# NFSv4

ALEX MCDONALD

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, and sales and a period as an independent consultant, Alex is now part of NetApp's Office of the CTO, which supports industry activities and promotes technology and standards-based solutions, and is chair of the SNIA CSI Education Subcommittee and co-chair of the SNIA NFS Special Interest Group.
alexmc@netapp.com

NFSv4 has been a standard file-sharing protocol since 2003 but has not been widely adopted. Yet NFSv4 improves on NFSv3 in many important ways. In this article I explain how NFSv4 is better suited to a wide range of datacenter and HPC uses than its predecessor NFSv3, as well as providing resources for migrating from v3 to v4. And, most importantly, I make the argument that users should, at the very least, be evaluating and deploying NFSv4.1 for use in new projects and, ideally, should be using it wholesale in their existing environments.

## The Background to NFSv4.1

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 to be a popular and robust protocol over the 15 years it has been in use, and with wide adoption it soon eclipsed some of the early competitive UNIX-based filesystem 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).

In April 2003 the Network File System (NFS) version 4 protocol was ratified as an Internet standard, described in RFC-3530, which superseded NFSv3.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 introduced the concept of state to ameliorate some of the less desirable features of NFSv3 and offered other enhancements to improve usability, management, and performance.

But shortly following NFSv4's release, an Internet draft written by Garth Gibson and Peter Corbett outlined several problems with it [2]: specifically, that of limited bandwidth and scalability, since NFSv4, like NFSv3, requires that access be to a single server. NFSv4.1 (as described in RFC-5661, ratified in January 2010) was developed to overcome these limitations, and new features such as parallel NFS (pNFS) were standardized to address these issues.

NFSv4.2 is now moving towards ratification [3]. In a change to the original IETF NFSv4 development work, where each revision took a significant amount of time to develop and ratify, the work group 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 ongoing yearly process. With these changes in the processes leading to standardization, features that will be ratified in NFSv4.2 (expected in March 2012) are available from many vendors and suppliers now.

## Adoption of NFSv4

While there have been a number of advances and improvements to NFS, many users have elected to continue with NFSv3. NFSv4 is a mature and stable protocol with significant advantages over its predecessors NFSv3 and NFSv2, yet adoption remains slow. Adequate for some purposes, NFSv3 is a familiar and well understood protocol; but with the demands being placed on storage by exponentially increasing data and compute growth, NFSv3 has become increasingly difficult to deploy and manage.

## 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, 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, among 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 who employ (usually very complicated) NFSv3 automounter maps and modify them to manage load balancing should find that 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 that 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 simplify 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.

## The Advantages of NFSv4.1

The Gibson and Corbett paper [2] identified some issues with NFSv4 that were successfully addressed in NFSv4.1, and NFSv4.1 is where the focus for end-user evaluation and implementation should be. References to features in NFSv4 apply equally to NFSv4.1, since it was a minor version update, unlike the changes from NFSv3 to NFSv4. Many of these base NFSv4 features, such as the pseudo file system, are covered by my SNIA white paper "Migrating from NFSv3 to NFSv4" [4] and will not be covered here.

### *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, metadata operations are largely synchronous and serialized. Operations such as file lookup (LOOKUP), the fetching of attributes (GETATTR), and so on, make up the largest percentage by count of the average workload (Table 1).

| NFSv3 Operation | SPECsfs2008 |
|---|---|
| GETATTR | 26% |
| LOOKUP | 24% |
| READ | 18% |
| ACCESS | 11% |
| WRITE | 10% |
| SETATTR | 4% |
| READDIRPLUS | 2% |
| READLINK | 1% |
| READDIR | 1% |
| CREATE | 1% |
| REMOVE | 1% |
| FSSTAT | 1% |

**Table 1:** SPECsfs2008 %ages for NFSv3 operations [5]

This mix of a typical NFS set of RPC calls in versions prior to NFSv4 requires that each RPC call be 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 considerably reduce latency for multiple operations.

### *Delegations*

Servers are employing ever more quantities of RAM and flash technologies, and very large caches, on the order of terabytes, 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 I/O 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 and the ability to migrate data to provide VM location independence are essential. NFSv4 provides facilities for 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.

### *Sessions*

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. 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.

### *Security*

One area of great confusion is that 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 to migrate 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 [6] was "retrofitted" to NFSv2 and NFSv3 and specified in RFC-2623.

Although access to an NFSv2, 3, or 4 filesystem 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* [7]. 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 [4].

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 1510; 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 [8].

## *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 filesystem 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. pNFS benefits workloads with many small files, or very large files, and is suitable for a range of HPC-type workloads.

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 separate, the MDS is not normally a standalone system but is part of the facilities the cluster provides.) The data may be on many dataservers and is accessed directly by the client over multiple paths. Layouts can be recalled by the server, as is the case for delegations, if there are conflicting multiple client requests.

By allowing the aggregation of bandwidth, pNFS relieves performance issues that are associated with point-to-point connections. The parallel nature of the client connecting directly to multiple dataservers ensures that no single storage node is a bottleneck and that data can be better load-balanced to meet the needs of the client (see Figure 1).

The pNFS specification also accommodates support for many different layouts. Currently, three layouts are specified: 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).

Many vendors have provided vendor-specific modifications that provide similar functionality to pNFS. So although pNFS is relatively new and there may appear to be a limited amount of best practice advice for employing it, the experience of users with proprietary extensions to NFSv3 systems shows that high bandwidth access to data with pNFS will be of considerable benefit.

Potential performance of pNFS is definitely superior to that of NFSv3 for similar configurations of storage, network, and server. The management is definitely 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.

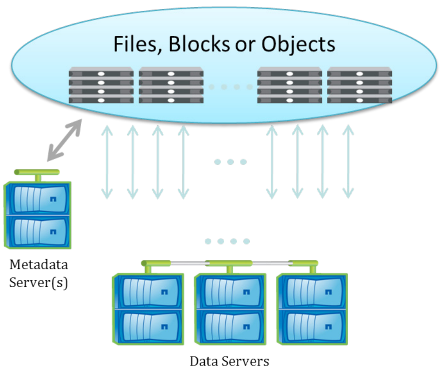

**Figure 1:** Conceptual pNFS data flow

## Some Proposed NFSv4.2 Features

NFSv4.2 promises many features that end users have been requesting, which will make NFS more relevant as not only an "every day" protocol, but one that has application beyond the datacenter.

### *Server-Side Copy (SSC)*

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 without client involvement, and it removes the limitations on server-to-client bandwidth and the possible congestion they may cause.

### *Application Data Blocks (ADB)*

ADB allows definition of the format of a 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 a 300 GB database or a VM image on the server.

### *Guaranteed Space Reservation and 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 is often problematic to manage in fast-growing environments. The guaranteed space reservation feature in 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 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 (Figure 2, next page).

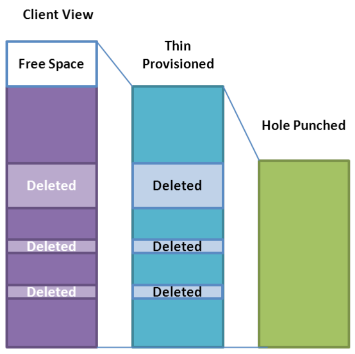

**Figure 2:** Thin-provisioned and hole-punched data

## Availability of Servers and Clients

With this background on the features of NFS, there is considerable interest in the end-user community for NFSv4.1 support from both servers and clients. Many network attached storage (NAS) vendors now support NFSv4, and in recent months there has been a flurry of activity and many developments in server support of NFSv4.1 and pNFS. For NFS server vendors, refer to their Web sites, where you will get up-to-date information.

On the client side, there is Linux Red Hat 6.2, which has an NFSv4.1 Technical Preview [9], and Fedora 15 and 16, available at fedoraproject.org. Both distributions support files-based NFSv4.1 and pNFS. For the adventurous who wish to build their own kernels and want to explore the latest block or objects access, there is full file, block, and object-based pNFS support in the upstream 3.0 and 3.1 Linux kernels.

For Windows, Microsoft has publicly indicated that it will be supporting NFSv4.1 in Windows 8. An open source client implementation preview is available from the Center for Information Technology Integration (CITI) at the University of Michigan [10].

## Conclusion

NFSv4.1 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

The goal for NFSv4.1 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 promise further enhancements to the standard that will increase its applicability to today's application requirements. It is due to be ratified in March 2012, and we can expect to see server and client implementations that provide NFSv4.2 features soon after this; in some cases, the features are already being shipped now as vendor-specific enhancements.

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 uses—home directories, HPC storage servers, backups, and so on.

## *Resources*

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

[2] The "pNFS Problem Statement": http://tools.ietf.org/html/draft-gibson-pnfs-problem-statement-01.

[3] NFSv4.2 proposed specification: http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-06.txt; the draft as of November 2011.

[4] Alex McDonald (SNIA white paper), "Migrating from NFSv3 to NFSv4": http://www.snia.org/sites/default/files/Migrating_NFSv3_to_NFSv4-Final.pdf.

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

[6] "Kerberos Overview—An Authentication Service for Open Network Systems": http://www.cisco.com/application/pdf/paws/16087/1.pdf.

[7] For examples of NFSv4 without Kerberos, see 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.

[8] Windows Security and Directory Services for UNIX Guide v1.0: http://technet.microsoft.com/en-us/library/bb496504.aspx.

[9] Red Hat 6.2 NFSv4.1 Technical Preview: http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/ch12s02.html.

[10] NFSv4 Client for Windows at CITI: http://www.citi.umich.edu/projects/nfsv4/windows/.