most flexible and efficient restore from backup options available today. For this reason, NFS earns high marks for ease of backup and restore capability.

FQDN and IP Address

There is some historical context to the mounting of an NFS datastore on a vSphere host with Fully Qualified Domain Names (FQDN) versus mounting with IP addresses. The recommendation to use IP addresses was originally made to eliminate any complexities encountered when relying on DNS. If you use FQDNs, and your DNS infrastructure suffers a failure, you will not be able to mount datastores.

Another concern is if DNS was resolving FQDN incorrectly, there might be issues with some vSphere features. For instance, if two vSphere hosts were mounting the same datastore, but the FQDN resolved to different IP addresses on each host, the datastore on the second host would be renamed and have an appended “(1)” on the name. This led to unwanted behavior with vSphere features such as vMotion and vSphere HA not working in the past. In vSphere 5.0, a number of enhancements were made in this area. Therefore, even if the FQDNs are resolved differently, the NFS datastores will not be renamed, and features such as vMotion and vSphere HA work as expected.

In the past, some of our storage partners also made recommendations on which method to use, for example, in the case of SRM interoperability.

From the perspective of VMware, both methods are fully supported with NFS. There is currently no requirement to use IP address over FQDN currently, or vice versa, but you should always check with your array vendor about their latest recommendation.

Round-Robin DNS

Round-Robin DNS can be useful in the context of load balancing. If you have an NAS array with multiple network interfaces from which to mount volumes, DNS round robin will work on a rotating basis, so that each client connection will receive a new IP address for mounting. After the address is handed out to a client, it is placed at the back of the list and the next client request gets a new IP address. This can be a useful method for automating some load balancing/load sharing.

Disk Alignment

This is not a recommendation specific to NFS, as it also can have an adverse effect on the performance of block storage. Nevertheless, in the context of trying to cover all bases, it should be considered a best practice to have the partitions of the guest OS running with the virtual machine aligned to the storage. Detailed descriptions on how to do this alignment are beyond the scope of this white paper. Refer to the documentation of your individual storage-array vendor for further details.

Considerations for .vswp File Placement

Historically, VMware discouraged the placement of the .vswp (Virtual Machine swap) file on NFS datastores. When the VMkernel is not able to allocate any additional memory to a virtual machine, it must use this on-disk swap file. This was a concern only when customers used nonenterprise NFS servers, and performance seriously degraded when a virtual machine was swapping. This recommendation no longer applies to modern enterprise NAS devices.

Conclusion

Network-attached storage has matured significantly in recent years and it offers a solid availability and high performance foundation for deployment with virtualization environments. Following the best practices outlined in this paper will ensure successful deployments of vSphere on NFS. Many people have deployed vSphere on NFS and are very pleased with the results they are experiencing. NFS offers a solid storage platform for virtualization.

Both performance and availability are holding up to expectations and as best practices are being further defined, the experience of running VMware technology on NFS is proving to be a solid choice of storage protocols. Several storage technology partners are working closely with VMware to further define the best practices and extend the benefits for customers choosing to deploy VMware vSphere on this storage protocol option.

Acknowledgments

I would like to thank James Walkenhorst of EMC, Paul Morrissey of HDS and Joel Kaufman, Peter Learmonth and Nick Howell of NetApp for contributing to this paper. I would also like to thank Duncan Epping, Frank Denneman and Glenn Skinner of VMware for their technical review of the white paper.

About the Author

Cormac Hogan is a senior technical marketing architect within the Cloud Infrastructure Product Marketing group at VMware. He is responsible for storage in general, with a focus on core VMware vSphere storage technologies and virtual storage, including the VMware vSphere Storage Appliance. Cormac has written a number of storage-related white papers and has given numerous presentations on storage best practices and new features. He has been with VMware since 2005.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com