An Updated Overview of NFSv4

NFSv4.0, NFSv4.1, pNFS, and NFSv4.2

October 2015

Table of Contents

Figures

Tables

Introduction

This white paper was first published in June 2012 as *An Overview of NFSv4*. In the intervening period, the IETF NFS workgroup has been working on new features & functionality, and is close to ratifying NFSv4.2. This update includes refreshes of all of the material from the original June 2012 publication, and adds new material where appropriate.

During the last few years, NFSv4 has become the Network File System (NFS) version of choice for many users, and many are making the transition from NFSv3 to NFSv4. This is attributable to several reasons, not the least of which is the relatively straightforward transition from NFSv3. And lately we've seen new clients of NFSv4 servers beyond the standard Linux client, including support in VMware's vSphere for virtual machine datastores. A decade's worth of experience with NFSv4 should encourage everyone that this technology is stable, reliable and that it works well.

In this white paper, we explain how NFSv4 is better suited to a wide range of datacenter and high performance compute (HPC) uses than its predecessor NFSv3, as well as providing resources for migrating from v3 to v4. And, most importantly, we make the argument that users should, at the very least, be evaluating and deploying NFSv4 for use in new projects; and ideally, should be using it wholesale in their existing environments.

The Background of NFSv4

NFSv2 and its popular successor NFSv3 (specified in RFC-1813[1], but never an Internet standard) was first released in 1995 by Sun. It has proved a popular and robust protocol over the 20 years it has been in use, and with wide adoption it soon eclipsed some of the early competitive UNIX-based file system protocols such as DFS and AFS.

NFSv3 was extensively adopted by storage vendors and OS implementers beyond Sun's Solaris; it was available on an extensive list of systems, including IBM's AIX, HP's HP-UX, Linux and FreeBSD. Even non-UNIX systems adopted NFSv3; Mac OS, OpenVMS, Microsoft Windows, Novell NetWare, and IBM's AS/400 systems. In recognition of the advantages of interoperability and standardization, Sun relinquished control of future NFS standards work, and work leading to NFSv4 was by agreement between Sun and the Internet Society (ISOC), and is undertaken under the auspices of the Internet Engineering Task Force (IETF).

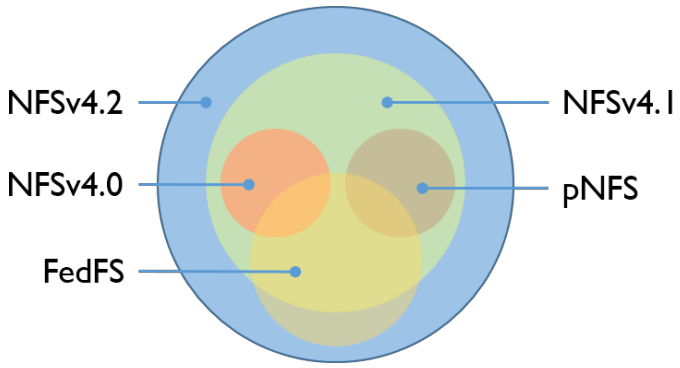


Figure 1; Relationship between NFS Versions

[1] NFSv3 specification: http://tools.ietf.org/html/rfc1813. Other IETF RFCs mentioned in the text can be found at the same site.

An Updated Overview of NFSv4

In April 2003, the Network File System (NFS) version 4 Protocol[2] was ratified as an Internet standard, described in RFC-3530, which superseded NFSv3. This was the first open file system and networking protocol from the IETF. NFSv4 introduces the concept of state to ameliorate some of the less desirable features of NFSv3, and other enhancements to improve usability, management and performance.

But shortly following the release of RFC-3530, an Internet draft written by Garth Gibson and Peter Corbett outlined several problems with NFSv4[3]; specifically, that of limited bandwidth and scalability, since NFSv4 like NFSv3 requires that access is to a single server. NFSv4.1 (as described in RFC-5661[4], ratified in January 2010) was developed to overcome these limitations, and new features such as parallel NFS (pNFS) were standardized to address these issues.

Additionally, practical experience with RFC-3530, the original NFSv4 specification, led to a number of proposed clarifications and changes, and all these were consolidated and published in a new specification as RFC-7530[5] dated March 2015.

Now NFSv4.2 is moving towards ratification[6]. In a change to the original IETF NFSv4 development work, where each revision took a significant amount of time to develop and ratify, the workgroup charter was modified to ensure that there would be no large standards documents that took years to develop, such as RFC-5661, and that additions to the standard would be an on-going yearly process. With these changes in the processes leading to standardization, features that will be ratified in NFSv4.2 (expected in 2015) are available from many vendors and suppliers today. These relationships are shown in Figure 1**Error! Not a valid bookmark self-reference.**.

FedFS[7] (a "federated file system") is a currently unratified standards proposal that provides a set of open protocols that permit the construction of a scalable, federated file system namespace accessible to unmodified NFSv4 clients. Work is in progress; again, some of the features expected in FedFS are already available.

Adoption of NFSv4

While there have been many advances and improvements to NFS, many users have elected to continue with NFSv3. In June 2012, it was correct to state that

> "NFSv4 is a mature and stable protocol with many advantages in its own right over its predecessors NFSv3 and NFSv2, ***yet adoption remains slow***."

---

[2] By informal convention, the NFSv4 specification is called NFSv4.0. This paper uses NFSv4 to mean all of the minor versions from NFSv4.0 thru NFSv4.2; specific minor version numbers are used to indicate features only available in that version or greater.
[3] The "pNFS Problem Statement": http://tools.ietf.org/html/draft-gibson-pnfs-problem-statement-01
[4] RFC-5661 Network File System (NFS) Version 4 Minor Version 1 Protocol is at https://tools.ietf.org/html/rfc5661
[5] RFC-7530 Network File System (NFS) Version 4 Protocol is at https://tools.ietf.org/html/rfc7530
[6] NFSv4.2 proposed specification: http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-38.txt; the draft as of April 2015
[7] An overview of FedFS: http://people.redhat.com/steved/Bakeathon-2010/fedfs_fast10_bof.pdf

That is no longer true; the industry is seeing an increased pace of adoption. The criticisms of NFSv3 are more relevant now than they were in 2012. Although NFSv3 is adequate for some purposes and is a familiar and well understood protocol, the demands being placed on storage by exponentially increasing data and compute growth has finally caught up with it. NFSv3 has become increasingly difficult to deploy, manage and scale.

So What's The Problem With NFSv3?

In essence, NFSv3 suffers from problems associated with statelessness. While some protocols such as HTTP and other RESTful APIs[8] see benefit from not associating state with transactions – it considerably simplifies application development if no transaction from client to server depends on another transaction – in the NFS case, statelessness has led, amongst other downsides, to performance and lock management issues.

NFSv4.1 and parallel NFS (pNFS) address well-known NFSv3 "workarounds" that are used to obtain high bandwidth access; users that employ (usually very complicated) NFSv3 automounter maps and modify them to manage load balancing should find pNFS provides comparable performance that is significantly easier to manage.

Extending the use of NFS across the WAN is difficult with NFSv3. Firewalls typically filter traffic based on well-known port numbers, but if the NFSv3 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. As a result of this promiscuous use of ports, the multiplicity of "moving parts" and a justifiable wariness on the part of network administrators to punch random holes through firewalls, NFSv3 is not practical to use in a WAN environment. By contrast, NFSv4 integrates many of these functions, and mandates that all traffic (now exclusively TCP) uses the single well-known port 2049.

One of the most annoying NFSv3 "features" has been its handling of locks. Although NFSv3 is stateless, the essential addition of lock management (NLM) to prevent file corruption by competing clients means NFSv3 application recovery is slowed considerably. Very often stale locks have to be manually released, and the lock management is handled external to the protocol. NFSv4's built-in lock leasing, lock timeouts, and client-server negotiation on recovery simplifies management considerably.

In a change from NFSv3, these locking and delegation features make NFSv4 stateful, but the simplicity of the original design is retained through well-defined recovery semantics in the face of client and server failures and network partitions. These are just some of the benefits that make NFSv4.1 desirable as a modern datacenter protocol, and for use in HPC, database and highly virtualized applications.

NFSv4.2 continues that work. A major goal of the design of NFSv4.2 was to take common local file system features and offer them remotely; hence facilities such as space reservations, hole punching and

[8] HTTP and RESTful APIs explained in overview: https://en.wikipedia.org/wiki/Representational_state_transfer

I/O advise. NFSv4.2 extends the protocol to provide valuable functionality that can't be accomplished at all by NFSv3.

The Advantages of NFSv4.1 & NFSv4.2

In the following sections, NFSv4 features are presented in roughly the order of the relevant IETF standards; that is, NFSv4.0 features, followed by NFSv4.1 and NFSv4.2. Remember that clients and servers are not required to implement all of these features; while some are mandatory (eg TCP for transport, Kerberos based security), many are optional (eg pNFS and hole punching).

Pseudo File system

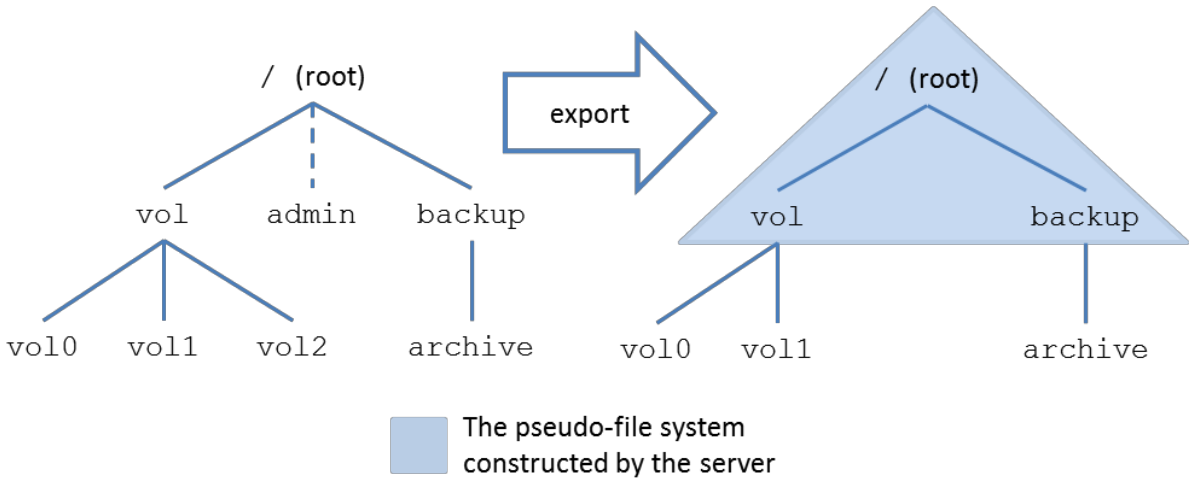


Figure 2; Pseudo File System

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 2.

In NFSv4, a server's shared name space is a single hierarchy. In the example illustrated in Figure 2, 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 file system id (*fsid*) that allows a client to browse the hierarchy of exported file systems.

The flexibility of the pseudo file system 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. For example, to contrast the differences between NFSv3 and NFSv4 name spaces, consider the mount of the root file system `/` 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 file system does not contain it, it will not be visible. An explicit mount of `vol/vol2` will be required.

The flexibility of pseudo-file systems permits easier migration from NFSv3 directory structures to NFSv4, without being overly concerned as to the server directory hierarchy and layout. However, if there are applications that traverse the file system structure or assume the entire file system 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 for Transport

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, or the ability to apply quality of service (QoS) criteria.

The NFSv4.0 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.

NFSv4.0 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. NFSv4.0 relies on the timers that are built into the connection-oriented transport.

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 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.

Mounts and Automounter

The automounter daemons and the utilities on different flavors of UNIX and Linux are capable of identifying different NFS versions. 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.

This is undesirable if you are extending the use of NFSv4 beyond traditional NFSv3 environments, so in preference the widely available *"mirror mount"* facility can be used. It enhances the behavior of the NFSv4 client by creating 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[9].

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.

Internationalization Support; UTF-8

In a welcome recognition that the ASCII character set no longer provides the descriptive capabilities demanded by languages with larger alphabets or those that use an extensive range of non-Roman glyphs, 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.

Compound RPCs

Latency in a WAN is a perennial issue, and is very often measured in tenths of a second to seconds. NFS uses RPC to undertake all its communication with the server, and although the payload is normally small, meta-data operations are largely synchronous and serialized. Operations such as file

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

lookup (LOOKUP), the fetching of attributes (GETATTR) and so on, make up the largest percentage by count of the average workload (see Table 1).

NFSv3 Operation	SPECsfs2008
GETATTR	26%
LOOKUP	24%
READ	18%
ACCESS	11%
WRITE	10%
SETATTR	4%
Others	7%

Table 1; SPECsfs2008 percentages for NFSv3 operations[10]

This mix of a typical NFS set of RPC calls in versions prior to NFSv4 requires each RPC call is a separate transaction over the wire. NFSv4 avoids the expense of single RPC requests and the attendant latency issues and allows these calls to be bundled together. For instance, a lookup, open, read and close can be sent once over the wire, and the server can execute the entire compound call as a single entity. The effect is to reduce latency considerably for multiple operations.

Delegations

Servers are employing ever more quantities of RAM and non-volatile RAM technologies like flash, and very large 16TB caches are not uncommon. Applications running over NFSv3 can't take advantage of these caches unless they have specific application support. With increasing WAN latencies, doing every IO over the wire introduces significant delay.

NFSv4 allows the server to delegate certain responsibilities to the client, a feature that allows caching locally where the data is being accessed. Once delegated, the client can act on the file locally with the guarantee that no other client has a conflicting need for the file; it allows the application to have locking, reading and writing requests serviced on the application server without any further communication with the NFS server. To prevent deadlocking conditions, the server can recall the delegation via an asynchronous callback to the client should there be a conflicting request for access to the file from a different client.

Migration, Replicas and Referrals

For broader use within a datacenter, and in support of high availability applications such as databases and virtual environments, copying data for backup and disaster recovery purposes, or the ability to migrate it to provide VM location independence are essential. NFSv4 provides facilities for both transparent replication and migration of data, and the client is responsible for ensuring that the application is unaware of these activities. An NFSv4 referral allows servers to redirect clients from this server's namespace to another server; it allows the building of a global namespace while maintaining the data on discrete and separate servers.

[10] http://www.spec.org/sfs2008/docs/usersguide.html#_Toc191888936 gives a typical mix of RPC calls from NFSv3.

Sessions

NFSv4.1 has brought two major pieces of functionality: sessions and pNFS.

Sessions bring the advantages of correctness and simplicity to NFS semantics. In order to improve the correctness of NFSv4, NFSv4.1 sessions introduce "exactly-once" semantics. This is important for supporting operations that were non-idempotent (that is, operations that if executed twice or more return different results, for example the file RENAME operation). Making such operations idempotent is a significant practical problem when the file system and the storage are separated by a potentially unreliable communications link, as is the case with NFS.

Servers maintain one or more session states in agreement with the client; a session maintains the server's state relative to the connections belonging to a client. Clients can be assured that their requests to the server have been executed, and that they will never be executed more than once. Sessions extend the idea of NFSv4 delegations, which introduced server-initiated asynchronous callbacks; clients can initiate session requests for connections to the server. For WAN based systems, this simplifies operations through firewalls.

Parallel NFS (pNFS) and Layouts

Parallel NFS (pNFS) represents a major step forward in the development of NFS. Ratified in January 2010 and described in RFC-5661, pNFS depends on the NFS client understanding how a clustered file system stripes and manages data. It's not an attribute of the data, but an arrangement between the server and the client, so data can still be accessed via non-pNFS and other file access protocols.

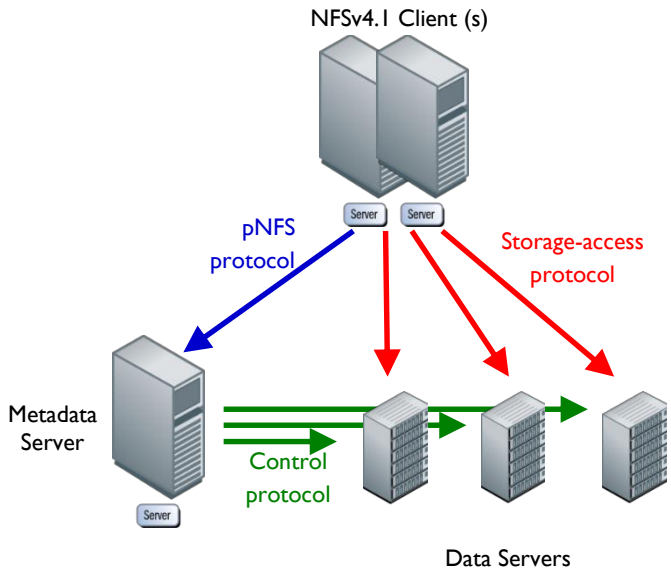
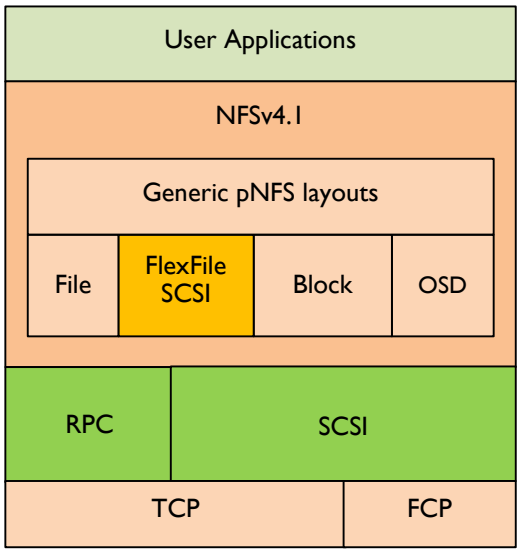


Figure 3; pNFS Conceptual Data Flow

pNFS benefits workloads with many small files, or very large files, especially those run on compute clusters requiring simultaneous, parallel access to data.

Clients request information about data layout from a Metadata Server (MDS), and get returned layouts that describe the location of the data. (Although often shown as a separate server, the MDS may or may not be standalone nodes in the storage system depending on a particular storage vendor's hardware architecture.) The data may be on many data servers, and is accessed directly by the client over multiple paths. Layouts can be recalled by the server, as in the case for delegations, if there are multiple conflicting client requests.

By allowing the aggregation of bandwidth, pNFS relieves performance issues that are associated with point-to-point connections. With pNFS, clients access data servers directly and in parallel, ensuring that no single storage node is a bottleneck. pNFS also ensures that data can be better load balanced to meet the needs of the client (see figure 3).



OSD: Object based Storage Device

Figure 4; Relationships of pNFS Layouts to NFSv4.1

The pNFS specification also accommodates support for multiple layouts, defining the protocol used between clients and data servers. Currently, three layouts are specified and standardized; files as supported by NFSv4, objects based on the *Object-based Storage Device Commands* (OSD) standard (INCITS T10) approved in 2004, and block layouts (either FC or iSCSI access). The layout choice in any given architecture is expected to make a difference in performance and functionality. For example, pNFS object based implementations may perform RAID parity calculations in software on the client, to allow RAID performance to scale with the number of clients and to ensure end-to-end data integrity across the network to the data servers.

The experience of users with pNFS shows that high bandwidth access to data with pNFS is of considerable benefit. Performance of pNFS is definitely superior to that of NFSv3 for similar configurations of storage, network and server. The management is much easier, as NFSv3 automounter maps and hand-created load balancing schemes are eliminated; and by providing a standardized interface, pNFS ensures fewer issues in supporting multi-vendor NFS server environments.

pNFS allows more than just the three layouts available today. Proposed is a further "Flex-files" layout that provides flexible, per-file striping patterns and simple device information suitable for aggregating standalone NFSv3 and NFSv4 servers into a centrally managed pNFS cluster; and a SCSI pNFS layout that provides block layouts closer integration into the SCSI architecture.

Trunking

Trunking is the use of multiple connections between a client and server in order to increase the speed of data transfer. NFSv4.1supports two types of trunking: session trunking and client ID trunking.

Session trunking is the association of multiple connections, each with potentially different target and/or source network addresses, to the same session. Client ID trunking is the association of multiple sessions to the same client ID. Between them, and with the use of pNFS, NFSv4.1 brings a high degree of parallelization of storage and the potential for optimal resource consumption of network bandwidth.

Security

An area of great confusion, many believe that NFSv4 *requires* the use of strong security. The NFSv4 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 their existing Kerberos security.

Security is increasingly important as NFSv4 makes data more easily available over the WAN. This feature was considered so important by the IETF NFS working group that the security specification using Kerberos v5[14] was "retrofitted" to NFSv2 and NFSv3 and specified in RFC-2623.

Although access to an NFSv2, 3 or 4 file system without strong security such as provided by Kerberos is possible, across a WAN it should really be considered only as a temporary measure. In that spirit, it should be noted that *NFSv4 can be used without implementing Kerberos security*[15]. The fact that it is possible does not make it desirable! A fuller description of the issues and some migration considerations can be found in the SNIA White Paper "Migrating from NFSv3 to NFSv4".

Many of the practical issues faced in implementing robust Kerberos security in a UNIX environment can be eased by using a Windows Active Directory (AD) system. Windows uses the standard Kerberos protocol as specified in RFC-2782; AD user accounts are represented to Kerberos in the same way as accounts in UNIX realms. This can be a very attractive solution in mixed-mode environments (but see the footnote)[16].

Application Data Holes (ADH)

NFSv4.2 ADH allows definition of the format of a file or part of the file; for example, a VM image or a database. This feature will allow initialization of data stores; a single operation from the client can create, for example, a 300GB database or a VM image on the server in a single operation, rather than sending a corresponding 300GB of largely repetitive data.

I/O Advise

Applications and clients want to advise the server as to expected I/O behavior. This NFSv4.2 feature lets clients communicate the anticipated future I/O behavior; for example, whether a file will be

---

[14] "Kerberos Overview− An Authentication Service for Open Network Systems http://www.cisco.com/application/pdf/paws/16087/1.pdf
15 For examples of NFSv4 without Kerberos, see Ubuntu Linux; https://help.ubuntu.com/community/SettingUpNFSHowTo and SUSE Linux Enterprise; http://www.novell.com/support/kb/doc.php?id=7005060
[16] Windows Identity Management for UNIX – https://msdn.microsoft.com/en-us/library/cc772571.aspx. "Identity Management for Unix (IDMU) is deprecated and will not ship in future versions of Windows Server" including Windows Server 2016, see http://blogs.technet.com/b/activedirectoryua/archive/2015/01/25/identity-management-for-unix-idmu-is-deprecated-in-windows-server.aspx. However, Windows Server 2012 R2 has a reasonably long support life left, so this is still a useful proposal.

accessed sequentially or randomly, and whether a file will or will not be accessed in the near future. This allows servers to optimize future I/O requests for a file by, for example, prefetching or evicting data in caches, or moving data to and from slow and fast external devices.

Server Side Copy

NFSv4.2's Server-Side Copy (SSC) removes one leg of a copy operation. Instead of reading entire files or even directories of files from one server through the client, and then writing them out to another, SSC permits the destination server to communicate directly to the source server to copy the data, without client involvement beyond registering the request and authentication, and removes the limitations on server to client bandwidth and the possible congestion it may cause.

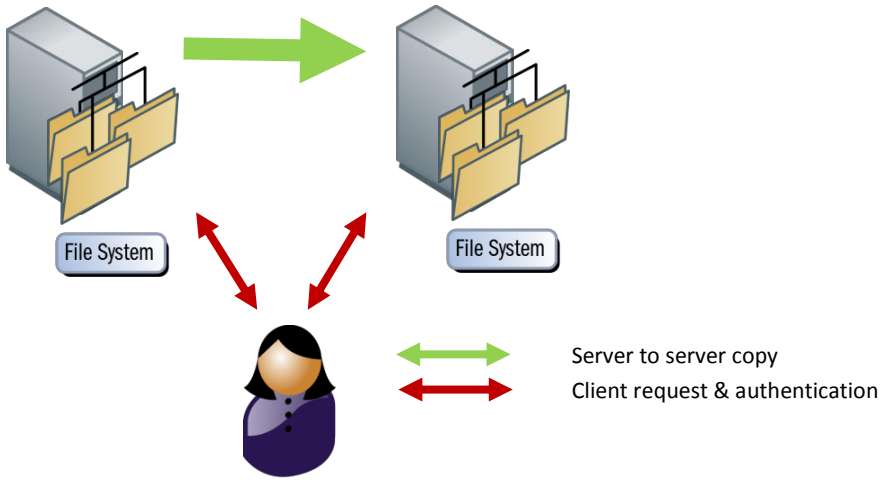


Figure 5; Server Side Copy

There are two types of copy. Even in single server file systems, currently the client has to copy out the data, and then return it to the same server. CLONE operations "optimize away" this overhead, and are copies carried out on the server directly.

COPY operations are server to server; and as different systems in different security realms, security becomes a major concern. NFSv4 formalized a new security model, and an update to that (RPCSEC_GSS Version 3[17]) is required & mandatory for these types of copy; see the discussion about NFSv4 security on page 10.

Guaranteed Space Reservation & Hole Punching

As storage demands continue to increase, various efficiency techniques can be employed to give the appearance of a large virtual pool of storage on a much smaller storage system. Thin provisioning, (where space appears available and reserved, but is not committed) is commonplace, but often problematic to manage in fast growing environments. The guaranteed space reservation feature in

[17] The RPCSEC_GSS Version 3 draft is available at https://tools.ietf.org/html/draft-ietf-nfsv4-rpcsec-gssv3-12

NFSv4.2 will ensure that, regardless of the thin provisioning policies, individual files will always have space available for their maximum extent.

While such guarantees are a reassurance for the end-user, they don't help the storage administrator in his or her desire to fully utilize all his available storage. In support of better storage efficiencies, NFSv4.2 will introduce support for sparse files. Commonly called "hole punching", deleted and unused parts of files are returned to the storage system's free space pool (see Figure 6). These technques are key to the effective management and real space eduction beneficial to technologies like VM datastores and expensive storage devices based on flash (for instance SSDs) and other non-volatile storage class memory.

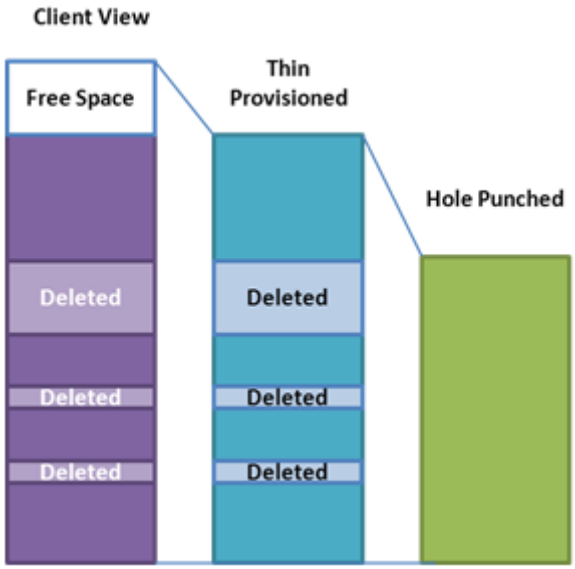


Figure 6; Reservations & Hole Punching

Obtaining Servers and Clients

In June 2012, a comprehensive list of supported OSes and NAS systems would have not been difficult to enumerate. Since then, the number of implementations has increased dramatically. With the features of NFSv4, there is considerable interest in the end-user community for NFSv4 support from both servers and clients. No authoritative list is possible; just be aware that many NFSv4 features are optional and not required to be implemented, so systems vary as to the features they support.

Most Network Attached Storage (NAS) vendors now support NFSv4, and some have server support of NFSv4.1 and pNFS. Features in NFSv4.2, although not standardized yet, are available in a number of solutions. For NFS server vendors, refer to their websites, where you will get the latest up-to-date information.

On the OS server and client side, there are a number of potential solutions from RedHat, Novell SUSE, a variety of BSD based distributions, Oracle Solaris, VMware vSphere and Microsoft Windows. Change is so rapid in this area that it's best to refer to the individual group's website for the exact level of NFSv4 support, and whether clients and/or servers are available.

Conclusion

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
- Optional cryptography quality security, with access control facilities that are compatible across UNIX® and Windows®
- Support for parallelism and data striping
- Allowing clients to take advantage of more sophisticated storage server data management techniques

The goal for NFSv4 and beyond is to define how you get to storage, not what your storage looks like. That has meant inevitable changes. Unlike earlier versions of NFS, the NFSv4 protocol integrates file locking, strong security, operation coalescing, and delegation capabilities to enhance client performance for data sharing applications on high-bandwidth networks.

NFSv4.1 servers and clients provide even more functionality such as wide striping of data to enhance performance. NFSv4.2 and beyond give further enhancements to the standard that increase its applicability to today's application requirements. It is due to be ratified in late 2015, and there are already server and client implementations that provide NFSv4.2 features being shipped now.

With careful planning, migration to NFSv4.1 and NFSv4.2 from prior versions can be accomplished without modification to applications or the supporting operational infrastructure, for a wide range of applications; home directories, HPC storage servers, backup jobs and so on.

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 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 http://www.snia.org.

The June 2012 SNIA whitepaper paper on which this update is based was drawn from an article written by Alex McDonald entitled "NFSv4", originally published in ";login: The Magazine of USENIX", vol. 37, no. 1 (Berkeley, CA: USENIX Association, February 2012), pp. 28-35.