Best Practices For Running NFS with VMware vSphere

VMware Storage

Table of contents

Best Practices For Running NFS with VMware vSphere

Introduction

This section briefly covers how NFS and NAS have affected the virtualization environment, and why running vSphere on NFS is a very viable option for many virtualization deployments.

The significant presence of Network Filesystem Storage (NFS) in the datacenter today has lead to many people deploying virtualization environments with Network Attached Storage (NAS) shared storage resources.

For the purpose of clarity, both NFS and NAS refer to the same type of storage protocol and the terms will be used interchangeably throughout this paper.

Running vSphere on NFS is a very viable option for many virtualization deployments as it offers strong performance and stability if configured correctly. The capabilities of VMware vSphere™ on NFS are very similar to the VMware vSphere™ on block-based storage. VMware offers support for almost all features and functions on NFS—as it does for vSphere on SAN.

This paper provides an overview of the considerations and best practices for deployment of VMware vSphere on NFS based storage. It also examines the myths that exist and will attempt to dispel confusion as to when NFS should and should not be used with vSphere. It will also provide some troubleshooting tips and tricks.

Background

This section provides a background on the first use of NFS and iSCSI and how they worked.

VMware has supported IP based storage for quite a number of years. Both iSCSI and NFS storage were introduced in VI3 back in 2006 as storage resources that could be shared across a cluster of ESXi servers.

The considerations for choosing a storage resource (e.g. NAS, block, HCI) tend to hinge on the issue of cost, performance, availability, and ease of manageability. However, an additional factor could also be the legacy environment and the storage administrator familiarity with one protocol vs. the other based on what is already installed.

The more important consideration that often leads people 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 fiber channel (FC) storage to set up and maintain.

Overview of the Steps to Provision NFS Datastores

This section provides an overview of the steps to provision NFS datastores.

Before NFS storage can be addressed by an ESXi host, the following issues need to be addressed:

1. Have a virtual switch with a VMkernel NIC configured for IP based storage.
2. The NFS storage target needs to have been configured to export a mount point that is accessible to the ESXi hosts on a trusted network. For more details on NFS storage options and setup, consult the best practices for VMware provided by your storage vendor.

Regarding item one above, to configure the vSwitch for IP storage access you will need to create a new portgroup, indicating it is a VMkernel type connection. You will need to populate the network access information.

With these items addressed, an NFS datastore can now be added to the ESXi host following a similar process to the one used to configure a data store for block based datastores. For item two, when mounting the NFS storage on the ESXi host(s), one would need to provide the NFS storage target mount point, the IP or hostname of the target, and a datastore name that will be used to identify the NFS storage on the ESXi host(s).

Networking Settings

This section explains the different types of network settings and how they work in different VMware NFS versions.

VMware supports both NFS version 3 and NFS version 4.1 over TCP/IP. NFS version 3 has been supported since the early days of ESXi. Support for NFS version 4.1 was introduced in vSphere 6.0.

There are still some limits to the multipathing and load-balancing approaches that can be taken. Although there are two connections when an NFS datastore is mounted to an ESXi host (one connection for control, the other connection for data), there is still only a single TCP session for I/O.

It is important to understand that with NFS version 3 there is only one active pipe for the connection between the ESXi host and a single storage target. This means that although there may 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 with NFS version 3.0, an ESXi host may have multiple connections to the storage targets. One would need 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. The configuration of traffic spread across two or more datastores configured on separate connections between the ESX server and the storage array is load sharing.

With NFS version 4.1, new multipathing and load balancing capabilities are introduced. It is important to understand that these new features can only be leveraged so long as the NFS target supports them. Check the supportability with your NAS vendor.

Throughput Options with NFS

Let us now 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. These are applicable to both NFS v3 and NFS v4.1.

1. **10GigE** . 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 1GigE pipe, then a fatter pipe isn't going to help you. But let's assume that there are enough virtual machines and enough datastores for 10GigE to be beneficial.

2. **Jumbo Frames** . While this feature can deliver additional throughput by increasing the size of the payload in each frame from a default MTU of 1500 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 it to make sense (array controller, physical switches, NICs and VMkernel ports).

3. **Load Sharing** . One of the configuration options for NFS is to use multiple connections from the ESXi host to the storage targets. To implement this, one would need to configure multiple datastores, with each datastore using separate connections between the host and the storage, i.e. NFS shares presented on different IP addresses, as shown in the following diagram:

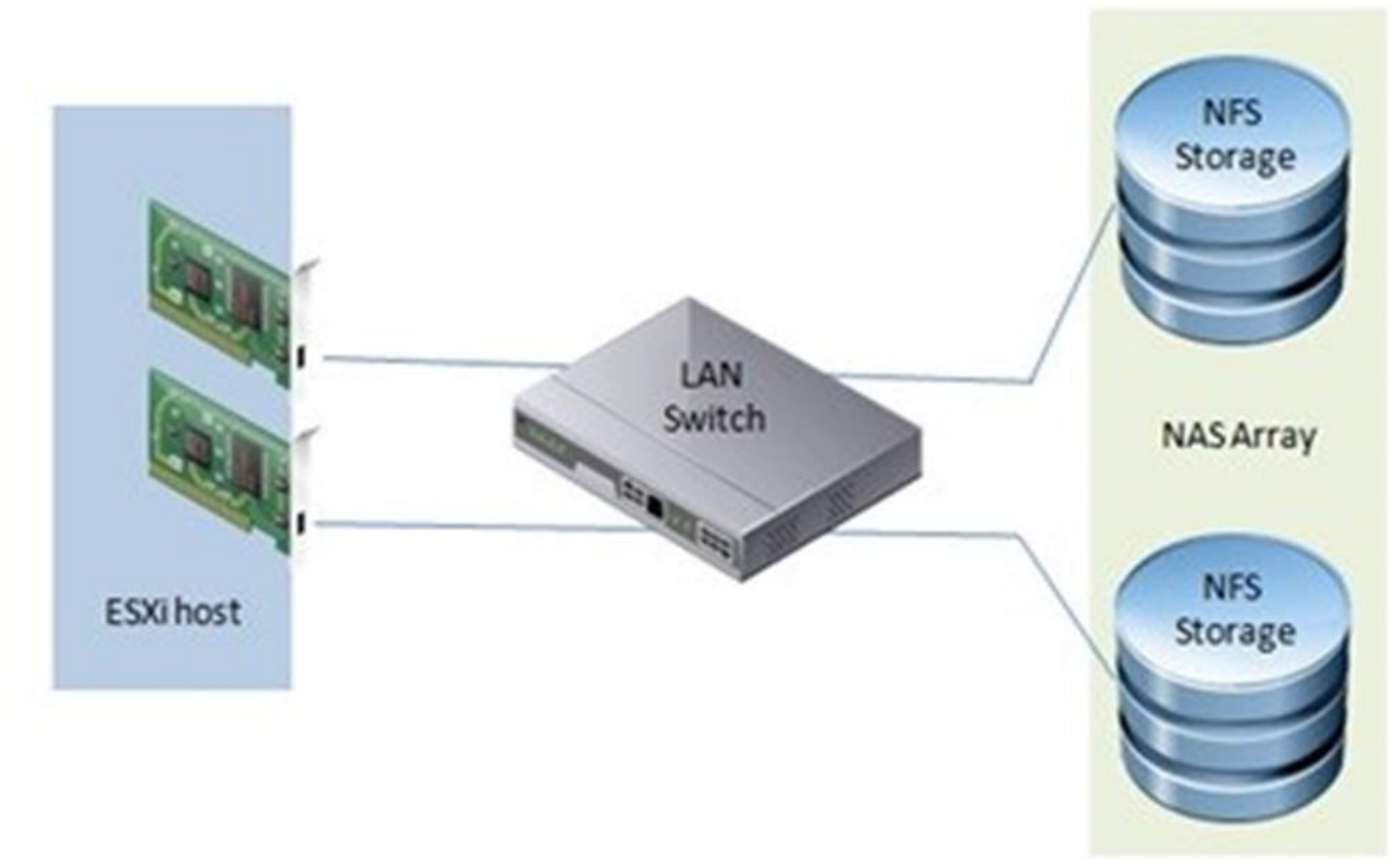

Figure 1 – NFS datastores

Figure 1 – NFS datastores presented via different IP Addresses

4. **Link Aggregation** . Another possible way to increase throughput is via the use of link aggregation. Multiple VMkernel interfaces are bound in a LAG (Link Aggregation Group) and this is then used to access the NFS target. This isn't always guaranteed to deliver additional performance, but does in many cases. Link Aggregation is discussed in the context of availability later on in the post. For detailed configuration steps, refer to the official vSphere Networking documentation on vmware.com.

Throughput Options with NFS version 4.1

NFS version 4.1 allows the same datastore to be presented via multiple data paths. The use of multiple paths enables both redundancy and load-balancing. This is one of the benefits of NFS version 4.1 over NFS version 3. With load balancing comes the ability to improve throughput of course. Note that the NFS target must support multipathing for this feature to work.

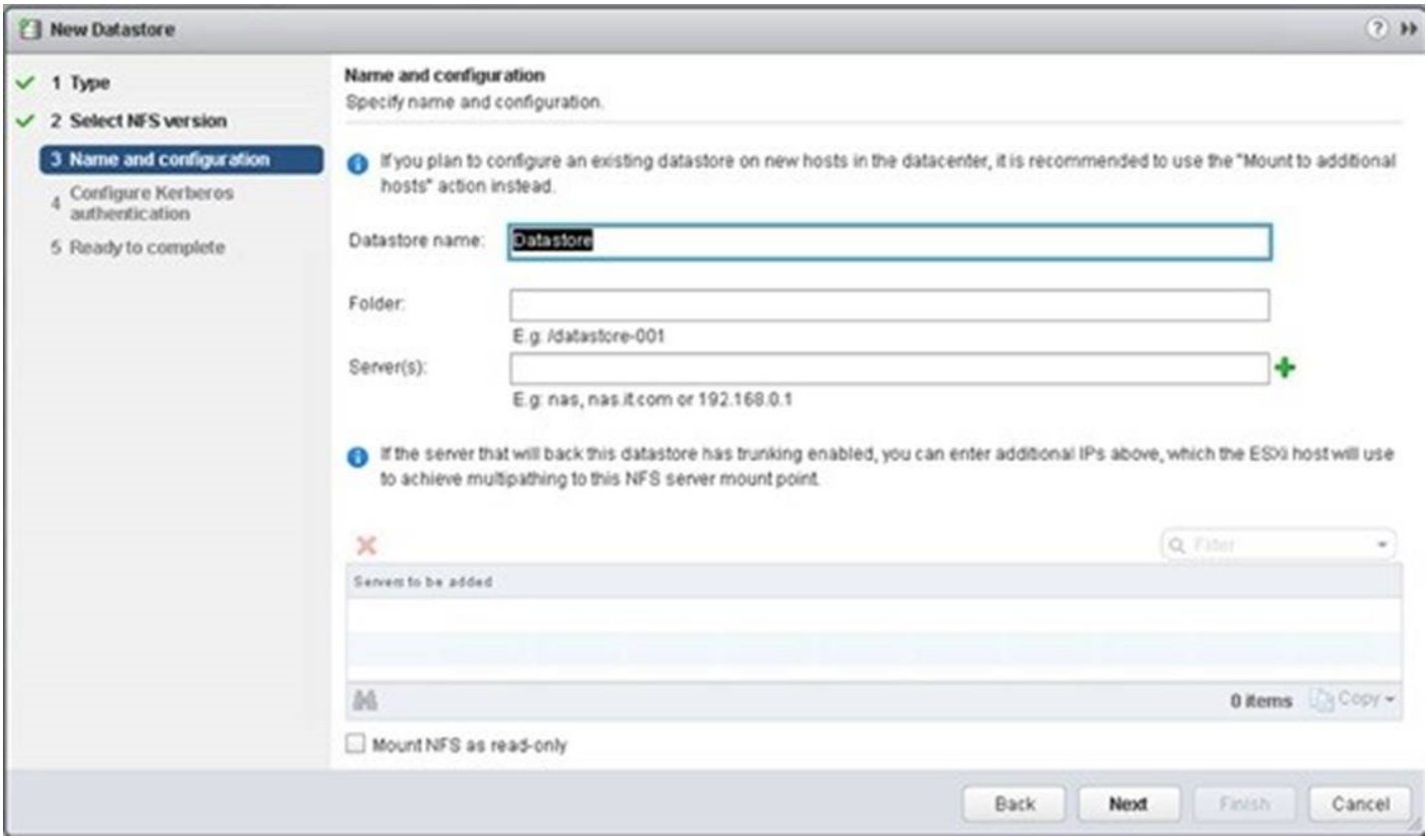

Figure 2 - Multipathing setup with NFS v4.1

Minimizing Latency

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

Ideally, one would not route between the ESXi host and the NFS target; one should try to keep them both on the same subnet. In fact, prior to ESXi 5.0U1, one could not route between the ESXi host and the NFS target, but this restriction was removed in vSphere release 5.0U1. There are still quite a number of restrictions with routing NFS and the vSphere release notes should be examined in detail to determine whether it is suitable for your environment.

Security with NFS version 3

All NAS array vendors agree that it is good practice to isolate NFS traffic for security reasons. By default, NFS traffic is sent in clear text over the traffic. Therefore, it is considered best practice to use NFS storage on trusted networks only. This would mean isolating the NFS traffic on its own separate physical switches or leveraging a dedicated VLAN (IEEE 802.1Q).

Another security concern is that the ESXi host mounts the NFS datastores using root privileges. Since this is NFS version 3, none of the security features implemented in later versions of NFS are available. To address the concern, again it is considered a best practice to use either a dedicated LAN or a VLAN for protection and isolation.

Security with NFS version 4

Another major enhancement with NFS v4.1 is the security aspect. With this version, Kerberos and thus non-root user authentication are both supported. With version 3, remote files were accessed with root permissions, and servers had to be configured with the *no_root_squash* option to allow root access to files (more on this shortly). This is known as the AUTH_SYS mechanism. While this method is still supported with NFS v4.1, Kerberos is a much more secure mechanism. An AD user is now defined on each ESXi host for the NFS Kerberos Credentials, and this is the user that is used for NFS (remote file access).

Figure 3 - NFS Kerberos

Figure 3 - NFS Kerberos Credentials, per host

There is a requirement on Active Directory for this to work, and each ESXi host should be joined to the AD domain. Kerberos is enabled when the NFS v4.1 datastore is being mounted to the ESXi host. A warning message is displayed in the vSphere client that each host mounting this datastore needs to be part of an AD domain.

In vSphere 6.5, an additional Kerberos feature was added to NFSv41 which was to use Kerberos for authentication and data integrity (krb5i):

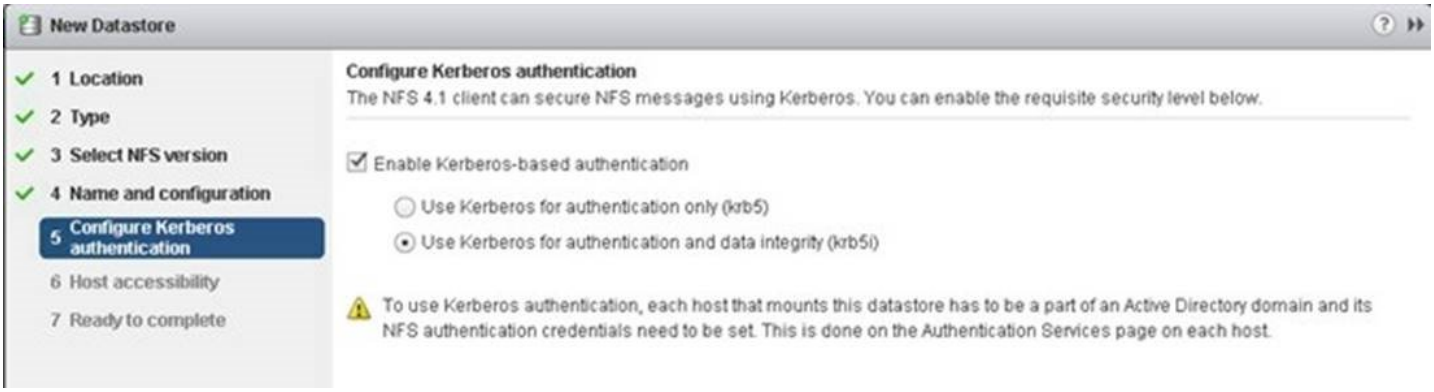

Figure 4 - Selecting krb5 for

Figure 4 - Selecting krb5 for authentication only, or krb5i for authentication and data integrity

Note: One should use the same AD/Kerberos user on all hosts. If two hosts are using different users, you might find that a vMotion task will fail.

Availability

To achieve high availability, the LAN on which the NFS traffic will run needs to be designed with availability, downtime-avoidance, isolation, and no single-fail-point of failure in mind. Multiple administrators need to be involved in designing for high-availability: Both the virtualization administrator and the network administrator. This section outlines these steps and investigates a number of options which can be utilized to make your NFS datastores highly available.

1) **NIC teaming** can be enabled at the host level with IP hash failover enabled. This will allow for redundancy, but will not provide much in the way of load-balancing. A common practice is to set the NIC Teaming fail-back Option to **no** . The reason for this is to avoid a flapping NIC if there is some intermittent issue on the network.

Figure 5 – NIC Teaming for

Figure 5 – NIC Teaming for Availability

The above design is somewhat simplified. There are still issues with the physical LAN switch being a single point of failure (SPOF). To avoid this, a common design is to use NIC teaming in a configuration which has two physical switches. With this configuration, there are four NIC cards in the ESXi host, and these are configured in two pairs of two NICs with IP hash failover. Each pair is configured as a team at their respective LAN switch.

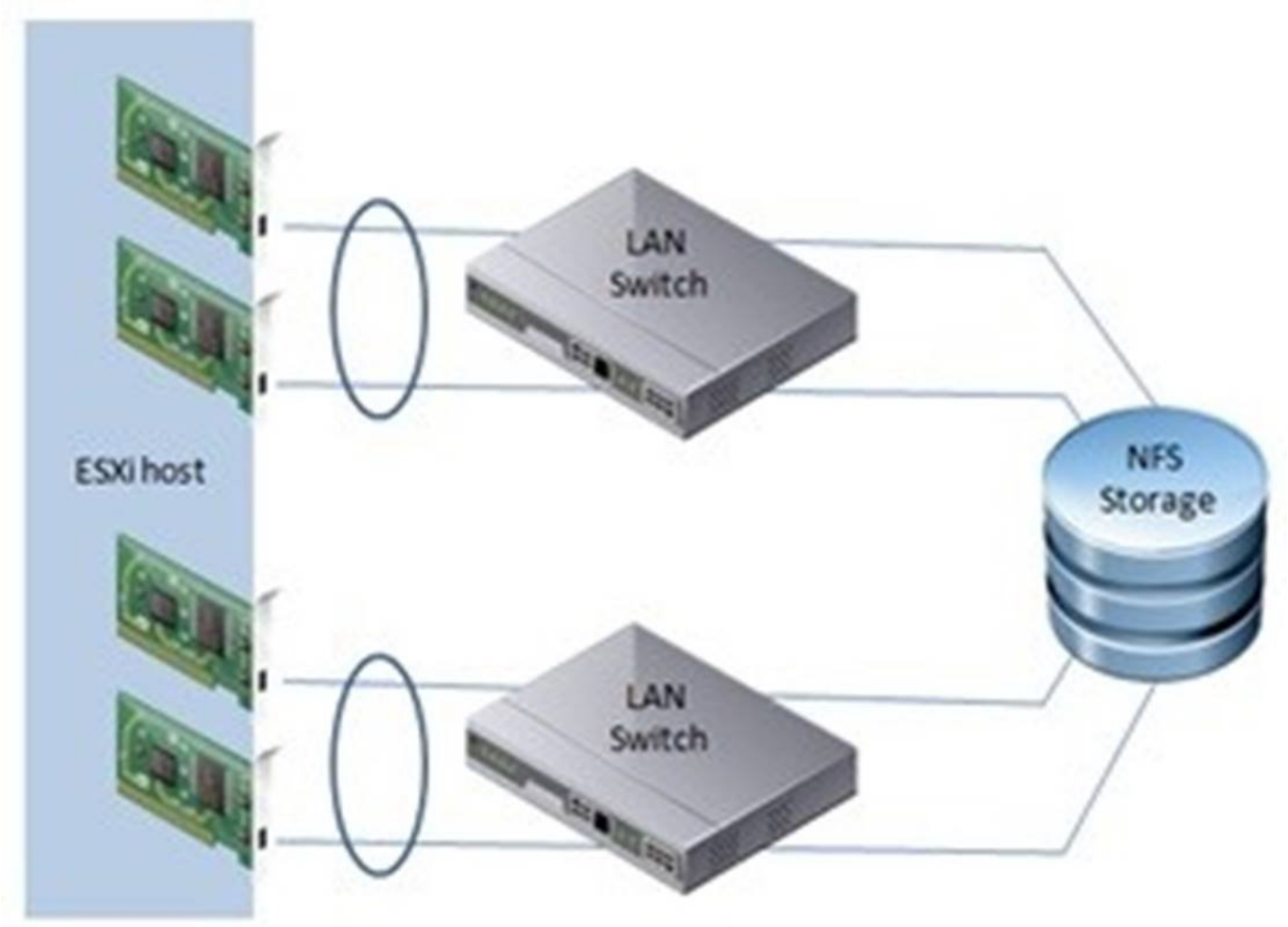

Figure 6 – NIC Teaming with

Figure 6 – NIC Teaming with redundant switch for Availability

2) **Link Aggregation Control Protocol** (LACP) is another option one could consider. Link Aggregation enables you to combine multiple physical interfaces into a single logical interface. Now this may or may not improve throughput/performance since NFS version 3 is limited to a single connection, but it does allow protection against path failures. Many NFS array vendors support this feature at the storage controller port level.

One of the features of LACP is its ability to respond to events on the network and decide which ports should be part of the logical interface. Many failover algorithms only respond to link down events. This means that a switch could inform the array that an alternate path needs to be chosen, rather than the array relying on a port failure.

Support for Link Aggregation on ESXi was introduced in vSphere 5.1. This feature may provide additional availability in so far as a failover to an alternate NIC can now occur based on feedback from the physical switch as opposed to just relying on a link failure event.

Miscellaneous Network Features

By way of completeness, 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 (1GigE or 10GigE), some vendors make recommendations about turning flow control off and allowing congestion to be managed higher up the stack. One should always refer to storage vendor's best practices for guidelines.

The second is a recommendation around switch ports when **Spanning Tree Protocol** (STP) is used in an environment. STP is responsible for ensuring that there are no network loops in a bridged network by disabling network links and ensuring that there is only a single active path between any two network nodes. If there are loops, this can have severe performance impacts on your network with unnecessary forwarding of packets taking place, eventually leading to a saturated network. Some storage array vendors recommend setting the switch ports which connect their storage ports as either RSTP edge ports or Cisco portfast. This

means that ports immediately transition its forwarding state to active. Refer to your storage vendor's best practices for advice on this setting, and if it is appropriate for their storage target.

Advanced Settings

This section provides an explanation of the tunable parameters that are available when using NFS datastores.

There are quite a number of tunable parameters which are available to you when using NFS datastores. Before we drill into these advanced settings in a bit more detail, it is important to understand that the recommended values for some of these settings may (and probably will) vary from storage array vendor to storage array vendor. My objective is to give you a clear and concise explanation of the tunable parameters and allow you to make your own decisions when it comes to tuning the values.

Maximum Number of NFS Mounts per ESXi host

By default, for NFS version 3, the advanced parameter NFS.MaxVolumes value is 8. For NFS Version 4, the advanced parameter NFS41.MaxVolumes is also set to 8. This means that 8 is the maximum number of NFS volumes which can be mounted to an ESXi host by default. This can be changed, as VMware supports a maximum of **256** NFS volumes mounted to an ESXi host. However, storage array vendors make their own recommendation around this value. The recommendations to change the setting varies from storage vendor to storage vendor. It is best to 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 MB) which is allocated up front by the VMkernel to TCP/IP heap. Net.TcpIpHeapMax is the maximum amount of memory which can be consumed by TCP/IP as heap. In vSphere 6.5, the default value for Net.TcpIpHeapSize is 0MB and the default value for Net.TcpIpHeapMax is 512MB. The maximum value for Net.TcpIpHeapMax is 1.5GB (1536MB).

In earlier versions of ESXi, the default max TCPIP Heap size was much smaller than it is now. VMware used to recommend increasing this values as the number of NFS volumes was increased. Again, follow the advice from your storage vendor. The vendors make different recommendations for these values. Since default heap size is 512MB in the current version of ESXi, it should be sufficient even for 256 NFS volumes mounted. These are per ESXi host settings and again must be done on all hosts. Changing this advanced setting requires a host reboot to take effect.

Heartbeats

The four heartbeat settings: NFS.HeartbeatFrequency, NFS.HeartbeatDelta, NFS.HeartbeatTimeout and NFS.HeartbeatMaxFailures can be discussed together. Basically, they are used for checking that an NFS v3 datastore is operational. The NFS.HeartbeatFrequency is set to 12 seconds by default. This means that every 12 seconds, the ESXi host will check to see if it needs to request a heartbeat from an NFS datastore and ensure that the datastore is still accessible. To prevent unnecessary heartbeating, the host will only request a new heartbeat from the datastore if it hasn't done any other operation to the datastore, confirming its availability, in the last NFS.HeartbeatDelta (default: 5 seconds).

This begs the question "what happens if the datastore is not accessible?" This is where NFS.HeartbeatTimeout & NFS.HeartbeatMaxFailures come in.

The NFS.HeartbeatTimeout value is set to 5 seconds. This is how long ESXi waits for an outstanding heartbeat before giving up on it. NFS.HeartbeatMaxFailures is set to 10 by default. If ESXi gives up on 10 consecutive heartbeats, it treats the NFS datastore as unreachable. This is 125 seconds in total (10 heartbeats at 12 second intervals + 5 seconds time-outs for the last heartbeat) before ESXi decides an NFS datastore is no longer operational.

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

There are no specific vendor recommendations to change NFS.HeartbeatFrequency, NFS.HeartbeatTimeout, NFS.HeartbeatDelta or NFS.HeartbeatMaxFailures from the default. One suspects the default values meet the requirements of most, if not all vendors.

Locking

Similar to heartbeat, the lock related settings for NFS v3 are NFS.DiskFileLockUpdateFreq, NFS.LockUpdateTimeout & NFS.LockRenewMaxFailureNumber and can be discussed together.

To begin, the first thing to mention is that VMware isn't using the Network Lock Manager (NLM) protocol for NFS locking. Rather, VMware is using its own proprietary 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 only ever issued to the file on an NFS datastore when the client is the lock holder and the lock lease has not expired yet. Once a lock file is created, updates are sent to the lock file every NFS.DiskFileLockUpdateFreq (default: 10) seconds. This lets the other ESXi hosts know that the lock is still active.

Locks can be preempted. Consider a situation where vSphere HA detects a host failure and wishes to start a VM on another host in the cluster. In this case, another host must be able to take ownership of that VM, so a method to timeout the previous lock must exist. By default, a host will make 3 polling attempts (defined by NFS.LockRenewMaxFailureNumber) at 10 seconds intervals (defined by NFS.DiskFileLockUpdateFreq) to update the file lock. Each lock update attempt has a 5 seconds timeout (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 lock conflict. It then takes 3 polling attempts with 10 seconds intervals for the competing host to declare that the lock has expired and 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.

The author is not aware of any storage 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, it can result in some very undesirable behavior.

TCP Connections per IP Address

SunRPC.MaxConnPerIP defines the maximum number of unique TCP connections that can be opened for a given IP address. Currently VMware set this advanced setting to 4 although a maximum of 128 TCP connections is supported.

If the number of mounts to an IP address is more than SunRPC.MaxConnPerIP, existing connections are shared by different mountd.

Even with the maximum, the existing TCP connections need to be shared to mount 256 volumes. For example, if there are 4 IP addresses from which mounts are exposed, then there will be 32 connections per IP. This will give us a total of 4*32=128 TCP connections. If each mount is exporting 64 volumes then the connection to these 64 volumes will be shared among the 32 TCP connections. With this setting we can have 4*64=256 volumes.

One caveat with sharing connections however is that when one of the NFS volumes runs out of space, other NFS volumes that share the same Remote Procedure Calls (RPC) client might also report no space errors when they still have adequate space. If you have only a few NFS volumes, you can bump the SunRPC.MaxConnPerIP value to 128 to avoid connection sharing.

Security Considerations

This section provides details about the security concerns and considerations.

VMware vSphere implementation of NFS supports NFS version 3 in TCP. Storage traffic is transmitted as clear text 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 leverage a private VLAN.

Another security concern is that the ESXi host must mount the NFS server with root access. This raises some concerns about hackers getting access to the NFS server. To address the concern, it is best practice to use of either dedicated LAN or VLAN to provide protection and isolation.

Encryption in NFS version 4.1

Kerberos support with NFS version 4.1 alleviates some of the above security concerns. With the introduction of Kerberos, it is no longer necessary to have root access.

Enhancements to Kerberos for NFS version 4.1 in vSphere 6.5 include adding AES encryption support. AES256-CTS-HMAC-SHA1-96 and AES128-CTS-HMAC-SHA1-96 are both supported. The DES-CBC-MD5 encryption type is not supported in NFSv4.1 in vSphere 6.5.

As mentioned, vSphere 6.5 also introduces a new Kerberos integrity checking mechanism for NFS v4.1, called SEC_KRB5I. This feature performs integrity checking of NFS operations using secure checksum to prevent data manipulation.

Interoperability Considerations

This section covers features which are in some way related to storage, and NFS storage in particular.

In this section, features which are in some way related to storage, and NFS storage in particular, are discussed. While many of my regular readers will be well versed in most of these technologies, I'm hoping there will still be some items of interest. Most of the interoperability features are tried and tested with NFS, but I will try to highlight areas that might be cause for additional consideration.

VMware Docs: NFS Protocols and ESXi

Comparing Versions of NFS Clients

The following table lists capabilities that the NFS version 3 and 4.1 support.

Characteristics	NFS version 3	NFS version 4.1
Security mechanisms	AUTH_SYS	AUTH_SYS and Kerberos (krb5 and krb5i)
Encryption algorithms with Kerberos	N/A	AES256-CTS-HMAC-SHA1-96 and AES128-CTS-HMAC-SHA1-96
Multipathing	Not supported	Supported through the session trunking
Locking mechanisms	Propriety client-side locking	Server-side locking
Hardware acceleration	Supported	Supported
Thick virtual disks	Supported	Supported
IPv6	Supported	Supported for AUTH_SYS and Kerberos
ISO images presented as CD-ROMs to virtual machines	Supported	Supported
Virtual machine snapshots	Supported	Supported
Virtual machines with virtual disks greater than 2 TB	Supported	Supported

NFS Protocols and vSphere Solutions

The following table lists major vSphere solutions that NFS versions support.

vSphere Features	NFS version 3	NFS version 4.1
vMotion and Storage vMotion	Yes	Yes
High Availability (HA)	Yes	Yes
Fault Tolerance (FT)	Yes	Yes
Distributed Resource Scheduler (DRS)	Yes	Yes
Host Profiles	Yes	Yes
Storage DRS	Yes	**No**
Storage I/O Control	Yes	**No**
Site Recovery Manager	Yes	**No**
Virtual Volumes	Yes	Yes
vSphere Replication	Yes	Yes
vRealize Operations Manager	Yes	Yes

Storage I/O Control

The whole point of Storage I/O Control (SIOC) is to prevent a single virtual machine (VM) residing on one ESXi host from consuming more than its fair share of bandwidth on a datastore that it shares with other VMs which reside on other ESXi hosts.

Historically, we have had a feature called 'disk shares' which can be setup on a per ESXi host basis. This will work quite well for all VMs residing on the same ESXi host sharing the same datastore (i.e. local disk). However, this could not be used as a fairness mechanism for VMs from different ESXi hosts sharing the same datastore. This is what Storage I/O Control does for us. SIOC will modify the I/O queues on various ESXi hosts to ensure that VMs which have a higher priority get more queue entries that those VMs which have a lower priority, allowing these higher priority VMs 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 only triggered when the latency value on the datastore rises above a pre-defined threshold.

SIOC was first introduced for block storage back in vSphere 4.1. It was introduced for NFS datastores back in vSphere 5.0. If you have a group of VMs sharing the same datastore spread across multiple ESXi hosts, and you want to avoid a single VMs I/O impacting the I/O (and thus performance) of other VMs, you should certainly consider using SIOC. With SIOC you can set shares to reflect the priority of VMs, but you can also implement an IOPS limit per VM. This means that you can limit the number of IOPS that a single VM can do to a shared datastore.

This is an enterprise+ feature.

Network I/O Control

The Network I/O Control (NIOC) feature ensure that when the same NICs are used for multiple traffic types (e.g. 10Gb NICs), the NFS traffic not impacted by other traffic types on the same NICs. It works by setting priority and bandwidth using priority tags in TCP/IP packets. With 10Gb networks, this feature can be very useful as you will typically be sharing one pipe with multiple other traffic types. With 1Gb, the likelihood is that you have dedicated the pipe solely to NFS traffic. The point to note is that Network I/O Control is a congestion driven. If there is no congestion, any traffic type can consume as much bandwidth as it needs. NIOC only kicks in when there are different traffic types competing for bandwidth.

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

Not only that, but one can also set the priority of different VM traffic. So if certain VM traffic is important to you, these VMs can be

grouped into one virtual machine port group while lower priority VMs can be placed into another Virtual Machine port group. NIOC can now be used to prioritize VM traffic and ensure that the high priority VMs get more bandwidth when there is competition for bandwidth on the pipe.

SIOC and NIOC can co-exist and in fact complement one another.

This is an enterprise + feature and is only available on vSphere Distributed Switches

Storage DRS

Storage DRS, introduced in vSphere 5.0, fully supports NFS datastores. When you enable Storage DRS on a datastore cluster (group of datastores), balancing based on space usage is automatically configured. The threshold is set to 80% but can be modified if you so wish. What this means is that if space on a particular datastore is utilized 80% or more, Storage DRS will try to move VMs to other datastores using **Storage vMotion** to bring this usage value back down below 80%. The usage statistics of the datastores are checked on an ongoing basis.

If the cluster is set to **automatic** mode of operation, Storage DRS will use Storage vMotion to automatically migrate VMs to other datastores in the datastore cluster if the threshold is exceed. If the cluster is set to **manual** , the administrator will be given a set of recommendations to apply. Storage DRS will provide the best recommendations to balance the space usage of a datastores. As before, once you apply the recommendations, Storage vMotion will be used to move one or more VMs between datastores in the same datastore cluster.

Another feature of Storage DRS is its ability to balance VMs across datastores in the datastore cluster based on I/O metrics, specifically based on **latency** .

Storage DRS uses **Storage I/O Control** (SIOC) to evaluate datastore capabilities & capture latency information regarding all the datastores in the datastore cluster. As mentioned earlier. SIOC's purpose is to ensure that no single VM uses all the bandwidth of a particular datastore, and it modifies the queue depth to the datastores on each ESXi host to achieve this.

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

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 Storage DRS. If the latency value for a particular datastore is above the threshold value (default 15ms) for a significant percentage of time over an observation period (default 16 hours), Storage DRS will try to rebalance the VMs across the datastores in the datastore cluster so that the latency value returns below the threshold. This may involve a single or multiple Storage vMotion operations. In fact, even if Storage DRS is unable to bring the latency below the defined threshold value, it may still move VMs between datastores to balance the latency.

When starting out with evaluation Storage DRS, VMware makes the same recommendation that we made for DRS initially. The recommendation is to run Storage DRS in manual mode first, monitoring the recommendations that Storage DRS is surfacing and making sure that they make sense. After a period of time, if the recommendations make sense, and you build a comfort level with Storage DRS, consider switching it to automated mode.

There are a number of considerations when using Storage DRS with certain array features. VMware has already produced a very detailed white paper regarding the use of Storage DRS with array features like Tiered storage, Thin provisioning, deduplication, etc.

VAAI-NAS

Many NAS storage arrays now supports a number of vSphere API for Array Integration (VAAI) primitives. The purpose of this API is to allow the ESXi host to offload certain storage operations to the storage array rather than consuming resources on the ESXi host to do the same operation.

The first primitive we will discuss is **Full File Clone** , which allows you 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 only be deployed when the VM is powered off. Storage vMotion on NFS datastores continue to use the VMkernel software data mover.

The next primitive is called **Fast File Clone** . This is where the creation of linked clones is offloaded to the array. With the release of VMware View 5.1, this feature was supported as a tech preview. A future release of view (at the time of writing) is needed for full support of this primitive. With the release of vSphere 5.1 and with vCloud director 5.1, this primitive is fully supported for 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, we never had the ability to pre-allocate or zero out space for 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. 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 which guarantees that the space will be available. This is equivalent to VMFS lazyzeroedthick, and the blocks are zeroed on first write. However, selecting *Flat pre-initialized* also sends a Space Reserve NAS VAAI command, plus it does ESX-based zero writing to the VMDK – equivalent to a VMFS eagerzeroedthick. This means that it is a slow operation, and any writes are sent over the wire – they are not offloaded. For zeroing operations, it is safe to say that block arrays have an advantage.

As an aside, we just said that VAAI NAS Reserve Space allows you to create virtual disks in Thick Provision Lazy Zeroed (lazyzeroedthick) or Thick Provision Eager Zeroed (eagerzeroedthick) format on NFS datastores on arrays which 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 allows us to query how much space a VMDK actually consumed on an NFS datastore. For example, you might have created a 100GB thin VMDK, but actually consume only 25GB of space on the array. This was something vSphere previously never had any insight into. This was not a necessary feature to have for VMFS, since vSphere understands VMFS very well. But we did need something like this for NFS.

Remember that a VAAI NAS plugin is required from your respective storage array vendor for any of these primitives to work. The plugin must be installed on each ESXi host that wishes to leverage the VAAI NAS primitives.

What about the **Thin Provisioning** (TP) primitives? We did some work around these primitives in vSphere 5.0, such as raising an alarm when a TP volume reached 75% capacity at the backend, TP-Stun and of course the UNMAP primitive. However, these TP primitives are for SCSI only. The VAAI space threshold alarm is only supported 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 which should be propagated up the stack. This should induce a VM stun similar to how it happens with VAAI TP. This behavior does not need the VAAI-NAS plugin, and should work on all NFS datastores, whether the host has VAAI enabled or not. Finally, the UNMAP primitive is also for SCSI – the ‘dead space reclaiming’ is not an issue on NAS arrays.

One point to note with VAAI-NAS is that when NFS version 4.1 was initially launched with vSphere 6.0, there was no support for VAAI-NAS. Support for VAAI-NAS with NFS version 4.1 was introduced with the vSphere 6.5 release. Of course, this is not natively included in vSphere. Once again, a VAAI NAS plugin is required from your respective storage array vendor for any of these primitives to work with NFS version 4.1, and the plugin must be installed on each ESXi host that wishes to leverage the VAAI NAS primitives.

Site Recovery Manager/vSphere Replication

Site Recovery Manager (SRM) fully supports array based replication on NFS datastores. vSphere Replication fully supports replicating Virtual Machines which reside on NFS datastores.

With regards to best practice, I was reliably informed that one should consider storing the swap in a different directory when using a replicated NFS datastore in SRM. This will allow us to reduce the amount of replicated content which gets recreated on a failover. It also saves on us having to delete and recreate the “.vswp” files on the destination datastore after a failover. This has caused unnecessary delays during failover in earlier versions of SRM as file handles on NFS needed to expire on the “.vswp” files before they can be deleted and recreated. Some of this has been improved 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. Please reach out to your storage array vendor for guidance on whether or not this is a requirement.

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 splits writes to the source and destination datastore once a migration is initiated. This should mean speedier migrations since there is only a single copy operation now needed, unlike the recursive copy process used in previous versions which leveraged Change Block Tracking (CBT).

The one consideration and this has been called out already, is that Storage vMotion operations cannot be offloaded to the array with VAAI. All Storage vMotions on NFS datamovers will be done by the software datamover.

The only other considerations with Storage vMotion are relevant to both block & NAS, namely the configuration maximums. At the time of writing, the maximum number of concurrent Storage vMotion operations per ESXi host was 8, and the maximum number of Storage vMotion operations per NFS datastore was 2. This is to prevent any single datastore from being unnecessarily impacted by Storage vMotion operations. Note that a new enhancement in vSphere 5.1 allows up to 4 VMDKs belonging to the same VM to be migrated in parallel, so long as the VMDKs reside on different datastores.

Sizing Considerations

This section provides a brief about the sizing considerations and how it affects the NFS datastore.

Maximum volume Size/Sweet Spot

Many of you will have observed the following statement in the VMware Configuration Maximums Guide – *Contact your storage array vendor or NFS server vendor for information about the maximum NFS volume size.* The purpose of this first question was to see if there was a volume size which worked well which the partners agreed on. In fact, all partners responded saying that there was no performance gain or degradation depending on the volume size – customers could 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 TB, but consensus appears to be that the majority of NFS datastores are in the 10s of TB in size. The datastores sizes vary quite a bit from customer to customer.

Sizing of volumes is typically proportional to the number of VMs you wish to deploy, plus snapshots/changed blocks for backup purposes. Another consideration is the fact that many arrays now have deduplication and compression features, which will also reduce capacity requirements.

Recommended Block Size

From the vendors that I spoke to, this parameter is not tuneable 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 VMware's virtual disk format (VMDK). For those vendors who have it set to 8KB, the recommendation is to format the volumes in the Guest OS to a matching 8KB block size for optimal performance. The one vendor who did have a tuneable block size had different recommendations depending on the workload – 4KB for random workloads and 32KB for sequential workloads. Since having multiple virtual machines sharing an NFS datastore is going to result in random workloads, you probably would be using 4KB. In this area, it is best to speak to your storage array vendor to get vendor specific advice. The questions one should ask are:

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

RSIZE & WSIZE

I thought it useful just to add a note about the RSize and WSize parameters when discussing sizing. These parameters, defined at mount time, define the I/O chunk transfer sizes between the host and the target. These sizes are not tuneable. However, feedback from the partners suggest that the current size of 64KB meets their requirements. While some storage arrays can now support much larger WSize & RSize values (up to 1MB), there is still no way to change these settings in vSphere at the time of writing.

Recommended number of VMs per NFS datastore

This was again met with a pretty consistent response from all vendors. The number of virtual machines which can run on a single datastore is directly proportional to the infrastructure and the workloads running in the VMs, i.e. hundreds of low I/O VMs versus a few very intensive I/O VMs. Network congestion was an important factor. Since VMware still only supports NFS v3 over TCP/IP, one recommendation was to use more datastores presented over different interfaces so that traffic could 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 single datastore with lots of VMs, how long are you willing to wait while this is restored in the event of a failure. This is starting to become the major consideration in the 'how many VMs per datastore' debate.

Another consideration which a customer highlighted is the snapshot technology used by the backup product – is it using array based snapshots or is it using virtual machine snapshots. Performance considerations are needed if using virtual machine snapshots to concurrently capture point in time copies of VMs. In many cases, array based snapshots have less impact on the datastores and are more scalable when it comes to backups. There may be some array based limitations to take into considerations too – for instance, the number of snapshot copies of a VM that a customer wishes to maintain versus the number of snapshot copies an array can support. This would vary from vendor to vendor so best to check this configuration maximum with your storage array vendor.

From a vSphere snapshot perspective, refer to this KB article 1015180 for further details around snapshots and their usage. From KB article 1025279, VMs can support up to 32 snapshots in a chain, *but VMware recommends that you use only 2-3 snapshots in a chain, and to use no single snapshot for more than 24-72 hours.*

Additional Attributes of NFS Storage

This section details the additional attributes of NFS storage.

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 criterion might make one choice a better selection than the alternative.

Thin Provisioning

Virtual disks (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 purpose 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. As such, less allocation of VMFS volume storage space than is needed for the same set of virtual disks provisioned as thick format. Although the thin format virtual disks can also be provisioned on block-based VMFS datastores. VMware vCenter™ now provides support for both the creation of thin virtual disks as well as the monitoring the datastores that may be over-committed (i.e. that can trigger alarms for various thresholds of either space used approaching capacity or provisioned pace out pacing capacity by varying degrees of over commit).

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 have been able to 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 products that utilize vSphere Data Protection APIs (VADP). Another is placing agents in each Guest OS, or leveraging array based snapshot technology. However, the process of restoring the virtual machine gets to be a bit more complicated as there are often many levels of granularity that are requested for restore. Some recovery solutions might require the entire datastore to be restored while 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 on what level of granularity can be restored. With an array-based snapshot of a 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 open up a bit of a security risk, NFS does provide one of the most flexible and efficient restore from backup option available today. For this reason, NFS earns high marks for ease of backup and restore capability.

Summary of Best Practices

This section summarizes the best practices for running VMware vSphere on network attached storage.

To address the security concerns and provide optimal performance, VMware provides the following best practices:

Networking Settings

To isolate storage traffic from other networking traffic, it is considered best practice to use either dedicated switches or VLANs for your NFS and iSCSI ESX server traffic. The minimum NIC speed should be 1GbE. In VMware vSphere, use of 10GbE is supported. Best to look at the VMware HCL to confirm which models are supported.

It is important to not over-subscribe the network connection between the LAN switch and the storage array. The retransmitting of dropped packets can further degrade the performance of an already heavily congested network fabric.

Datastore Settings

The default setting for the maximum number of mount points/datastore an ESXi host can concurrently mount is eight, although the limit can be increased to 256 in the current vSphere release. If you increase max NFS mounts above the default setting of eight in previous versions of vSphere, make sure to refer to the vendor specific documentation around increasing Net.TcpipHeapSize if necessary.

Conclusion and About the Author

This section includes the conclusion and a few lines about the author.

Conclusion

In short, 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 very 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 deployment VMware vSphere on this storage protocol option.