Migrating from NFSv3 to NFSv4

April 2011

Table of Contents

Introduction

In April 2003, the Network File System (NFS) version 4 Protocol was ratified as an Internet standard, described in RFC-3530[1], which superseded NFS Version 3 (NFSv3, specified in RFC-1813[2]). Since the ratification of NFSv4, further advances have been made to the standard, notably NFSv4.1 (as described in RFC-5661[3], ratified in January 2010) that includes several new features such as parallel NFS (pNFS).

While there have been many advances and improvements to NFS, some IT organizations have elected to continue with NFSv3. While adequate for many purposes and a familiar and well understood protocol, NFSv3 has become increasingly difficult to justify. For example, NFSv3 makes promiscuous use of ports, something that is unsuitable for a variety of security reasons for use over a wide area network (WAN); plus increased server & client bandwidth demands and improved functionality of Network Attached Storage (NAS) arrays have outstripped NFSv3's ability to deliver high throughput.

NFSv4 includes features intended to enable its use in global wide area networks (WANs). These advantages include:

- Firewall-friendly single port operations
- Advanced and aggressive cache management features
- Internationalization support
- Replication and migration facilities
- Mandatory use of strong RPC security flavors that depend on cryptography, with support of access control that is compatible with both UNIX® and Windows®
- Use of character strings instead of integers to represent user and group identifiers

Many additional enhancements to NFSv4 are available with NFSv4.1, the latest revision to the standard. Although client and server implementations of NFSv4.1 are available, they are in early stages of implementation and adoption. However, to take advantage of them in the future, it is important to move to widely available NFSv4 servers and clients as a stepping-stone. NFSv4 is a mature and stable protocol with many advantages in its own right over its predecessors NFSv3 and NFSv2.

This paper focuses on an overview of the important issues that should be considered while migrating from an NFSv3 to NFSv4 environment. Throughout, there are practical suggestions (***Action:***) about what to look out for. The paper assumes that the reader is familiar with the features and major differences between NFSv3 and NFSv4. If you are unfamiliar with NFSv4, it's recommended that you refer to one of the excellent NFSv4 descriptions available on the leading vendors' websites.
A subsequent SNIA white paper will deal with implementing NFSv4.1.

---

[1] "Network File System (NFS) version 4 Protocol" http://www.ietf.org/rfc/rfc3530.txt
[2] "NFS Version 3 Protocol Specification" http://www.ietf.org/rfc/rfc1813.txt
[3] "Network File System (NFS) Version 4 Minor Version 1 Protocol" http://www.ietf.org/rfc/rfc5661.txt

NFSv3 to NFSv4 Considerations[4]

Pseudo Filesystem

On most operating systems, the name space describes the set of available files arranged in a hierarchy. When a system acts as a server to share files, it typically exports (or "shares") only a portion of its name space, excluding perhaps local administration and temporary directories. Consider a file server that exports the following directories:

```
/vol/vol0
/vol/vol1
/backup/archive
```

The server provides a single view of the exported file systems to the client as shown in Figure 1.

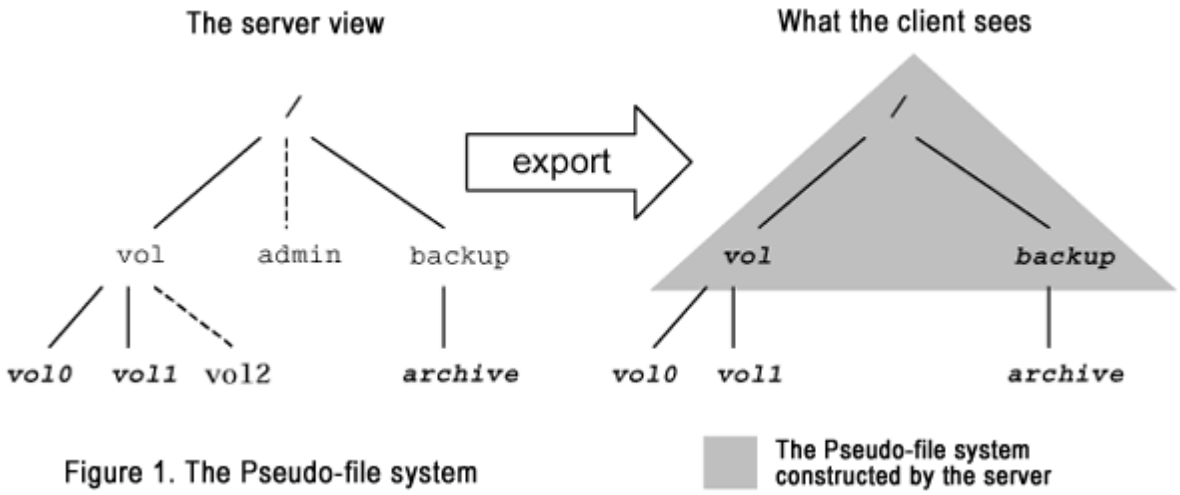

Figure 1. The Pseudo-file system

In NFSv4, a server's shared name space is a single hierarchy. In the example illustrated in Figure 1, the export list and the server hierarchy is disjoint, and not connected. When a server chooses to export a disjoint portion of its name space, the server creates a pseudo-file system (the area shown in grey) to bridge the unexported portions of the name space allowing a client to reach the export points from the single common root. A pseudo-file system is a structure containing only directories, created by the server, having a unique filesystem id (*fsid*) that allows a client to browse the hierarchy of exported file systems.

The flexibility of the pseudo filesystem as presented by the server can be used to limit the parts of the name space that the client can see, a powerful feature that can be used to considerable advantage.

[4] https://wiki.archlinux.org/index.php/NFSv4 contains additional information for installing and configuring NFSv4 on Linux

For example, to contrast the differences between NFSv3 and NFSv4 name spaces, consider the mount of the root filesystem `/` in Figure 1. A mount of `/` over NFSv3 allows the client to list the contents of `/vol/vol2` as the *fsid* for `/` and `/vol/vol2` is the same. An NFSv4 mount of `/` over NFSv4 generates a pseudo *fsid*. As `/vol/vol2` has not been exported and the pseudo filesystem does not contain it, it will not be visible. An explicit mount of `vol/vol2` will be required.

- ***Action:*** Consider using the flexibility of pseudo-filesystems to permit easier migration from NFSv3 directory structures to NFSv4, without being overly concerned as to the server directory hierarchy and layout.
- ***Action:*** However, if there are applications that traverse the filesystem structure or assume the entire filesystem is visible, caution should be exercised before moving to NFSv4 to understand the impact presenting a pseudo filesystem, especially when converting NFSv3 mounts of `/` to NFSv4.

TCP only; no UDP support

Although NFSv3 supports both TCP and UDP, UDP is employed for applications that support it because it is perceived to be lightweight and faster in comparison with TCP. The downside of UDP is that it's an unreliable protocol. There is no guarantee that the datagrams will be delivered in any given order to the destination host -- or even delivered at all -- so applications must be specifically designed to handle missing, duplicate or incorrectly ordered data. UDP is also not a good network citizen; there is no concept of congestion or flow control, nor the ability to apply quality of service (QoS) criteria.

The NFSv4 specification requires that any transport used provides congestion control. The easiest way to do this is via TCP. By using TCP, NFSv4 clients and servers are able to adapt to known frequent spikes in unreliability on the Internet; and retransmission is managed in the transport layer instead of in the application layer, greatly simplifying applications and their management on a shared network.

NFSv4 also introduces strict rules about retries over TCP in contrast to the complete lack of rules in NFSv3 for retries over TCP. As a result, if NFSv3 clients have timeouts that are too short, NFSv3 servers may drop requests. NFSv4 relies on the timers that are built into the connection-oriented transport.

- ***Action:*** review applications and existing mounts that use UDP as a transmission protocol for NFSv3. Evaluate whether TCP can be used on those applications' mount points.
- ***Action:*** review existing NFSv3 clients using TCP to ensure timeout handling is compatible with the chosen NFSv4 client and its handling of TCP retries.

Internationalization support; UTF-8

NFSv4 uses UTF-8 for file names, directories, symlinks and user and group identifiers. As UTF-8 is backwards compatible with 7 bit encoded ASCII, any names that are 7 bit ASCII will continue to work. However, pre-existing names that contain 8 bit characters will be misinterpreted by NFSv4 as UTF-8 multibyte characters, which may result in errors such as not finding files.

For example, an NFSv3 file created with the name `René` contains an 8 bit ASCII character in the last position. NFSv4 will assume that the `é` indicates a multibyte UTF-8 encoding, which will lead to unexpected results.

- ***Action:*** review existing NFSv3 names to ensure that they are 7 bit ASCII clean.

Network Ports

To access an NFS server, an NFSv3 client must contact the server's *portmapper* to find the port of the `mountd` server. It then contacts the mount server to get an initial file handle, and again contacts the *portmapper* to get the port of the NFS server. Finally, the client can access the NFS server.

This creates problems for using NFS through firewalls, because firewalls typically filter traffic based on well known port numbers. If the client is inside a firewalled network, and the server is outside the network, the firewall needs to know what ports the *portmapper*, `mountd` and `nfsd` servers are listening on. The mount server can listen on any port, so telling the firewall what port to permit is not practical. While the NFS server usually listens on port 2049, sometimes it does not. While the *portmapper* always listens on the same port (111), many firewall administrators, out of excessive caution, block requests to port 111 from inside the firewalled network to servers outside the network. As a result, NFSv3 is not practical to use through firewalls.

NFSv4 uses a single port number by mandating the server will listen on port 2049. There are no "auxiliary" protocols like `statd`, `lockd` and `mountd` required (but see the section *Mounts and automounter* below), as the mounting and locking protocols have been incorporated into the NFSv4 protocol. This means that NFSv4 clients do not need to contact the *portmapper*, and do not need to access services on floating ports.

As NFSv4 uses a single TCP connection with a well-defined destination TCP port, it traverses firewalls and network address translation (NAT) devices with ease, and makes firewall configuration as simple as configuration for HTTP.

- ***Action:*** review and consider opening port 2049 for TCP on firewalls

Mounts and automounter

The automounter daemons and the utilities on different flavors of UNIX and Linux are capable of identifying different NFS versions. Be sure to check with your specific vendor client implementation. For example, Red Hat Enterprise Linux 6 (RHEL6) with its later kernel versions is able to understand the `vers=4` mount option, and the default mount version in newer versions of *nfs-utils* can be specified in `/etc/nfsmount.conf`. Some UNIX clients detect NFSv4 and subsequently mount the filesystem over NFSv4 by default; to maintain mounting over NFSv3 `vers=3` or its equivalent may need to be explicitly specified.

For older versions of UNIX and Linux, the `auto.home`, `auto.misc` or `auto.*` files can be modified to ensure that NFSv4 is the default filesystem:

```
-fstype=nfs4, rw, proto=tcp, port=2049
```

- ***Action:*** When using an automounter, caution should be exercised before moving to NFSv4 to understand the impact of the pseudo filesystem, especially when converting NFSv3 mounts of `/` to NFSv4
- ***Action:*** Review mount parameters and UNIX/Linux version documentation to ensure that the correct mount and automounter parameters are changed to get the NFS version desired

However, using the automounter will require at least port 111 to be permitted through any firewall between server and client, as it uses the *portmapper* (see the *Network Ports* section). This is undesirable if you are extending the use of NFSv4 beyond traditional NFSv3 environments, so consider using Solaris' or Linux's (e.g. available in RHEL5 and RHEL6) *"mirror mount"* facility which enhances the behavior of the NFSv4 client. It creates a new mountpoint whenever it detects that a directory's *fsid* differs from that of its parent and automatically mounts filesystems when they are encountered at the NFSv4 server[5].

This enhancement does not require the use of the automounter and therefore does not rely on the content or propagation of automounter maps, the availability of NFSv3 services such as `mountd`, or opening firewall ports beyond the single port 2049 required for NFSv4.

- ***Action****:* Investigate *"mirror mounts"* for extending NFSv4 out to less secure environments beyond a firewall

Security

NFSv4 includes ACL support based on the Microsoft Windows®NT model (named strings), not the POSIX ACL model (32 bit integers).

There is much confusion over NFSv4 security. Many believe that NFSv4 requires the use of strong security. The specification simply states that *implementation* of strong RPC security by servers and clients is mandatory, not the *use* of strong RPC security. This misunderstanding may explain the reluctance of users from migrating to NFSv4 due to the additional work in implementing or modifying existing Kerberos security.

NFSv4 can be implemented without implementing Kerberos security.

Implementing security without Kerberos

NFSv3 represents users and groups via 32 bit integers. The NFS protocol uses user and group identifiers in the results of a get attribute (GETATTR) operation and in the arguments of a set attribute (SETATTR) operation.

[5] The Linux "mirror mounting" feature is not specific to NFSv4; NFSv2 and NFSv3 mounts can be configured to act in the same way.

NFSv4 represents users and groups in the form:

`user@domain` or `group@domain`

where domain represents a registered DNS domain, or a sub-domain of a registered domain. One issue with using string names, instead of integers, is that many UNIX systems and NAS arrays will still be using integers in the underlying file systems stored on disk.

As is the case with NFSv3, the security of all data in the NFSv4 share depends on the integrity of all clients and the security of the network connections. However, changes are required to ensure that UIDs and GUIDs are shown correctly; with an incorrect or incomplete configuration, UID and GUID will commonly display the string `nobody`.

NFSv4 requires all 32 bit integers, including NFSv3 UIDs and GUIDs, to be converted to all numeric strings with a valid domain name. To undertake the translation, integers can either be converted directly to strings, or by using the `nsswitch` translation service. For translation, the client must be running `idmapd`[6], and `/etc/idmapd.conf` should be modified to point to a default domain and specify the use of the translation service `nsswitch`. This maps integers to string names and back, and is controlled by the `/etc/nsswitch.conf` file. Using integers to represent users and groups requires that every client and server that might connect to each other to agree on user and group translations.

Implementing security with Kerberos

Security mapping using integers to represent users and groups is impractical across the WAN, and problematic for enterprises that export file systems with differing local representations, employ multiple security domains or employ multiple name translation services.

Employing the superior security available in NFSv4 requires using Kerberos, which divides user communities into realms. Each realm has an administrator responsible for maintaining a database of principals (users), one master Key Distribution Center (KDC), and one or more slave KDCs that give users tickets to access services on specific hosts in a realm. Users (`principal@realm`) in one realm can access services in another realm, but it requires the cooperation of the administrators in each realm to configure trust relationships by exchanging per-realm keys. Hierarchical organization and authentication of realms can be used to reduce the number of inter-realm relationships.

Virtually every exported file system uses numbers – for example, a 32bit UID/GID or 128bit Security Identifier (SID) – to represent users and groups on disk. All clients and servers in a Kerberos based NFS file system (no matter how many Kerberos realms are supported via cross-trust mechanisms) need to agree on ID to name translations. Using unique global string names (such as `name@domain` or `principal@realm`) to represent users and groups allows interoperability; for example, a server exporting a 128bit SID based file system, and a server exporting a POSIX 32bit UID/GID based file

[6] For examples, see for Ubuntu Linux; https://help.ubuntu.com/community/NFSv4Howto; and SUSE Linux Enterprise; http://www.novell.com/support/dynamickc.do?cmd=show&forward=nonthreadedKC&docType=kc&externalId=7005060&sliceId=1

system can then give access and set ACLs for the same user consistently. The use of a correct `user@domain` or `group@domain` string is required; integers converted directly to strings (NFSv3 32bit integer UIDs and GUIDs) are explicitly denied access.

The NFSv3 and NFSv4 security models are not compatible with each other. Although storage systems may support both NFSv3 and NFSv4 clients, be aware that there may be compatibility issues with ACLs. For example, they may be enforced but not visible to the NFSv3 client.

- ***Action:*** Review security requirements on NFSv4 filesystems; consider using Kerberos for robust security
- ***Action:*** If using no security, ensure UID and GUID mapping and translation is uniformly implemented across the enterprise
- ***Action:*** If using Kerberos, ensure it is installed and operating correctly and ensure translations between IDs and global names is uniformly implemented across the enterprise

Conclusion

Unlike earlier versions of NFS, the NFSv4 protocol integrates file locking, strong security, operation coalescing, and delegation capabilities to enhance client performance for narrow data sharing applications on high-bandwidth networks. Locking and delegation make NFS stateful, but simplicity of design is retained through well-defined recovery semantics in the face of client and server failures and network partitions.

With careful planning, migration to NFSv4 from prior versions can be accomplished without modification to applications or the supporting operational infrastructure, such as applications, storage servers, backup jobs and so on. Users are encouraged to move from NFSv3 to NFSv4 in anticipation of generally available NFSv4.1/pNFS servers and clients, which will provide even more functionality such as wide striping of data to enhance performance.

About the Author

Alex McDonald, Co-Chair SNIA ESF NFS SIG
Industry Forums & Standards, NetApp

Alex joined NetApp in 2005, after more than 30 years in a variety of roles with some of the best known names in the software industry (Legent, Oracle, BMC and others). With a background in software development, support, sales and a period as an independent consultant, Alex is now part of NetApp's Office of the CTO that supports industry activities and promotes technology & standards based solutions, and is co-chair of the SNIA NFS Special Interest Group.

About the Ethernet Storage Forum

The Ethernet Storage Forum (ESF) is the marketing organization within the Storage Networking Industry Association (SNIA) focused on Ethernet-connected storage networking solutions. Through the creation of vendor-neutral educational materials, ESF thought leaders leverage SNIA and Industry events and end-user outreach programs to drive market awareness and adoption of Ethernet-connected storage networking technologies, worldwide. For more information, visit www.snia.org/forums/esf.

About the SNIA

The Storage Networking Industry Association (SNIA) is a not-for-profit global organization, made up of some 400 member companies and 7,000 individuals spanning virtually the entire storage industry. SNIA's mission is to lead the storage industry worldwide in developing and promoting standards, technologies, and educational services to empower organizations in the management of information. To this end, the SNIA is uniquely committed to delivering standards, education, and services that will propel open storage networking solutions into the broader market. For additional information, visit the SNIA web site at www.snia.org.