Internet Engineering Task Force	M. Eisler, Ed.	
Internet-Draft	NetApp	
Intended status: Informational	September 17, 2010	
Expires: March 21, 2011		

TOC

Requirements for NFSv4.2
draft-ietf-nfsv4-minorversion-2-requirements-00

Abstract

This document proposes requirements for NFSv4.2.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
This Internet-Draft will expire on March 21, 2011.

Table of Contents

---

1. Introduction

TOC

NFSv4.1 [I-D.ietf-nfsv4-minorversion1] (Shepler, S., Eisler, M., and D. Noveck, “NFS Version 4 Minor Version 1,” December 2008.) is an approved specification. The NFSv4 [RFC3530] (Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, “Network File System (NFS) version 4 Protocol,” April 2003.) community has indicated a desire to continue innovating NFS, and specifically via a new minor version of NFSv4, namely NFSv4.2. The desire for future innovation is primarily driven by two trends in the storage industry:

- High efficiency and utilization of resources such as, capacity, network bandwidth, and processors.

- Solid state flash storage which promises faster throughput and lower latency than magnetic disk drives and lower cost than dynamic random access memory.

Secondarily, innovation is being driver by the trend to stronger compliance with information management. In addition, as might be expected with a complex protocol like NFSv4.1, implementation experience has shown that minor changes to the protocol would be useful to improve the end user experience.
This document proposes requirements along these four themes, and attempts to strike a balance between stating the problem and proposing solutions. With respect to the latter, some thinking among the NFS community has taken place, and a future revision of this document will reference embodiments of such thinking.

1.1. Requirements Language

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.) [RFC2119].

2. Efficiency and Utilization Requirements

TOC

2.1. Capacity

TOC

Despite the capacity of magnetic disk continuing to increase at exponential rates, the storage industry is under pressure to make the storage of data increasingly efficient, so that more data can be stored within the same physical space. The driver for this counter-intuitive demand is that disk access times are not improving anywhere near as quickly as capacities. The industry has responded to this development by increasing data density via limiting the number of times a unique pattern of data is stored in a storage device. For example some storage devices support de-duplication. When storing two files, a storage device might compare them for shared patterns of data, and store the pattern just once, and setting reference counts on the blocks of the unique pattern to two. With de-duplication the number of times a storage device has to read a particular pattern would be reduced to just once, thus improving average access time.
For a file access protocol such as NFS, there are several implied requirements for addressing this capacity efficiency trend:

- The "space_used" attribute of NFSv4 does not report meaningful information. Removing a file with a "space_used" value of X bytes does not mean that the file system will see an increase of X available bytes. Providing more meaningful information is a requirement.

- Because it is probable, especially for applications such as hypervisors, the NFSv4 client is accessing multiple files with shared blocks of data, it is in the interest of the client and server for the client to know which blocks are share so that they

are are not read multiple times, and not cached multiple times. Providing a block map of shared blocks is a requirement. For an example of how NFSv4 could deal with this, see [I-D.eisler-nfsv4-pnfs-dedupe] (Eisler, M., "Storage De-Duplication Awareness in NFS," October 2008.).

*If an NFSv4 client is aware of which patterns exist on which files, when it wants to write pattern X to file B to offset J, and it knows that X also exists in offset I of file A, then if it can advise the server of its intent, the server can arrange for pattern X to appear in file A being a zero copy. Even if the server does not support de-duplication, it can at least perform a local copy that saves network bandwidth and processor overhead on the client and server.

*File holes are patterns of zeros that in some file systems do are unallocated blocks. In a sense, holes are the ultimate de-duplicated pattern. While proposals to extend NFS to support hole punching have been around since the 1980s, until recently there have not been NFS clients that could make use of hole punching. The Information Technology (IT) trend toward virtualizing operating environments via hypervisors has resulted in a need for hypervisors to translate a (virtual) disk command to free a block into an NFS request to free that block. On the read side, if a file contains holes, then again, as the ultimate in de-duplication, it would be better for the client to be told the region it wants to read has a hole, instead of of returning long arrays of zero bytes. Even if a server does not support holes on write or read, avoiding the transmission of zeroes will save network bandwidth and reduce processor overhead.

2.2. Network Bandwidth and Processing

TOC

The computational capabilities of processors continues to grow at an exponential rate. However, as noted previously, because disk access times are not showing a commensurate exponential decrease, disk performance is not tracking processor performance. In addition, while network bandwidth is exponentially increasing, unlike disk capacities and processor bandwidth, the improvement is not seen on a 1-2 year cycle, but happens on something closer to a 10 year cycle. The lag between disk and network performance compared to processor performance means that there is often a discontinuity between the processing capabilities of NFS clients and the speed at which they can extract data from an NFS server. For some use cases, much of the data that is read by one client from an NFS server also needs to be read by other clients. Re-reading this data is will result in a waste of the network

bandwidth and processing of the NFS server. This same observation has driven the creation of peer-to-peer content distribution protocols, where data is directly read from peers rather than servers. It is apparent that a similar technique could be used to offload primary storage, such as that proposed in [I-D.myklebust-nfsv4-pnfs-backend] (Myklebust, T., “Network File System (NFS) version 4 pNFS back end protocol extensions,” July 2009.)

The pNFS protocol distributes the I/O to a set of files across a cluster of data servers. Arguably, its primary value is in balancing load across storage devices, especially when it can leverage a back end file system or storage cluster with automatic load balancing capabilities. In NFSv4.1, no consideration was given to metadata. Metadata is critical to several workloads, to the point that, as defined in NFSv4.1, pNFS will not not offer much value in those cases. The load balancing capabilities of pNFS need to be brought to metadata. An example of how to do so is in [I-D.eisler-nfsv4-pnfs-metastripe] (Eisler, M., “Metadata Striping for pNFS,” October 2008.).

From an end user perspective, the operations performed on a file include creating, reading, writing, deleting, and copying. NFSv4 has operations for all but the last. While file copy has been proposed for NFS in the past, it was always rejected because of the lack of Application Programming Interfaces (APIs) within existing operating environments to send a copy operation. The IT trend toward virtualization via hypervisors has changed the situation, where the emerging use case is to copy a virtual disk. The use of a copy operation will save network bandwidth on the client and server, and where the server supports it, intra-server file copy has the potential to avoid all physical data copy. For an example, see [I-D.lentini-nfsv4-server-side-copy] (Lentini, J., Eisler, M., Kenchammana, D., Madan, A., and R. Iyer, “NFS Server-side Copy,” July 2010.).

---

3. Flash Memory Requirements

TOC

Flash memory is rapidly filling the wide gap between expensive but fast Dynamic Random Access Memory (DRAM) and inexpensive but cheap magnetic disk. The cost per bit of flash is between DRAM and disk. The access time pet bit of flash is between DRAM and disk. This has resulted in the File access Operations Per Second (FOPS) per unit of cost of flash exceeding DRAM and disk. Flash can be easily added as another storage medium to NFS servers, and this does not require a change to the NFS protocol. However, the value of flash's superior FOPS is best realized when flash is closest to the application, i.e. on the NFS client. One approach would be to forgo the use of network storage and de-evolve back to Direct Attached Storage (DAS). However, this would require that data protection value that exists in modern storage devices be brought

into DAS, and this is not always convenient or cost effective. A less traumatic way to leverage the full FOPS of flash would be for NFSv4 clients to leverage flash for caching of data.
Today NFSv4 supports whole file delegations for enabling caching. Such a granularity is useful for applications like user home directories where there is little file sharing. However, NFS is used for many more workloads, which include file sharing. In these workloads, files are shared, whereas individual blocks might not be. This drives a requirement for sub-file caching. A derivative of [I-D.eisler-nfsv4-pnfs-dedupe] (Eisler, M., “Storage De-Duplication Awareness in NFS,” October 2008.) could provide sub-file caching, and could be integrated with [I-D.myklebust-nfsv4-pnfs-backend] (Myklebust, T., “Network File System (NFS) version 4 pNFS back end protocol extensions,” July 2009.) to provide off-NFS-server sub-file caching.

---

4. Compliance

TOC

New regulations for the IT industry limit who can view what data. NFSv4 has Access Control Lists (ACLs), but the ACL can be changed by the nominal file owner. In practice, the end user that owns the file (essentially, has the right to delete the file or give permissions to other users), is often not the legal owner of the file. The legal owner of the file wants to control not just who can access (both read and modify) the file, but who they can pass the content of the file to. The legal owner of the file also wants to control which software can manipulate the files of the legal owner (for example the legal owner might want to only allow software that has been certified).
In the past, the IT industry has addressed these requirements with notion of security labeling. Labels are attached to devices, files, users, applications, network connections, etc. When the labels of two objects match, data can be transferred from one to another. For example a label called "Secret" on a file results in only users with a compatible security clearance (e.g. "Secret" or higher) being allowed to view the file, despite what the ACL says.
In environments where labeling is mandated, this often means that a file access protocol like NFSv4 is not permitted, despite the fact that NFSv4 meets many of the other security and non-security requirements of such environments. Thus, it is necessary NFSv4 support labeling and highly desired that label enforcement and application be supported by both the NFSv4 client and server.
To attach a label on a file requires that it be created atomically with the file, which means that a new RECOMMENDED attribute for a security label is needed such as that proposed in [I-D.quigley-nfsv4-sec-label] (Quigley, D. and J. Morris, “MAC Security Label Support for NFSv4,” February 2010.).

5. Incremental Improvements

TOC

Implementation experience with NFSv4.1 and related protocols, such as SMB2, has shown a number of areas where the protocol can be improved.

*Hints for the type of file access, such as sequential read. While traditionally NFS servers have been able to detect read-a-head patterns, with the introduction of pNFS, this will be harder. Since NFS clients can detect patterns of access, they can advise servers. In addition, the UNIX/Linux madvise() API is an example of where applications can provide direct advice to the NFS server.

*Head of line blocking. Consider a client that wants to send a three operations: a file creation, a read for one megabyte, and a write for one megabyte. Each of these might be sent on a separate slot. The client determines that it is not desirable for the read operation to wait for the write operation to be sent, so it sends the create. However, it does not want to serialize the read and write behind the create, so the read gets sent, followed by the write. On the reply side, the server does not know that client wants the create satisfied first, so read and write operations are first processed. By the time the create is performed on the server, the response to the read is still filling the reply side. While NFSv4.1 could solve this problem by associating two connections with the session, and using one connection for create, and the other for read or write, multiple connections come at a cost. The requirement is to solve this head of line blocking problem. Tagging a request as one that should go to the head of the line for request and response processing is one possible way to address it.

*pNFS connectivity/access indication. If a pNFS client is given a layout that directs it to a storage device it cannot access due to connectivity of access control issues, it has no way in NFSv4.1 to indicate the problem to the metadata server. See a proposal to address this in [I-D.faibish-nfsv4-pnfs-access-permissions-check] (Faibish, S., Black, D., Eisler, M., and J. Glasgow, “pNFS Access Permissions Check,” July 2010.).

*RPCSEC_GSS sequence window size on backchannel. The NFSv4.1 specification does not have a way to for the client to tell the server what window size to use on the backchannel. The specification says that the window size will be the same as what the server uses. Potentially, a server could use a very large window size that the client does not want.

*Trunking discovery. The NFSv4.1 specification is long on how a client verifies if trunking is available between two connections, but short on how a client can discover destination addresses that can be trunked. It would be useful if there was a method (such as an operation) to get a list of destinations that can be session or client ID trunked, as well as a notification when the set of destinations changes.

---

6. IANA Considerations

TOC

None.

---

7. Security Considerations

TOC

None.

---

8. Acknowledgements

TOC

Thanks to Dave Noveck and David Quigley for reviewing this document and providing valuable feedback.

---

9. References

TOC

9.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

9.2. Informative References

TOC

[I-D.eisler-nfsv4-pnfs-dedupe]	

	Eisler, M., "Storage De-Duplication Awareness in NFS," draft-eisler-nfsv4-pnfs-dedupe-00 (work in progress), October 2008 (TXT).
[I-D.eisler-nfsv4-pnfs-metastripe]	Eisler, M., "Metadata Striping for pNFS," draft-eisler-nfsv4-pnfs-metastripe-01 (work in progress), October 2008 (TXT).
[I-D.faibish-nfsv4-pnfs-access-permissions-check]	Faibish, S., Black, D., Eisler, M., and J. Glasgow, "pNFS Access Permissions Check," draft-faibish-nfsv4-pnfs-access-permissions-check-03 (work in progress), July 2010 (TXT).
[I-D.ietf-nfsv4-minorversion1]	Shepler, S., Eisler, M., and D. Noveck, "NFS Version 4 Minor Version 1," draft-ietf-nfsv4-minorversion1-29 (work in progress), December 2008 (TXT).
[I-D.lentini-nfsv4-server-side-copy]	Lentini, J., Eisler, M., Kenchammana, D., Madan, A., and R. Iyer, "NFS Server-side Copy," draft-lentini-nfsv4-server-side-copy-05 (work in progress), July 2010 (TXT).
[I-D.myklebust-nfsv4-pnfs-backend]	Myklebust, T., "Network File System (NFS) version 4 pNFS back end protocol extensions," draft-myklebust-nfsv4-pnfs-backend-00 (work in progress), July 2009 (TXT).
[I-D.quigley-nfsv4-sec-label]	Quigley, D. and J. Morris, "MAC Security Label Support for NFSv4," draft-quigley-nfsv4-sec-label-01 (work in progress), February 2010 (TXT).
[RFC3530]	Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol," RFC 3530, April 2003 (TXT).

Author's Address

TOC

	Michael Eisler (editor)
	NetApp
	5765 Chase Point Circle
	Colorado Springs, CO 80919
	US
Phone:	+1 719 599 9026
Email:	mike@eisler.com