Network File Systems (NFS) in Lenovo ONTAP Best Practices and Implementation Guide

Abstract

This document provides basic concepts, support information, configuration tips, and best practices for NFS in Lenovo® ONTAP®. This guide covers the latest available ONTAP versions for currency and length. In most cases, the commands and best practices listed in this guide apply to all ONTAP versions.

Results can vary in older ONTAP versions. Consult the product documentation for your ONTAP version when necessary.

First edition (March 2021)

TABLE OF CONTENTS

1 Basic NFS Concepts in Lenovo ONTAP 8

1.1 Intended Audience and Assumptions........ 8

1.2 Stated Support for NFS........ 8

1.3 NFS Features 9

1.4 NFS Server Options........ 10

1.5 Namespace Concepts........ 12

1.6 File Locking Concepts........ 20

1.7 Export Concepts 22

1.8 The Anon User 29

1.9 The Root User 30

1.10 Limiting Access to the SVM Root Volume........ 30

1.11 Mapping All UIDs to a Single UID (squash_all)........ 32

1.12 Special Character Considerations........ 33

2 NFS Version Considerations 34

2.1 NFSv3 Considerations 34

2.2 NFSv4.x Considerations 38

2.3 NFSv4.0 40

2.4 NFSv4.1 49

2.5 NFSv4.2 53

3 Name Services 53

3.1 Domain Name System (DNS) 53

3.2 Identity Management Name Services 54

3.3 Local Files 54

4 Multiprotocol NAS 56

5 Nondisruptive Operations with NFS 60

5.1 Replay/Reply Cache 60

5.2 File Locking 60

5.3 NFSv4.1 Sessions 61

5.4 What Happens During LIF Migrations in NFSv4.x?........ 61

5.5 LIF Migrations with NFSv3........ 61

5.6 Direct Connect NFS........ 62

6 Volume Styles: FlexGroup or FlexVol? 63

7 NFS Auditing 63

7.1 NFS Audit Setup 63

8 NFS Best Practices 64

8.1 General ONTAP Best Practices 64

8.2 ONTAP Data LIF Best Practices with NFS Environments 64

8.3 NFS Security Best Practices 68

8.4 Name Services Best Practices 74

8.5 NFS Client Best Practices 75

9 Logging, Monitoring and Statistics 80

9.1 Managing NFS Locks 80

9.2 NFS Events – Event Messaging System (EMS) 80

9.3 NFS Statistics 81

9.4 Determining What Type of Virtual Machine Is Hosted on NFS 81

9.5 Determining If Oracle Is in Use 82

9.6 Performance Monitoring Enhancements 82

10 Advanced NFS Concepts 85

10.1 Unmask 85

10.2 NFS User nfsnobody 86

10.3 NFSv4.x – nobody:nobody 87

10.4 Hiding Snapshot Copies 89

10.5 Viewing and Managing NFS Credentials 89

10.6 Using NFSv4.x ACLs with NFSv3 mounts 94

10.7 Dynamic NAS TCP Autotuning test 96

10.8 Network Connection Concurrency: NFSv3 97

10.9 NFSv4.x Client IDs/NFS4ERR_CLID_INUSE 101

10.10 NFS Silly Renames 101

10.11 NAS Flow Control 102

10.12 Using FlexClone to Quickly Change All UIDs/GIDs in a Volume 102

10.13 Auxiliary GIDs – Addressing the 16 GID Limitation for NFS 102

10.14 High File Count Considerations 104

11 NFS on Nontraditional Operating Systems 113

11.1 NFS on Windows 113

11.2 NFS Using Apple OS 114

Appendix A: Examples, Configuration Output, and Other Information 115

Examples 115

Default NFS Ports in ONTAP....158

Where to Find Additional Information 159

RFCs 159

Contacting Support 159

Notices 160

Trademarks.... 161

Version History 162

LIST OF TABLES

Table 1) NFS version support in ONTAP....8
Table 2) NFS security support details. 8
Table 3) NFS feature support. 9
Table 4) Unsupported NFS features....9
Table 5) Export examples....15
Table 6) NFSv3 mode bits versus NFSv4.x ACL granularity. 36
Table 7) NFSv4.x lock terminology....41
Table 8) NFS lease and grace periods. 47
Table 9) Referrals versus migration versus pNFS. 48
Table 10) NFSv4.1 delegation benefits. 52
Table 11) Limits on local users and groups in ONTAP. 56
Table 12) Limitations of existing security styles....57
Table 13) Name mappings and security styles. 58
Table 14) Replay/reply cache NDO behavior. 60
Table 15) Lock state NDO behavior. 61
Table 16) UNIX mode bit levels. 72
Table 17) Virtual machine statistic masks. 81
Table 18) NFS credential cache settings....93
Table 19) Job comparisons - parallel dd with 65,536 and 128 RPC slots. 99
Table 20) Job comparisons - parallel dd with 65,536, 128, and 64 RPC slots....100
Table 21) Job comparisons - parallel dd – NFSv3 and NFSv4.1 with 65536 RPC slots. 100
Table 22) Inode defaults and maximums according to FlexVol size. 105
Table 23) Inode defaults resulting from FlexGroup member sizes and member volume counts. 105
Table 24) NFSv3 wsize/rsize performance summary. 142
Table 25) NFSv4.1 wsize/rsize performance summary. 145
Table 26) NFSv4.1 pNFS wsize/rsize performance summary. 148
Table 27) NFSv4.1 delegations wsize/rsize performance summary. 152
Table 28) NFSv4.1 pNFS/delegations wsize/rsize performance summary. 155
Table 29) Summary of Performance Statistics – High File Count: 800,000 files. 156

Table 30) Summary of completion times – parallel dd, sequential large files. 157
Table 31) Summary of performance statistics – parallel dd. 158
Table 32) Default NFS ports in ONTAP. 158

LIST OF FIGURES

Figure 1) Cluster namespace. 13
Figure 2) LS mirror protection of vsroot volumes. 14
Figure 3) Symlink example using vsroot. 18
Figure 4) Lenovo FlexGroup volumes. 19
Figure 5) Lenovo FlexCache volumes. 20
Figure 6) Qtree export specification – ThinkSystem Storage Manager for DM Series. 23
Figure 7) Reordering the rule index in ThinkSystem Storage Manager for DM Series. 26
Figure 8) pNFS data workflow. 51
Figure 9) Gratuitous ARP during LIF migration 62
Figure 10) Example of setting NFSv4 audit ACE. 64
Figure 11) Single LIF NAS interaction. 65
Figure 12) Multiple LIF NAS interaction. 66
Figure 13) Default actimeo latency - vdbench. 78
Figure 14) Actimeo=600 latency - vdbench. 78
Figure 15) Actimeo=600,nocto latency - vdbench. 79
Figure 16) Filtering events in ThinkSystem Storage Manager for DM Series GUI. 81
Figure 17) Top NAS clients view in ThinkSystem Storage Manager for DM Series. 82
Figure 18) Top files view for an ESXi environment in ThinkSystem Storage Manager for DM Series. 84
Figure 19) Effect of RPC slot tables on NFSv3 performance. 98
Figure 20) Parallel dd performance – NFSv3 and RPC slot tables; 1MB rsize/wsize. 99
Figure 21) Parallel dd performance – NFSv3 and RPC slot tables; 256K rsize/wsize. 99
Figure 22) ThinkSystem Storage Manager for DM Series event screen with `maxdirsize` warning. 112
Figure 23) High file count performance stats, 1MB wsize/rsize. 133
Figure 24) High file count performance stats, 256K wsize/rsize. 133
Figure 25) High file count performance stats, 64K wsize/rsize. 134
Figure 26) High file count performance NFSv3 – 800,000 files, 1MB wsize/rsize. 134
Figure 27) High file count performance NFSv3 – 800,000 files/single folder, 256K wsize/rsize. 135
Figure 28) High file count performance NFSv3 – 800,000 files/single folder, 64K wsize/rsize. 135
Figure 29) High file count performance NFSv4.1 – 800,000 files/single folder, 1MB wsize/rsize. 136
Figure 30) High file count performance NFSv4.1 – 800,000 files/single folder, 256K wsize/rsize. 136
Figure 31) High file count performance NFSv4.1 – 800,000 files/single folder, 64K wsize/rsize. 137
Figure 32) High file count performance NFSv4.1 (pNFS) – 800,000 files/single folder, 1MB wsize/rsize. 137
Figure 33) High file count performance NFSv4.1 (pNFS) – 800,000 files/single folder, 256K wsize/rsize. 137
Figure 34) High file count performance NFSv4.1 (pNFS) – 800,000 files/single folder, 64K wsize/rsize. 138

Figure 35) Parallel dd performance stats NFSv3, 1MB wsize/rsize. 138
Figure 36) Parallel dd performance stats NFSv3, 256K wsize/rsize. 140
Figure 37) Parallel dd performance stats NFSv3, 64K wsize/rsize. 141
Figure 38) Parallel dd performance stats NFSv4.1, 1MB wsize/rsize. 142
Figure 39) Parallel dd performance stats NFSv4.1, 256K wsize/rsize. 143
Figure 40) Parallel dd performance stats NFSv4.1, 64K wsize/rsize. 144
Figure 41) Parallel dd performance stats NFSv4.1 (pNFS), 1MB wsize/rsize. 145
Figure 42) Parallel dd performance stats NFSv4.1 (pNFS), 256K wsize/rsize. 147
Figure 43) Parallel dd performance stats NFSv4.1 (pNFS), 64K wsize/rsize. 148
Figure 44) Parallel dd performance stats NFSv4.1 (delegations); 1MB wsize/rsize. 149
Figure 45) Parallel dd performance stats NFSv4.1 (delegations), 256K wsize/rsize. 150
Figure 46) Parallel dd performance stats NFSv4.1 (delegations), 64K wsize/rsize. 151
Figure 47) Parallel dd performance stats NFSv4.1 (pNFS + delegations), 1MB wsize/rsize. 152
Figure 48) Parallel dd performance stats NFSv4.1 (pNFS + delegations), 256K wsize/rsize. 153
Figure 49) Parallel dd performance stats NFSv4.1 (pNFS + delegations), 64K wsize/rsize. 154

1 Basic NFS Concepts in Lenovo ONTAP

1.1 Intended Audience and Assumptions

This technical report is for storage administrators, system administrators, and data center managers. It assumes basic familiarity with the following:

- The Lenovo ONTAP operating system
- Network file sharing protocols (NFS in particular)

This document contains some advanced and diag-level commands. Exercise caution when using these commands. If there are questions or concerns about using these commands, contact Lenovo Support for assistance.

1.2 Stated Support for NFS

NFS Client Support

ONTAP supports all NFS clients that comply with the standards defined in Request for Comments (RFC) documentation that is ratified by the Internet Engineering Task Force (IETF). Client support for NFSv4.x is also dependent on the stated client support for each vendor.

These are the most recent RFCs for each supported NFS version in ONTAP:

- RFC-1813: NFSv3
- RFC-7530: NFSv4.0
- RFC-5661: NFSv4.1
- RFC-7862: NFS4.2 (currently unsupported in ONTAP)

Windows NFS Support

Windows NFS does not comply with RFC standards, so you must make some additional considerations to use it. For details, see the section "NFS on Windows."

NFS Security Support

Table 2 illustrates the supported security methods for securing NFS, both at-rest and in-transit.

- **At-rest security.** Refers to security for data that is in-place.
- **In-flight security.** Refers to security for data that is being transmitted across the network.

Table 1) NFS security support details.

Supported at-rest security for NFS	Supported in-flight security for NFS
• Lenovo Volume Encryption (LVE) • Lenovo Aggregate Encryption (LAE) • Self-encrypting Drives (SED) • Lenovo Storage Encryption Drives (LSE) • Export Policies and Rules • Access Control Lists (ACLs) • Identity management (users, groups, file ownership)	• Kerberos (krb5, krb5i, krb5p) • Supported encryption types include AES-256, AES-128, 3DES, and DES

Note: NFS over SSH and NFS over Stunnel are not supported with ONTAP.

1.3 NFS Features

Each NFS version adds new features to the protocol to enhance operations, performance, and business use cases. Table 3 shows the supported NFS features in ONTAP, along with the associated NFS version information for that feature.

Table 2) NFS feature support.

NFS Version	Features Available
All versions	• Volume and qtree level export rules • 1024 maximum auxiliary GIDs for NFS operations • 64-bit file IDs • Ability to junction volumes across a namespace to create pseudo-filesystems • UNIX-style mode bit permissions (rwx) • TCP and UDP (NFSv4.x is TCP only) • Changing ports • Limiting port range between 1-1024 (mount-rootonly, nfs-rootonly) • FSID changes across file systems • Showmount (see the “Showmount” section for information and limitations) • NFS client to volume mapping • Multiprotocol NAS access (CIFS/SMB and NFS) • LDAP/NIS for UNIX Identity mapping • Netgroups • Windows NFS
NFSv3	• All RFC standard features for NFSv3 are supported
NFSv4.0/4.1	• ACLs (up to 1024) • Delegations • Migrations • Referrals • Lease timeout configuration • pNFS (v4.1)
NFSv4.2	• Currently unsupported

Table 4 shows features that are currently unsupported for NFS.

Table 3) Unsupported NFS features.

NFS Version	Unsupported Features
All versions	• POSIX ACLs • Subdirectory exports • SSSD Dynamic UIDs
NFSv3	• N/A
NFSv4.0/4.1	• Session trunking/multipath
NFSv4.2	• Currently unsupported

1.4 NFS Server Options

NFS servers in ONTAP are configurable through server options. By default, the options are configured with best practices in mind and should not require specific modification. However, in some cases (such as enabling NFSv4.x support), the NFS options must be changed.

These default values, as well as the available NFS options and descriptions, are covered in the man pages for your version of ONTAP or by running `man nfs modify` from the CLI.

The rootonly Options—nfsrootonly and mountrootonly

The `rootonly` options are added to avoid untrusted client access. Untrusted clients (those not part of the export rules) can potentially access data by using SSH tunneling to trusted clients. However, those requests would come from untrusted ports (ports greater than 1,024). This can provide a back door for clients that are not intended to have access.

Therefore, the enabling or disabling of the `rootonly` options hinges upon need. Does the environment require more ports to allow NFS to function properly? Or is it more important to prevent untrusted clients from accessing mounts?

One potential compromise is to make use of NFSv4.x and/or Kerberos authentication for a higher level of secured access to NFS exports.

In these scenarios, using the `mount-rootonly` and/or `nfs-rootonly` options can alleviate these issues.

To check port usage on the client, run the following command:

```
# netstat -na | grep [IP address]
```

To check port usage on the cluster, run the following command:

```
cluster::> network connections active show -node [nodename] -vserver [vservername] -service nfs*
```

Showmount

ONTAP clusters can potentially have thousands of export rules, so a query for all exports can be process intensive. Additionally, exports are not in flat files and are applied to volumes as rules, so the export path and export rules would live in two different places.

See the following example of `showmount –e` from client with `showmount` disabled in ONTAP:

```
[root@nfsclient /]# showmount -e x.x.x.a
Export list for x.x.x.a:
/ (everyone)
```

The Storage Virtual Machine (SVM, also known as vserver in the CLI) has a vsroot volume mounted to `/`, which is the volume returned in the `showmount` query. All other volumes are mounted below that mount point and are not returned to the client when the `showmount` NFS option is disabled.

What Happens During a Showmount Query?

Showmount leverages the MOUNT protocol in NFSv3 to issue an EXPORT query to the NFS server. If the mount port is not listening or is blocked by a firewall or if NFSv3 is disabled on the NFS server, showmount queries fail:

```
# showmount -e x.x.x.a
mount clntudp_create: RPC: Program not registered
```

The following shows output from a packet trace of the `showmount` command being run against a data LIF in ONTAP with the option disabled:

```
x.x.x.x x.x.x.a MOUNT   170     V3 EXPORT Call (Reply In 17)
Mount Service
Program Version: 3
V3 Procedure: EXPORT (5)

x.x.x.a x.x.x.x MOUNT   202     V3 EXPORT Reply (Call In 16)
Mount Service
Export List Entry: /unix ->
```

Note that the trace shows that the server returns `/unix ->`. However, this export path has a specific client in the rule set.

```
cluster::> vol show -vserver NFS83 -junction-path /unix -fields policy
  (volume show)
vserver volume policy
------- ------ --------
NFS83   unix   restrict

cluster::> export-policy rule show -vserver NFS83 -policyname restrict
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
NFS83        restrict        1       any      x.x.x.y        any
```

If client match is required in showmount functionality, the showmount utility provides that functionality.

Showmount in ONTAP

Showmount functionality is required for some applications to work properly, such as Oracle OVM.

This functionality is disabled by default. It can be enabled with the following command:

```
cluster::> nfs server modify -vserver NFS -showmount
    enabled  disabled
```

Note: To use showmount in ONTAP, the parent volume (including vsroot, or /) must allow read or traverse access to the client/user attempting to run showmount, and vsroot (/) should use UNIX security style.

After this functionality is enabled, clients can query data LIFs for export paths. However, the clientmatch (access from clients, netgroups, and so on) information is not available. Instead, each path reflects everyone as having access, even if clients are specified in export policy rule sets.

See the following sample output of showmount in clustered ONTAP:

```
# showmount -e x.x.x.a
Export list for x.x.x.a:
/unix        (everyone)
/unix/unix1 (everyone)
/unix/unix2 (everyone)
/            (everyone)
```

Note: If you are using Windows NFS, showmount should be enabled to prevent issues with renaming files and folders.

Showmount Caching

When showmount is run from a client, it requests information from the NFS server on the cluster. Because export lists can be large, the cluster maintains a cache of this information to reduce the number of requests made to the NFS server.

When a volume is unmounted from the cluster namespace (see "The Cluster Namespace") using the `volume unmount` command or from ThinkSystem Storage Manager for DM Series, the cache does not update, so the exported path remains in cache until it expires or is flushed.

To flush the showmount cache, run the following command:

```
cluster::> export-policy cache flush -vserver SVM -cache showmount
```

The cache only flushes on the node you are logged in to. For example, if you are logged in to node1's management LIF, then the cache on node1 flushes. This means that only clients connecting to data LIFs local to node1 benefit from the cache flush. To flush the cache on other nodes, log into the node management LIF on those nodes. The node that is flushing is displayed when running the command.

```
cluster::> export-policy cache flush -vserver SVM -cache showmount

Warning: You are about to flush the "showmount" cache for Vserver "SVM" on node "node1", which
will result in increased traffic to the name servers. Do you want to proceed with flushing the
         cache? {y|n}: y
```

1.5 Namespace Concepts

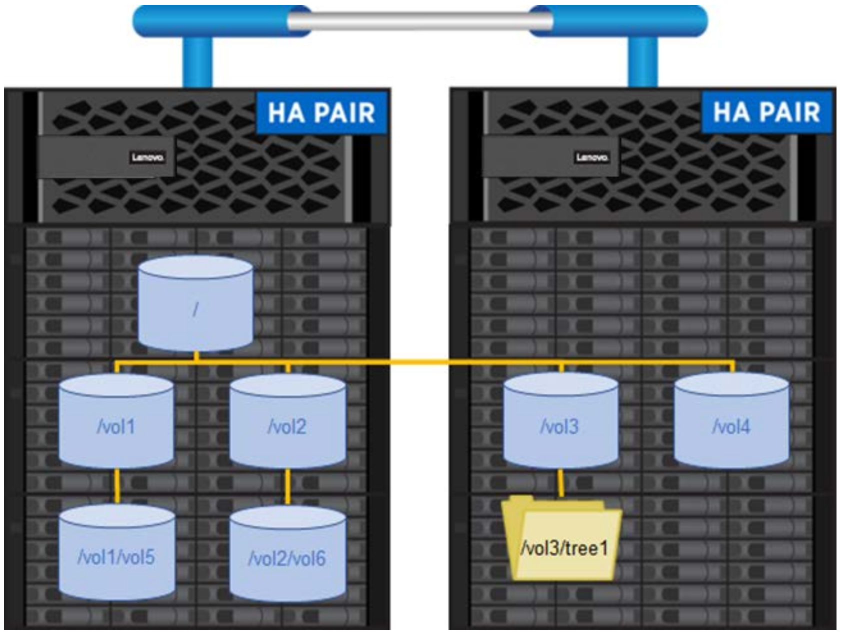
This section covers the concept of a namespace in NFS environments and how ONTAP offers multiple solutions for providing a global namespace for NFS clients.

The Cluster Namespace

A namespace in ONTAP is a collection of file systems hosted across different nodes in the cluster to provide scalable performance and capacity. Each SVM has a file namespace that consists of a single root volume. This namespace starts at the location of "/". Subsequent volumes and qtrees all traverse "/" and have their export paths defined by the volume option `-junction-path`. The SVM namespace can consist of one or more volumes linked by means of junctions that connect from a named junction inode in one volume to the root directory of another volume. A cluster can have more than one SVM, but each SVM only has one vsroot and one "/," which results in each SVM having a unique set of file system IDs. This prevents volumes in different SVMs from sharing file system IDs/file handles and avoids issues mounting NFS exports in multitenant environments.

All the volumes belonging to the SVM are linked into the global namespace in that cluster using the "/" export path. The cluster namespace is mounted at a single point in the cluster. The top directory of the cluster namespace within a cluster ("/") is a synthetic directory containing entries for the root directory of each SVM namespace in the cluster. The volumes in a namespace can be Lenovo FlexVol® volumes or FlexGroup volumes.

Figure 1) Cluster namespace.



Protecting Your Namespace

A vsroot volume only lives on a single node in a cluster, even though the SVM is accessible through multiple nodes. Because the vsroot is how NFS clients traverse the namespace, it is vital to NFS operations.

```
cluster::> vol offline -vserver NFS -volume vsroot

Warning: Offlining root volume vsroot of Vserver NFS will make all volumes on that Vserver
inaccessible.
Do you want to continue? {y|n}: y
Volume "NFS:vsroot" is now offline.
```

If the vsroot volume is somehow unavailable, then NFS clients issue whenever the vsroot volume is needed to traverse the filesystem.

This includes (but might not be limited to) the following behaviors:

- Mount requests hang.
- If / is mounted, traversal from / to another volume, running `ls` (and so on) hangs.
- Umount operations might fail because the mount is busy, even when the volume is back online.
- If a volume is already mounted below “/” (such as /vol1), then reads, writes, and listings still succeed.

Load-sharing mirrors (LS mirrors) in ONTAP is a way to leverage the ONTAP SnapMirror capability to increase vsroot resiliency.

Note: LS mirrors are supported only with vsroot volumes. To share load across data volumes, consider using Lenovo FlexCache Volumes instead.

When LS mirrors are available for the vsroot volume, NFSv3 operations are able to leverage the LS mirror destination volumes to traverse the filesystem. When LS mirrors are in use, it is possible to access the source volume through the .admin folder within the NFS mount.

Lenovo highly recommends that you create LS mirror relationships for vsroot volumes in NFSv3 environments.

Note: NFSv4.x clients are unable to use LS mirror volumes to traverse filesystems due to the nature of the NFSv4.x protocol.

In Figure 2, we can see how LS mirrors can provide access to "/" in the event the vsroot is unavailable.

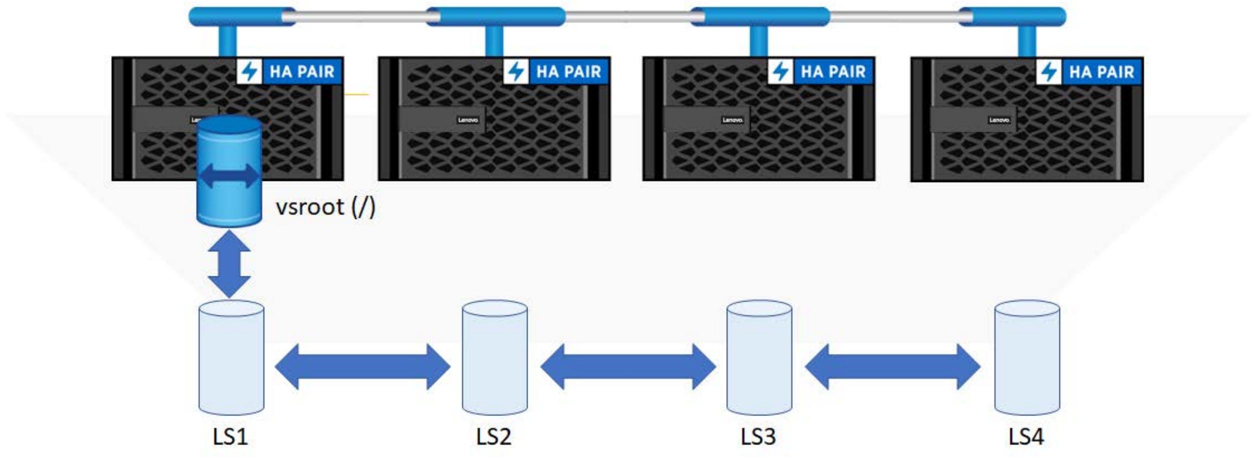


Figure 2) LS mirror protection of vsroot volumes.

To create an LS mirror for the vsroot volume, complete the following steps:

1. Typically, the vsroot volume is 1GB in size. Verify the vsroot volume size prior to creating new volumes and make sure that the new volumes are all the same size.
2. Create a destination volume to mirror the vsroot on each node in the cluster. For example, in a four-node cluster, create four new volumes with the -type DP.
3. Create a new SnapMirror relationship from the vsroot source to each new DP volume you created. Specify a schedule for updates depending on the change rate of your namespace root. For example, use hourly if you create new volumes regularly or daily if you do not.
4. Initialize the SnapMirror mirrors by running the `initialize-ls-set` command.

Pseudo Filesystems

The ONTAP architecture has made it possible to have a true pseudo-file system that complies with RFC 7530 NFSv4 standards:

```
Servers that limit NFS access to "shares" or "exported" file systems should provide a pseudo-file
system into which the exported file systems can be integrated, so that clients can browse the
server's namespace. The clients' view of a pseudo-file system will be limited to paths that lead
to exported file systems.
```

Also, in section 7.3:

```
NFSv4 servers avoid this namespace inconsistency by presenting all the exports within the
framework of a single-server namespace.  An NFSv4 client uses LOOKUP and READDIR operations to
browse seamlessly from one export to another.  Portions of the server namespace that are not
exported are bridged via a "pseudo-file system" that provides a view of exported directories
only.  A pseudo-file system has a unique fsid and behaves like a normal, read-only file system.
```

A pseudo-file system applies only in ONTAP if the permissions flow from more restrictive to less restrictive. For example, if the vsroot (mounted to `/`) has more restrictive permissions than a data volume (such as `/volname`) does, then pseudo-file system concepts apply.

Having a pseudo-file system allows storage administrators to create their own file system namespaces, if they desire, by way of mounting volumes to other volumes using junction paths. This concept is illustrated in Figure 1.

Pseudo File System and -actual Support

The use of -actual as an export option is not supported in ONTAP. The `-actual` option is used in cases where storage administrators wish to mask export paths and redirect to shorter names.

For example, if a storage administrator has a path of `/dept1/folder1`, then they might want to export that path as `/folder1`.

Working Around Lack of -actual Support

In most cases, the `-actual` export option is not necessary in ONTAP. The design of the operating system allows natural pseudo-file systems rather than those defined in export files. Everything is mounted beneath the SVM root volume, which is mounted to `/`. Exports can be set at the volume or qtree level, can be junctioned several levels deep, and can have names that do not reflect the actual volume names.

Table 4) Export examples.

Export Path	Exported Object
`/vol1`	volume named vol1
`/NFSvol`	volume named vol2
`/vol1/NFSvol`	volume named vol2 junctioned to volume named vol1
`/vol1/qtree`	Qtree named qtree with parent volume named vol1
`/vol1/NFSvol/qtree1`	Qtree named qtree1 with parent volume named NFSvol junctioned to volume named vol1

In ONTAP, the path for a qtree that NFS clients mount is the same path as the qtree is mounted to in the namespace. If this is not desirable, then the workaround is to leverage symlinks to mask the path to the qtree.

What Is a Symlink?

Symlink is an abbreviation for symbolic link. A symbolic link is a special type of file that contains a reference to another file or directory in the form of an absolute or relative path. Symbolic links operate transparently to clients and act as actual paths to data.

Relative Paths Versus Absolute Paths

Symlinks can leverage either relative or absolute paths. Absolute paths are paths that point to the same location on one file system regardless of the present working directory or combined paths. Relative paths are paths relative to the working directory.

For example, if a user is inside a directory at /directory and wants to go to /directory/user, that user can use a relative path:

```
# cd user/
# pwd
/directory/user
```

Or the user can use the absolute path:

```
# cd /directory/user
# pwd
/directory/user
```

When mounting a folder using NFS, it is better to use a relative path with symlinks, because there is no guarantee that every user mounts to the same mount point on every client. With relative paths, symlinks can be created that work regardless of what the absolute path is.

Using Symlinks to Simulate –actual Support

For example, if a qtree exists in the cluster, the path can look like this:

```
cluster::> qtree show -vserver flexvol -volume unix2 -qtree nfstree

                      Vserver Name: flexvol
                       Volume Name: unix2
                        Qtree Name: nfstree
                        Qtree Path: /vol/unix2/nfstree
                    Security Style: unix
                       Oplock Mode: enable
                  Unix Permissions: ---rwxr-xr-x
                          Qtree Id: 1
                      Qtree Status: normal
                     Export Policy: volume
        Is Export Policy Inherited: true
```

The parent volume is `unix2 (/unix/unix2)`, which is mounted to volume `unix (/unix)`, which is mounted to `vsroot` (/).

```
cluster::> vol show -vserver flexvol -volume unix2 -fields junction-path
  (volume show)
vserver volume junction-path
------- ------ -------------
flexvol unix2  /unix/unix2
```

The exported path would be `/parent_volume_path/qtree`, rather than the `/vol/parent_volume_path/qtree` seen earlier. The following is the output from a `showmount -e` command from an NFS client:

```
/unix/unix2/nfstree                                              (everyone)
```

Some storage administrators might not want to expose the entire path of `/unix/unix2/nfstree`, because it can allow clients to attempt to navigate other portions of the path. To allow the masking of that path to an NFS client, a symlink volume or folder can be created and mounted to a junction path. For example:

```
cluster::> vol create -vserver flexvol -volume symlinks -aggregate aggr1 -size 20m -state online
-security-style unix -junction-path /NFS_links
```

The volume size can be small (minimum of 20MB), but that depends on the number of symlinks in the volume. Each symlink is 4k in size, so you might need to create larger volume sizes to accommodate the number of symlinks. Alternatively, create a folder under vsroot for the symlinks.

After the volume or folder is created, mount the vsroot to an NFS client to create the symlink.

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
```

Note: If using a directory under vsroot, mount vsroot and create the directory.

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
# mkdir /symlink/symlinks
# ls -la /symlink | grep symlinks
drwxr-xr-x.  2 root root 4096 Apr 30 10:45 symlinks
```

To create a symlink to the qtree, use the `-s` option (s = symbolic). The link path needs to include a relative path that directs the symlink to the correct location without needing to specify the exact path. If the link is inside a folder that does not navigate to the desired path, then ../ needs to be added to the path.

For example, if a folder named NFS_links is created under `/` and the volume UNIX is also mounted under `/`, then navigating to `/NFS_links` and creating a symlink cause the relative path to require a redirect to the parent folder.

See the following example of a symlink created in a symlink volume mounted to `/NFS_links`:

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink/
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /symlink/NFS_links
# pwd
/symlink/NFS_links
# ln -s ../unix/unix2/nfstree LINK
# ls -la /symlink/unix/unix2/nfstree/
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# cd LINK
# ls -la
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# pwd
/symlink/NFS_links/LINK
```

Note that, despite the fact that the symlink points to the actual path of `/unix/unix2/nfstree`, pwd returns the path of the symlink, which is `/symlink/NFS_links/LINK`. The file `you_are_here` has the same date and timestamp across both paths.

Note: Because the path includes ../, this symlink cannot be directly mounted.

See the following example of symlink created in vsroot:

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink/
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /symlink/
# pwd
/symlink
# ln -s unix/unix2/nfstree LINK1
# ls -la /symlink/unix/unix2/nfstree/
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# cd LINK1
# ls -la
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# pwd
/symlink/LINK1
```

Again, despite the fact that the actual path is `/unix/unix2/nfstree`, we see an ambiguated path of `/symlink/LINK1`. The file `you_are_here` has the same date and timestamp across both paths. Additionally, the symlink created can be mounted instead of the vsroot path, adding an extra level of ambiguity to the export path:

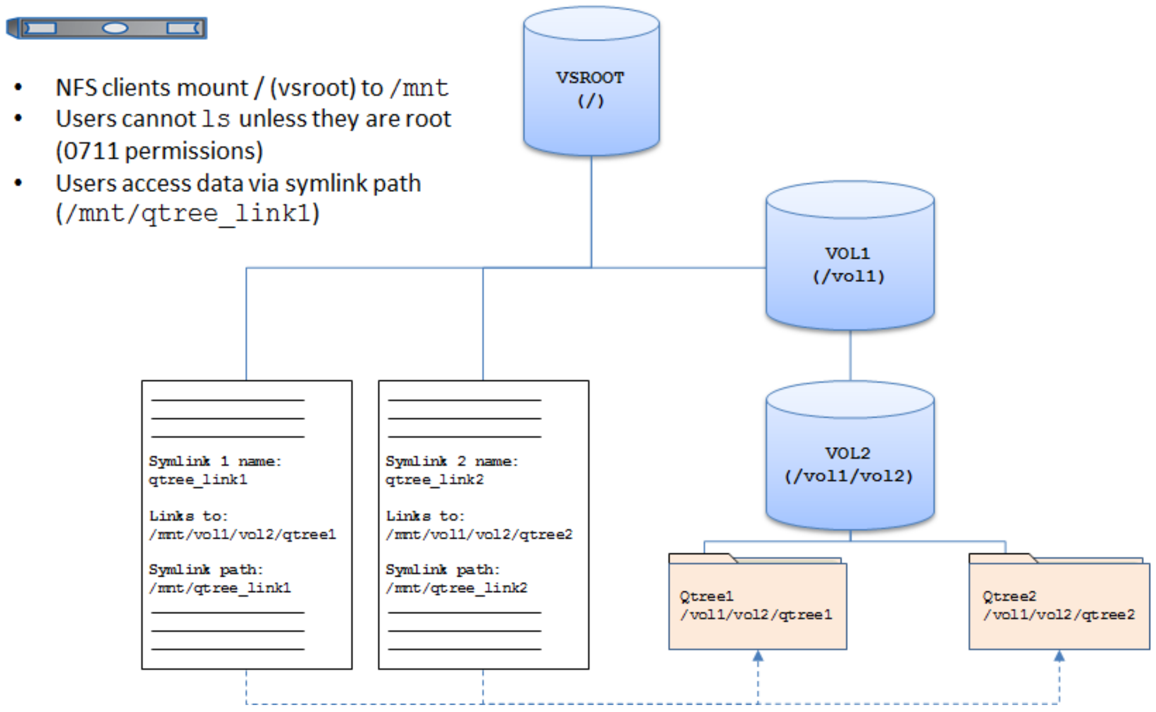
```
# mount -o nfsvers=3 x.x.x.e:/LINK1 /mnt
# mount | grep mnt
x.x.x.e:/LINK1 on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e
# cd /mnt
# pwd
/mnt
```

One use case for this setup is with automounters. Every client can mount the same path and never actually know where in the directory structure they are. If clients mount the SVM root volume (/), be sure to lock down the volume to non-administrative clients.

For more information about locking down volumes to prevent listing of files and folders, see the section “Limiting Access to the SVM Root Volume.”

Figure 3 shows how a namespace can be created to leverage symlinks to create ambiguation of paths for NAS operations.

Figure 3) Symlink example using vsroot.



• NFS clients mount / (vsroot) to /mnt
• Users cannot ls unless they are root (0711 permissions)
• Users access data via symlink path (/mnt/qtree_link1)
VSROOT (/)
VOL1 (/vol1)
VOL2 (/vol1/vol2)
Symlink 1 name: qtree_link1
Links to: /mnt/vol1/vol2/qtree1
Symlink path: /mnt/qtree_link1
Symlink 2 name: qtree_link2
Links to: /mnt/vol1/vol2/qtree2
Symlink path: /mnt/qtree_link2
Qtree1 /vol1/vol2/qtree1
Qtree2 /vol1/vol2/qtree2

Note: Export policies and rules can be applied to volumes and qtrees, but not folders or symlinks. This fact should be taken into consideration when creating symlinks for use as mount points. Symlinks instead inherit the export policy rules of the parent volume in which the symlink resides.

Lenovo FlexGroup Volumes

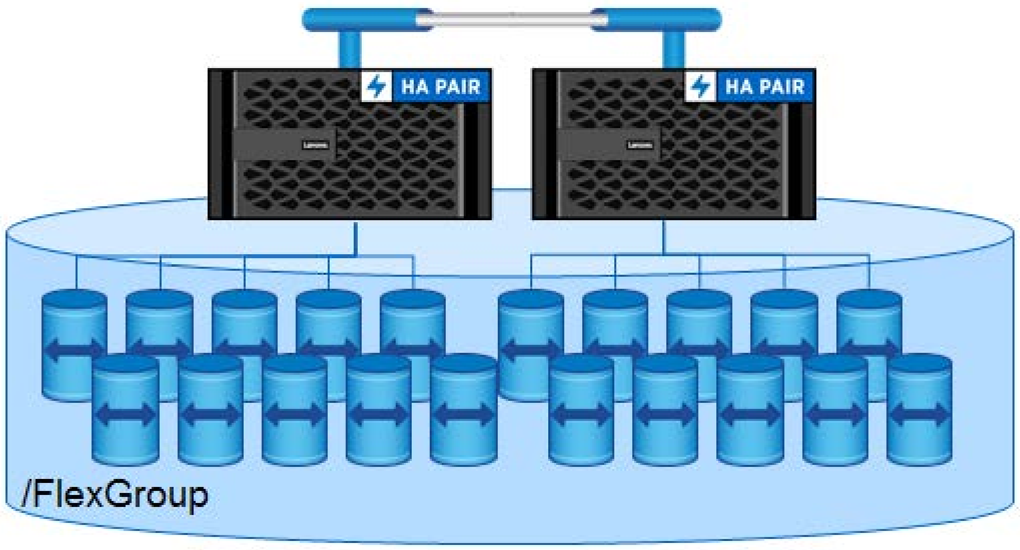
Beginning in ONTAP 9.1, a new way to present storage to NAS (CIFS/SMB and NFS) clients was introduced: the Lenovo FlexGroup.

A Lenovo FlexGroup volume is intended to provide the following benefits:

- Immense capacity for a single mount point (up to 20PB, 400 billion files)
- Performance gains over Lenovo FlexVol volumes by way of the concurrency of ingest operations
- Ease of deployment and management

A Lenovo FlexGroup volume provides NFS clients with a true global namespace—a single large bucket of storage that spans multiple nodes in a cluster and provides parallel metadata operations for NAS workloads.

Figure 4) Lenovo FlexGroup volumes.



Ideal Use Cases

A FlexGroup volume works best with workloads that are heavy on ingest (a high level of new data creation), heavily concurrent, and evenly distributed among subdirectories:

- Electronic design automation (EDA)
- Log file repositories
- Software build/test environments (such as GIT)
- Seismic, oil, and gas
- Media assets or HIPAA archives
- File streaming workflows
- Unstructured NAS data (such as home directories)
- Big data and data science
- Artificial Intelligence and Machine Learning
- Home directories
-

Lenovo FlexCache Volumes

ONTAP 9.5 introduced a new feature for balancing read-heavy NFS workloads across multiple nodes called Lenovo FlexCache volumes.

FlexCache in ONTAP provides a writable, persistent cache of a volume in a remote place. A cache is a temporary storage location that resides between a host and a source of data. The objective of a cache is to store frequently accessed portions of source data in a way that allows the data to be served faster than it would be by fetching the data from the source. Caches are beneficial in read-intensive environments where data is accessed more than once and is shared by multiple hosts. A cache can serve data faster in one of two ways:

- The cache system is faster than the system with the data source. This can be achieved through faster storage in the cache (for example, solid-state drives [SSD] versus HDD), increased processing power in the cache, and increased (or faster) memory in the cache.
- The storage space for the cache is physically closer to the host, so it does not take as long to reach the data.

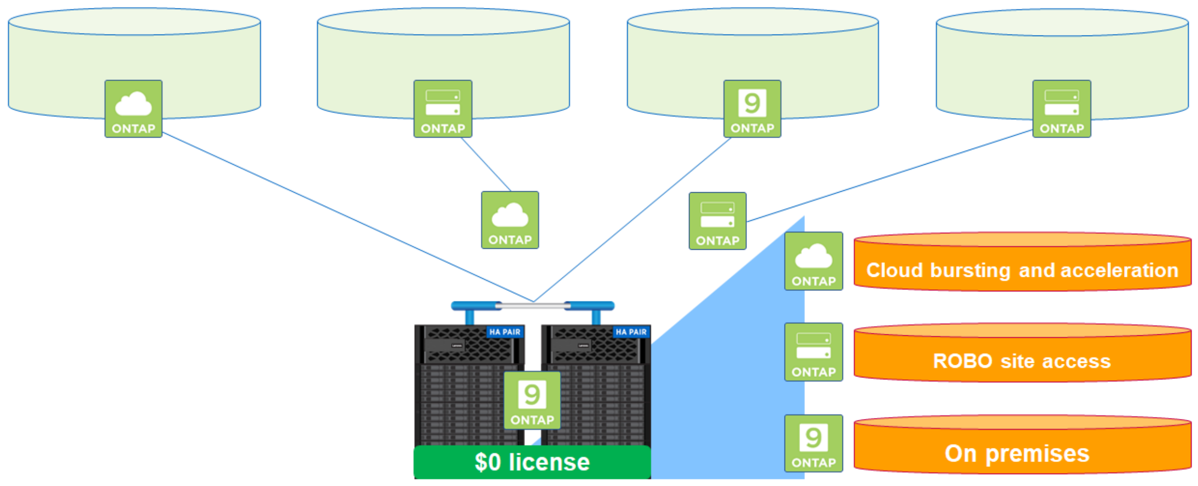
Caches are implemented with different architectures, policies, and semantics so that the integrity of the data is protected as it is stored in the cache and served to the host.

FlexCache offers the following benefits:

- Improved performance by providing load distribution
- Reduced latency by locating data closer to the point of client access
- Enhanced availability by serving cached data in a network disconnection situation

FlexCache provides all the above advantages while maintaining cache coherency, data consistency, and efficient use of storage in a scalable and high-performing manner.

Figure 5) Lenovo FlexCache volumes.



A FlexCache is a sparse copy; not all files from the origin dataset can be cached, and, even then, not all data blocks of a cached inode can be present in the cache. Storage is used efficiently by prioritizing retention of the working dataset (recently used data).

With FlexCache, the management of disaster recovery and other corporate data strategies only needs to be implemented at the origin. Because data management is only on the source, FlexCache enables better and more efficient use of resources and simpler data management and disaster recovery strategies.

-

1.6 File Locking Concepts

File locking is a way that applications can preserve the integrity of a file when it is open and in use by notifying other clients that attempt to open the file that it is currently locked. With NFS, file locking mechanisms depend on the NFS version being used.

NFSv3 locking

NFSv3 uses ancillary protocols like Network Lock Manager (NLM) and Network Status Monitor (NSM) to coordinate file locks between the NFS client and server. NLM helps establish and release locks, while NSM notifies peers of server reboots. With NFSv3 locking, when a client reboots, the server has to release the locks. When a server reboots, the client reminds the server of the locks it held. In some cases, the lock mechanisms don't communicate properly and stale locks get leftover on the server and must be manually cleared.

NFSv4.x locking

NFSv4.x uses a lease-based locking model that is integrated within the NFS protocol. This means there are no ancillary services to maintain or worry about; all the locking is encapsulated in the NFSv4.x communication.

When a server or client reboots, if the lock cannot be re-established during a specified grace period, then the lock expires. ONTAP NFS servers control this lock timeout period with the options `-v4-grace-seconds` and `-v4-lease-seconds`.

- `-v4-lease-seconds` refers to how long a lease is granted before the client has to renew the lease. The default is 30 seconds, with a minimum of 10 seconds and maximum of -1 second of the value of `-v4-grace-seconds`.
- `-v4-grace-seconds` refers to how long a client attempts to reclaim a lock from ONTAP during a reboot of a node (such as during failovers/givebacks). The default is 45 seconds and can be modified with a range of +1 second of the `-v4-lease-seconds` value and a maximum of 90 seconds.

In rare cases, locks might not be freed as quickly as stated by the lease seconds value, which results in the locks being freed over the course of two lease periods. For example, if grace seconds is set to 45 seconds, it can take 90 seconds to free the lock. See bug 957529 for more information.

Lock types

NFS locks come in several flavors:

- **Shared locks.** Shared locks can be used by multiple processes at the same time and can only be issued if there are no exclusive locks on a file. These are intended for read-only work but can be used for writes (such as with a database).
- **Exclusive locks.** These locks operate the same as exclusive locks in CIFS/SMB—only one process can use the file when there is an exclusive lock. If any other processes have locked the file, an exclusive lock cannot be issued unless that process was forked.
- **Delegations.** Delegations are only used with NFSv4.x and are assigned when the NFS server options are enabled and the client supports NFSv4.x delegations. Delegations provide a way to cache operations on the client side by creating a soft lock to the file being used by a client. This helps improve some aspects of performance for operations by reducing the number of calls being made between the client and server and are similar to SMB opportunistic locks. For more information on delegations, see "NFSv4.1 Delegations."
- **Byte-range locks.** Rather than locking an entire file, byte-range locks only lock a portion of a file.

Note: Locking behavior is dependent on the type of lock, the client OS version, and the NFS version being used. Be sure to test locking in your environment to gauge the expected behavior.

For more information on file locking in ONTAP, see the product documentation section called "Managing file locks."

Manually Establishing Locks on a Client

To test NFS locks, the client must tell the NFS server to establish a lock. However, not all applications use locks. For example, an application like “vi” does not lock a file; instead, it creates a hidden swap file in the same folder and then commits writes to that file when the application is closed. Then the old file is deleted, and the swap file is renamed to the filename.

There are utilities to manually establish locks, however. For example, flock can lock files.

1. To establish a lock on a file, first run “exec” to assign a numeric ID.

```
# exec 4<>v4user_file
```

2. Use flock to create a shared or exclusive lock on the file.

```
# flock

Usage:
 flock [options] <file|directory> <command> [command args]
 flock [options] <file|directory> -c <command>
 flock [options] <file descriptor number>

Options:
 -s  --shared             get a shared lock
 -x  --exclusive          get an exclusive lock (default)
 -u  --unlock             remove a lock
 -n  --nonblock           fail rather than wait
 -w  --timeout <secs>     wait for a limited amount of time
 -E  --conflict-exit-code <number>  exit code after conflict or timeout
 -o  --close              close file descriptor before running command
 -c  --command <command>  run a single command string through the shell

 -h, --help     display this help and exit
 -V, --version  output version information and exit

# flock -n 4
```

3. Check the ONTAP SVM for the lock.

```
cluster::*> vserver locks show -vserver DEMO

Notice: Using this command can impact system performance. It is recommended
that you specify both the vserver and the volume when issuing this command to
minimize the scope of the command's operation. To abort the command, press Ctrl-C.

Vserver: DEMO
Volume   Object Path               LIF         Protocol  Lock Type   Client
-------- ------------------------- ----------- --------- ----------- ----------
home     /home/v4user_file         data2       nlm       byte-range  10.x.x.x
                Bytelock Offset(Length): 0 (18446744073709551615)
```

4. Unlock the file.

```
# flock -u -n 4
```

Manually locking files allows you to test file open and edit interactions, as well as seeing how file locks handle storage failover events.

1.7 Export Concepts

Volumes in ONTAP are shared out to NFS clients by exporting a path that is accessible to a client or set of clients. When a volume is mounted to the SVM’s namespace, a file handle is created and presented to NFS clients when requested in a mount command. Permissions to these exports are defined by export policies and rules, which are configurable by storage administrators.

Export Policy and Rule Concepts

ONTAP offers export policies as containers for export policy rules to control security. These policies are stored in a replicated database, thus making exports available across every node in the cluster, rather than isolated to a single node.

To provide or restrict NFS access to these volumes, export policy rules are created. These rules can define read, write, and root access, as well as specifying client lists. Multiple rules can exist in a policy and multiple clients can exist in a single rule.

The Default Export Policy

A newly created SVM contains an export policy called “default.” This export policy cannot be deleted, although it can be renamed or modified. When an NFS server is created, the “default” policy is automatically created and applied to the vsroot volume. However, the “default” policy has no export rules in it, so access to volumes using the default export policy is not able to mount until rules are added. When new volumes are created, if no export policy is defined, then the export policy of the vsroot volume is inherited.

Vsroot and Volume Traversal

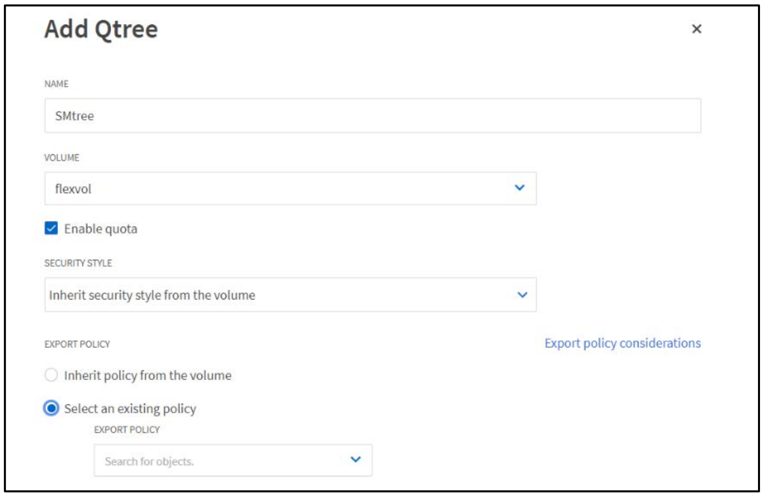
Because export policies are inherited by default, Lenovo recommends opening read access to the root volume of the SVM (vsroot) to NFS clients when a rule is assigned. Setting any rules for the “default” export policy that restrict read access to the vsroot denies traversal to the volumes created under that SVM and cause mounts to fail. That is because vsroot is “/” in the path to “/junction” and factors into the ability to mount and traverse.

Qtree Exports

In ONTAP, it is possible to set export policies and rules for volumes, as well as underlying qtrees. This offers a way to restrict/allow client access to storage-managed directories in ONTAP, which can help storage administrators more easily manage workloads such as home directories.

By default, qtrees inherit the export policy of the parent volume. You can explicitly choose or create an export policy and rule when creating qtrees in ThinkSystem Storage Manager for DM Series, or by using the `-export-policy` CLI option.

Figure 6) Qtree export specification – ThinkSystem Storage Manager for DM Series.



Add Qtree

NAME: SMtree

VOLUME: flexvol

[✓] Enable quota

SECURITY STYLE: Inherit security style from the volume

EXPORT POLICY

○ Inherit policy from the volume

● Select an existing policy

EXPORT POLICY: Search for objects.

Qtree IDs and Rename Behavior

After a non-inherited export policy is applied to a qtree, NFS file handles change slightly when dealing with operations between qtrees. ONTAP validates qtree IDs in NFS operations, which affects things like file renames or moves when moving to or from a qtree in the same volume as the source folder or qtree. This is considered a security feature, which helps prevent unwanted access across qtrees, such as in home directory scenarios. However, simply applying export policy rules and permissions can achieve similar goals.

For example, a move or rename to or from a qtree in the same volume results in “Access denied.” The same move or rename to or from a qtree in a different volume results in the file being copied. With larger files, the “copy” behavior can make it seem like a move operation is taking an unusually long time, where most move operations are near-instantaneous, because they are simple file renames when in the same file system/volume.

This behavior is controlled by the advanced privilege option.

To change the behavior of renames/moves across qtrees, modify `-validate-qtree-export` to `disabled`.

Note: There are no known negative effects caused by disabling the `-validate-qtree-export` option outside of allowing renames across qtrees.

File Handle effect for qtree Exports

Normally, the NFS export file handles that are handed out to clients are 32 bits or less in size. However, with qtree exports, extra bits are added to create 40-bit file handles. In most clients, this is not an issue. However, older clients (such as HPUX 10.20, introduced in 1996) might have problems mounting these exports. Be sure to test older client connectivity in a separate test SVM before enabling qtree exports, because there is currently no way to change the file handle behavior after qtree exports have been enabled.

Access Control to vsroot

To control read/write access to vsroot, use the volume unix permissions and/or ACLs. Lenovo recommends restricting the ability for nonowners of the volume to write to vsroot (at most, 0755 permissions).

When volumes are created, the following values are the defaults, unless specified otherwise:

- 0755 is the default UNIX security set on volumes.
- The default owner is UID 0, and the default group is GID 1.

To enable traversing vsroot, which also prevents read/list access to NFS clients that might mount “/”, there are two approaches:

Option 1: Lock Down UNIX Mode Bits on vsroot

The simplest way to lock down vsroot to users is to manage the ownership and permissions from the cluster.

1. Create a local UNIX user that is unique to the SVM. For example, this UNIX user could be the same name as the SVM itself.
2. Set the vsroot volume to the new UNIX user. Most NFS clients have a “root” user, which means, by default, the vsroot volume might have too much access by the root user.
3. Use UNIX permissions that limit groups and “others” to only traverse permissions but leave desired permissions for the volume owner (for example, 0611).

Option 2: Use NFSv4.x or NTFS ACLs to Lock Down vsroot

Another way to lock down vsroot is to use Access Control Lists to limit permissions to traverse for everyone except a select few users or groups. This can be done with NFSv4.x ACLs (even if you mount using NFSv3) or with NTFS permissions in environments that are also serving CIFS/SMB protocols. For information on using NFSv4.x ACLs with NFSv3 mounts, see "Using NFSv4.x ACLs with NFSv3 mounts."

Export Policy Rules: Options

The appendix of this document (Export Policy Rule Inheritance Example) includes a table that lists the various options used for export policy rules and what they are used for. Most export policy rule options can be viewed by using the `export-policy rule show` command or by using ThinkSystem Storage Manager for DM Series.

Export Policy Rules: Inheritance

In ONTAP, export policy rules affect only the volumes and qtrees that they are applied to. For example, if the SVM root volume has a restrictive export policy rule that limits root access to a specific client or subset of clients, the data volumes that exist under the SVM root volume (which is mounted at "`/`") honor only the export policies applied to them. The one exception is if a volume has an export policy rule that denies read access to a client and that client must traverse that volume in that path. There is currently not concept of bypass traverse checking for NFS in ONTAP. For an example of export policy rule inheritance, see "Export Policy Rule Inheritance Example."

Export Policy Rules: Index

ONTAP offers storage administrators the ability to set the priority for export policy rules so that they are honored in a specific order. The policy is evaluated when access is attempted and the rules are read in order from 0 to 999999999.

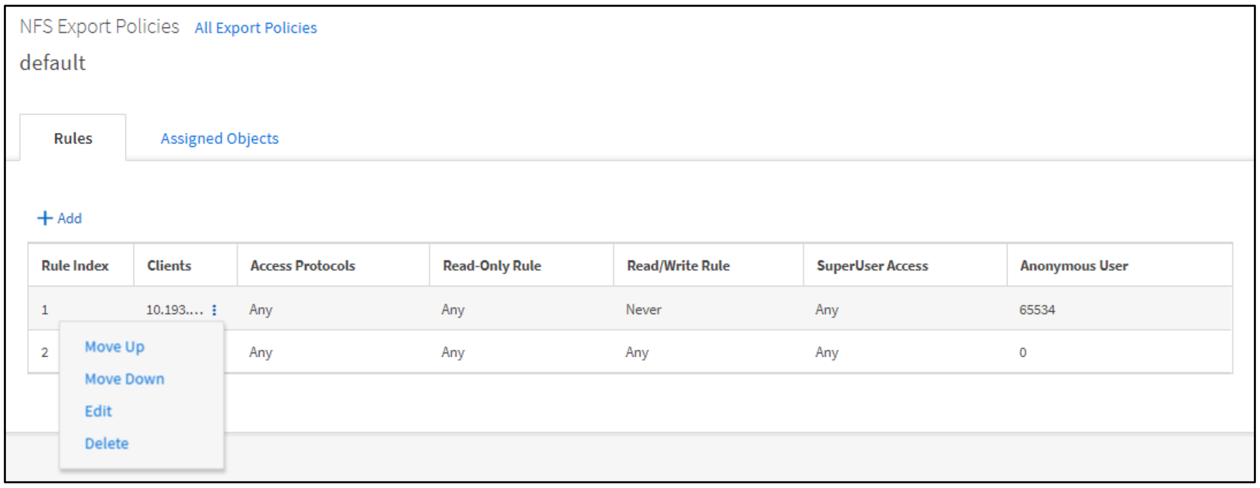
Note: A rule index of 999999999 is an absolute maximum, but Lenovo does not recommend it. Use more sensible numbers for the index.

If a rule index with a higher number (such as 1) is read and has allowed access for a subnet but later a host that is in that subnet is denied access through a rule at a lower index (such as 99), then that host is granted access based on the rule that allows access being read earlier in the policy.

Conversely, if a client is denied access through an export policy rule at a higher index and then allowed access through a global export policy rule later in the policy (such as 0.0.0.0/0 client match), then that client is denied access.

It is possible to reorder the rule index for policy rules with the `export-policy rule setindex` command, or in ThinkSystem Storage Manager for DM Series using "Move Up/Move Down."

Figure 7) Reordering the rule index in ThinkSystem Storage Manager for DM Series.


NFS Export Policies All Export Policies

default

Rules | Assigned Objects

+ Add

Rule Index	Clients	Access Protocols	Read-Only Rule	Read/Write Rule	SuperUser Access	Anonymous User
1	10.193....	Any	Any	Never	Any	65534
2		Any	Any	Any	Any	0

It is important to consider the order of the export policy rules when determining the access that is and is not allowed for clients in ONTAP. If you use multiple export policy rules, be sure that rules that deny or allow access to a broad range of clients do not step on rules that deny or allow access to those same clients. Rule-index ordering factors in when rules are read; higher-number rules override lower-number rules in the index.

Note: When you are using more granular rules (such as for a specific client such as an administrative host), they should be placed higher up in the rule index. Broader access rules should be placed lower. For example, an administrative host rule would be at rule index 1 and a policy for 0.0.0.0/0 would be at index 99.

Export Policy Rules: Clientmatch

The clientmatch option in an export policy rule allows storage administrators to define an access list for mounting NFS exports, as well as a way to control access permissions at a high level after a client is able to mount the export.

Valid entries for the NFS export policy rule clientmatch includethe following:

- IP addresses
- Host names
- Domains
- Subnets
- Netgroups

  Note: It is possible to define multiple comma-separated IP addresses or host names in a single rule, rather than creating unique policy rules for each.

You should consider the following issues:

- When hostnames are used in the clientmatch field or in netgroups, a working DNS server or manual host entries must be available to resolve the hostnames to IP addresses.
- When netgroups are used, an @ sign should be appended to the front of the netgroup to let ONTAP know that you are specifying a netgroup rather than a hostname.
- If relying on name services for name resolution or netgroup lookups, verify that there is a data LIF in the SVM that can reach the necessary name services.
-

Export Policy Rules: Caching

Export policy rules, client hostnames, and netgroup information is all cached in ONTAP to reduce the number of requests made to the cluster. This helps improve performance of requests, as well as alleviating load on networks and name service servers.

Clientmatch Caching

When a clientmatch entry is cached, it is kept local to the SVM and then flushes after the cache timeout period is reached or if the export policy rule table is modified. The default cache timeout period depends on the version of ONTAP and can be verified using the command `export-policy access-cache config show` in admin privilege.

In ONTAP 9.7, these are the default values:

```
TTL For Positive Entries (Secs): 3600
TTL For Negative Entries (Secs): 3600
Harvest Timeout (Secs): 86400
```

To view a specific client in the export policy access-cache, use the following advanced privilege command:

```
cluster::*> export-policy access-cache show -node node-02 -vserver NFS -policy default -address
x.x.x.x

                                        Node: node-02
                                     Vserver: NFS
                                 Policy Name: default
                                  IP Address: x.x.x.x
                    Access Cache Entry Flags: has-usable-data
                                 Result Code: 0
                 First Unresolved Rule Index: -
                      Unresolved Clientmatch: -
              Number of Matched Policy Rules: 1
         List of Matched Policy Rule Indexes: 2
                                Age of Entry: 11589s
                 Access Cache Entry Polarity: positive
Time Elapsed since Last Use for Access Check: 11298s
      Time Elapsed since Last Update Attempt: 11589s
               Result of Last Update Attempt: 0
                List of Client Match Strings: 0.0.0.0/0
```

Hostname/DNS Caching

When a clientmatch is set to a hostname, the name is then resolved to an IP address. This happens based on the order the SVM's name service-switch (ns-switch) uses. For example, if the ns-switch host database is set to `files,dns`, then ONTAP searches for the client match in local host files and then searches DNS.

After a name lookup, ONTAP caches the result in the hosts cache. This cache's settings are configurable and can be queried and flushed from the ONTAP CLI in advanced privilege.

To query the cache, run the following command:

```
cluster::*> name-service cache hosts forward-lookup show -vserver NFS
(vserver services name-service cache hosts forward-lookup show)
                   IP       Address IP                     Create
Vserver   Host     Protocol Family  Address        Source  Time       TTL(sec)
--------- -------- -------- ------- -------------- ------- ---------- --------
NFS       centos7.ntap.local
                   Any      Ipv4    x.x.x.x  dns     3/26/2020  3600
                                                             16:31:11
```

To view the hosts cache settings, run the following command:

```
cluster::*> name-service cache hosts settings show -vserver NFS -instance
  (vserver services name-service cache hosts settings show)

                   Vserver: NFS
         Is Cache Enabled?: true
Is Negative Cache Enabled?: true
              Time to Live: 24h
     Negative Time to Live: 1m
     Is TTL Taken from DNS: true
```

In some cases, if an NFS client’s IP address changes, the hosts entry might need to be flushed to correct access issues.

To flush a host’s cache entry, run the following command:

```
cluster::*> name-service cache hosts forward-lookup delete -vserver NFS ?
     -host      -protocol  -sock-type -flags     -family
```

Netgroup Caching

If you are using netgroups in the clientmatch field for export rules, then ONTAP performs additional work to contact the netgroup name service server to unpack the netgroup information. The netgroup database in ns-switch determines the order in which ONTAP queries for netgroups. In addition, the method ONTAP uses for netgroup support depends on whether netgroup.byhost support is enabled or disabled.

- If netgroup.byhost is disabled, then ONTAP queries the entire netgroup and populates the cache with all netgroup entries. If the netgroup has thousands of clients, then that process could take some time to complete. Netgroup.byhost is disabled by default.
- If netgroup.byhost is enabled, then ONTAP queries the name service only for the host entry and the associated netgroup mapping. This greatly reduces the amount to time needed to query for netgroups, because you do not need to look up potentially thousands of clients.

These entries are added to the netgroup cache, which is found in `vserver services name-service cache` commands. These cache entries can be viewed or flushed, and the timeout values can be configured.

To view the netgroups cache settings, run the following command:

```
cluster::*> name-service cache netgroups settings show -vserver NFS -instance
  (vserver services name-service cache netgroups settings show)

                   Vserver: NFS
         Is Cache Enabled?: true
Is Negative Cache Enabled?: true
              Time to Live: 24h
     Negative Time to Live: 1m
  TTL for netgroup members: 30m
```

When an entire netgroup is cached, it gets placed in the members cache:

```
cluster::*> name-service cache netgroups members show -vserver DEMO -netgroup netgroup1
  (vserver services name-service cache netgroups members show)

            Vserver: DEMO
           Netgroup: netgroup1
              Hosts: sles15-1,x.x.x.x
        Create Time: 3/26/2020 12:40:56
Source of the Entry: ldap
```

When only a single netgroup entry is cached, the ip-to-netgroup and hosts reverse-lookup caches are populated with the entry:

```
cluster::*> name-service cache netgroups ip-to-netgroup show -vserver DEMO -host x.x.x.y
  (vserver services name-service cache netgroups ip-to-netgroup show)
```

```
Vserver    IP Address  Netgroup     Source  Create Time
---------- ----------- ------------ ------- -----------
DEMO       x.x.x.y
                       netgroup1    ldap    3/26/2020 17:13:09

cluster::*> name-service cache hosts reverse-lookup show -vserver DEMO -ip x.x.x.y
  (vserver services name-service cache hosts reverse-lookup show)
Vserver    IP Address      Host                Source Create Time     TTL(sec)
---------- --------------- ------------------- ------ -------------- --------
DEMO       x.x.x.y   centos8-ipa.centos-ldap.local
                                               dns    3/26/2020 17:13:09
                                                                      3600
```

Cache Timeout Modification Considerations

Cache configurations can be modified to different values if needed.

- **Increasing** the timeout values keeps cache entries longer, but might result in inconsistencies in client access if a client changes its IP address (for example, if DHCP is used for client IP addresses and DNS does not get updated or the export rule uses IP addresses).
- **Decreasing** the timeout values flushes the cache more frequently for more up-to-date information, but could add additional load to name service servers and add latency to mount requests from clients.

In most cases, leaving the cache timeout values intact is the best approach.

Exportfs Support

In ONTAP, `exportfs` is replaced by the export-policy and name-service cache commands. When running `exportfs`, the following is seen:

```
  "exportfs" is not supported: use the "vserver export-policy" command.
```

Export Policy Rules: Access Verification

ONTAP offers a command (`export-policy check-access`) that allows you to check an export policy's access rule set against a client's access to help determine if an export policy rule is working properly for pre-deployment as well as troubleshooting. Its functionality is similar to `exportfs -c` functionality. This command uses all normal name service communication and cache interactions that a standard mount from an NFS client would use.

See the following example of export-policy check-access:

```
cluster1::*> vserver export-policy check-access -vserver vs1 -client-ip 1.2.3.4 -volume flex_vol
-authentication-method sys -protocol nfs3 -access-type read

                                        Policy    Policy       Rule
Path                          Policy     Owner     Owner Type  Index Access
----------------------------- ---------- --------- ---------- ------ ----------
/                             default    vs1_root  volume          1 read
/dir1                         default    vs1_root  volume          1 read
/dir1/dir2                    default    vs1_root  volume          1 read
/dir1/dir2/flex1              data       flex_vol  volume         10 read
```

1.8 The Anon User

The anon (anonymous) user ID specifies a UNIX user ID or user name that is mapped to client requests that arrive without valid NFS credentials. This can include the root user. ONTAP determines a user's file access permissions by checking the user's effective UID against the SVM's specified name-mapping and name-switch methods. After the effective UID is determined, the export policy rule is leveraged to determine the access that UID has.

The `–anon` option in export policy rules allows specification of a UNIX user ID or user name that is mapped to client requests that arrive without valid NFS credentials (including the root user). The default value of `–anon`, if not specified in export policy rule creation, is 65534. This UID is normally associated with the user name “nobody” or “nfsnobody” in Linux environments. Lenovo appliances use 65534 as the user “pcuser,” which is generally used for multiprotocol operations, Because of this difference, if using local files and NFSv4, the name string for users mapped to 65534 might not match. This discrepancy might cause files to be written as the user specified in the `/etc/idmapd.conf` file on the client (Linux) or `/etc/default/nfs` file (Solaris), particularly when using multiprotocol (CIFS and NFS) on the same datasets.

1.9 The Root User

The "root" user must be explicitly configured in ONTAP to specify which machine has "root" access to a share, or else "anon=0” must be specified. Alternatively, the `-superuser` option can be used if more granular control over root access is desired. If these settings are not configured properly, "permission denied" might be encountered when accessing an NFS share as the "root" user (0). If the `–anon` option is not specified in export policy rule creation, the root user ID is mapped to the "nobody" user (65534).

AUTH Types

When an NFS client authenticates, an AUTH type is sent. An AUTH type specifies how the client is attempting to authenticate to the server and depends on client-side configuration. Supported AUTH types include:

- **AUTH_NONE/AUTH_NULL.** This AUTH type specifies that the request coming in has no identity (NONE or NULL) and is mapped to the anon user. See http://www.ietf.org/rfc/rfc1050.txt and http://www.ietf.org/rfc/rfc2623.txt for details.
- **AUTH_SYS/AUTH_UNIX.** This AUTH type specifies that the user is authenticated at the client (or system) and comes in as an identified user. See http://www.ietf.org/rfc/rfc1050.txt and http://www.ietf.org/rfc/rfc2623.txt for details.
- **AUTH_RPCGSS**. This is kerberized NFS authentication.

There are several ways to configure root access to an NFS share. For examples, see “Examples of Controlling the Root User.”

1.10 Limiting Access to the SVM Root Volume

By default, when an SVM is created, the root volume is configured with 755 permissions and owner:group of root (0): root (0). This means that:

- The user root (0) has effective permissions of 7, or Full Control.
- The “group” and “others” permission levels are set to 5, which is Read & Execute.

When this is configured, everyone who accesses the SVM root volume can list and read junctions mounted below the SVM root volume, which is always mounted to “/” as a junction-path. In addition, the default export policy rule that is created when an SVM is configured using Storage Manager or `vserver setup` commands permits user access to the SVM root.

See the following example of the default export policy rule created by SVM setup:

```
cluster::> export-policy rule show -vserver nfs_svm -policyname default -instance
  (vserver export-policy rule show)

                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: none
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true
```

In the preceding export policy rule, all clients have `any` RO and RW access. Root is squashed to `anon`, which is set to `65534`.

For example, if an SVM has three data volumes, all would be mounted under "/" and could be listed with a basic `ls` command by any user accessing the mount.

```
# mount | grep /mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /mnt
# ls
nfs4  ntfs  unix
```

In some environments, this behavior might be undesirable, because storage administrators might want to limit visibility to data volumes to specific groups of users. Although read and write access to the volumes themselves can be limited on a per-data-volume basis using permissions and export policy rules, users can still see other paths using the default policy rules and volume permissions.

To limit the ability to users to be able to list SVM root volume contents (and subsequent data volume paths) but still allow the traversal of the junction paths for data access, the SVM root volume can be modified to allow only root users to list folders in SVM root. To do this, change the UNIX permissions on the SVM root volume to `0711` using the volume modify command:

```
cluster::> volume modify -vserver nfs_svm -volume rootvol -unix-permissions 0711
```

After this is done, root still has Full Control using the `7` permissions, because it is the owner. `Group` and `others` get `Execute` permissions as per the `1` mode bit, which only allows them to traverse the paths using cd.

When a user who is not the root user attempts `ls`, that user has access denied:

```
sh-4.1$ ls
ls: cannot open directory .: Permission denied
```

In many cases, NFS clients log into their workstations as the root user. With the default export policy rule created by Storage Manager and vserver setup, root access is limited:

```
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# ls -la
ls: cannot open directory .: Permission denied
```

This is because the export policy rule attribute `superuser` is set to `none`. If root access is desired by certain clients, this can be controlled by adding export policy rules to the policy and specifying the host IP, name, netgroup, or subnet in the clientmatch field. When creating this rule, list it ahead of any rule that might override it, such as a clientmatch of `0.0.0.0/0` or `0/0`, which is `all hosts`.

See the following example of adding an administrative host rule to a policy:

```
cluster::> export-policy rule create -vserver nfs_svm -policyname default -clientmatch x.x.x.x -
rorule any -rwrule any -superuser any -ruleindex 1

cluster::> export-policy rule show -vserver nfs_svm -policyname default -ruleindex 1
  (vserver export-policy rule show)

                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
```

```
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true


cluster::> export-policy rule show -vserver nfs_svm -policyname default
  (vserver export-policy rule show)
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
nfs_svm   default         1       any      x.x.x.x        any
nfs_svm   default         2       any      0.0.0.0/0             any
2 entries were displayed.
```

Now the client is able to see the directories as the root user.

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.x  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# ls
nfs4  ntfs  unix
```

Other clients are not able to list contents as root.

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# mount | grep mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# ls /mnt
ls: cannot open directory .: Permission denied
```

For more information about export policy rules and their effect on the root user, review the “Root User” section of this document.

For more information about mode bits, see this page on UNIX permissions help.

1.11 Mapping All UIDs to a Single UID (squash_all)

In some cases, storage administrators might want to control which UID (such as root) some or all users map to when coming in through NFS to a UNIX-security-style volume. If a volume has NTFS security style, doing so is as simple as setting a default Windows user in the NFS server options. However, when the volume is UNIX security style, no name mapping takes place when coming in from NFS clients. To control this situation, you can create an export policy rule.

Squashing All UIDs to 65534

The following export policy rule example shows how to force all UIDs (including root) coming into the system from a specific subnet to use the 65534 UID. This rule can be used to create `guest` access policies for users to limit access. This is done with the RO, RW, and superuser authentication type of `none`, anon value of `65534`, and the clientmatch value specifying the subnet:

```
                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
```

```
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.225.0/24
                             RO Access Rule: none
                             RW Access Rule: none
User ID To Which Anonymous Users Are Mapped: 65534
                    Superuser Security Types: none
                 Honor SetUID Bits in SETATTR: true
                    Allow Creation of Devices: true
```

Making All UIDs Root

The following export policy rule example shows how to force all UIDs (including root) coming into the system from a specific subnet to use the UID associated with root (0). This rule can be used to allow full access to users in a specific subnet to avoid overhead on permissions management. This access is enabled with the RO and RW authentication type of `none`, superuser value of `none`, anon value of `0`, and the clientmatch value specifying the subnet.

See the following example:

```
                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.225.0/24
                             RO Access Rule: none
                             RW Access Rule: none
User ID To Which Anonymous Users Are Mapped: 0
                    Superuser Security Types: none
                 Honor SetUID Bits in SETATTR: true
                    Allow Creation of Devices: true
```

1.12 Special Character Considerations

Most common text characters in Unicode (when they are encoded with UTF-8 format) use encoding that is equal to or smaller than three bytes. This common text includes all modern written languages, such as Chinese, Japanese, and German. However, with the popularity of special characters such as the emoji, some UTF-8 character sizes have grown beyond 3 bytes. For example, a trophy symbol is a character that requires 4 bytes in UTF-8 encoding.

Special characters include, but are not limited to, the following:

- Emojis
- Music symbols
- Mathematical symbols

When a special character is written to a FlexGroup volume, the following behavior occurs:

```
# mkdir /flexgroup4TB/🏆
mkdir: cannot create directory ‘/flexgroup4TB/\360\237\217\206’: Permission denied
```

In the preceding example, `\360\237\217\206` is hex `0xF0 0x9F 0x8F 0x86` in UTF-8, which is a trophy symbol.

ONTAP software did not natively support UTF-8 sizes that are greater than three bytes in NFS. To handle character sizes that exceed three bytes, ONTAP places the extra bytes into an area in the operating system known as `bagofbits`. These bits are stored until the client requests them. Then the client interprets the character from the raw bits. FlexVol and FlexGroup support `bagofbits` in ONTAP releases.

Also, ONTAP has an event management system message for issues with `bagofbits` handling.

```
Message Name: wafl.bagofbits.name
```

```
Severity: ERROR

Corrective Action: Use the "volume file show-inode" command with the file ID and volume name
information to find the file path. Access the parent directory from an NFSv3 client and rename
the entry using Unicode characters.

Description: This message occurs when a read directory request from an NFSv4 client is made to a
Unicode-based directory in which directory entries with no NFS alternate name contain non-Unicode
characters.
```

Support for utf8mb4 Volume Language

As mentioned before, special characters might exceed the supported three bytes UTF-8 encoding that is natively supported. ONTAP then uses the `bagofbits` functionality to allow these characters to work.

This method for storing inode information is not ideal, so starting in ONTAP 9.5, utf8mb4 volume language support was added. When a volume uses this language, special characters that are four bytes in size is stored properly and not in `bagofbits`.

Volume language is used to convert names sent by NFSv3 clients to Unicode, and to convert on-disk Unicode names to the encoding expected by NFSv3 clients. In legacy situations in which NFS hosts are configured to use non-UTF-8 encodings, you will want to use the corresponding volume language. Use of UTF-8 has become almost universal these days, so the volume language is likely to be UTF-8.

NFSv4 requires use of UTF-8, so there is no need to use non-UTF-8 encoding for NFSv4 hosts. Similarly, CIFS uses Unicode natively, so it works with any volume language. However, use of utf8mb4 is recommended because files with Unicode names above the basic plane are not converted properly on non-utf8mb4 volumes.

2 NFS Version Considerations

2.1 NFSv3 Considerations

This section covers functionality, known issues and considerations with NFSv3 in ONTAP.

What Happens During NFSv3 Mounts?

The following occurs when mounting a file system over NFSv3:

1. A Remote Procedure Call (RPC) is made to port 111 (portmapper) of the NFS server to attempt a TCP connection through the portmapper.
2. When the RPC has been acknowledged, portmapper issues a GETPORT call to port 111 of the NFS server data LIF to obtain which port NFS is allowed on.
3. The NFS server returns the port 2049 (NFS) to the client.
4. The client then closes the connection to port 111.
5. A new RPC call is made to port 2049 of the NFS server data LIF.
6. The NFS server returns the call successfully, and the client sends an NFS NULL call to port 2049 of the NFS server's data LIF. This checks whether the parent volume export policy rule allows access to the mount. In this case, the parent volume is mounted to /, or the SVM root.
7. The NFS NULL call is returned successfully, and the client proceeds with the mount attempt.
8. Portmapper sends another GETPORT call to the NFS server's data LIF asking for the mountd port and provides the credentials, NFS version number, and whether the mount uses TCP or UDP.
9. The cluster checks the NFS settings and verifies whether the credentials supplied are allowed to mount based on the exported volume's export policy rules. Name service servers (such as DNS, NIS,

LDAP) are contacted as necessary. Caches are populated. If the NFS version or TCP/UDP is not allowed, the client reports the error.

10. The NFS server replies successfully if the version provided is supported and if the mount can use the specified TCP or UDP connection. It also replies if the AUTH security provider is supported (AUTH_SYS or AUTH_GSS, for example).
11. When the GETPORT call passes, the client issues a V3 MNT call to the junction path specified in the mount command through port 635 (mountd) of the NFS server data LIF.
12. ONTAP looks for the provided path to the export. If the entry exists, the cluster gathers the file handle information, which is unique to each volume.
13. The NFS server returns the file handle to the client, as well as replies which AUTH varieties are supported by the export policy rule. If the AUTH variety provided by the server matches what the client sent, the mount succeeds.
14. Portmapper from the client then sends another GETPORT call for NFS, this time providing the client's host name.
15. The NFS server replies with port 2049, and the call succeeds.
16. Another NFS NULL call is made from the client over port 2049 of the NFS data LIF and is acknowledged by the NFS server.
17. A series of NFS packets with FSINFO and PATHCONF information is traded between the client and the NFS server.

Effects of File System ID (FSID) Changes in ONTAP

NFS uses a file system ID (FSID) when interacting between client and server. This FSID lets the NFS client know where data lives in the NFS server's file system. Because ONTAP can span multiple file systems across multiple nodes by way of junction paths, this FSID can change depending on where data lives. Some older Linux clients can have problems differentiating these FSID changes, resulting in failures during basic attribute operations, such as `chown` and `chmod`.

If you disable the FSID change with NFSv3, be sure to enable the `-v3-64bit-identifiers` option in ONTAP 9 (see "64-Bit File Identifiers"). But keep in mind that this option could affect older legacy applications that require 32-bit file IDs.

For information on how this affects high file count environments, see "How FSIDs Operate with SVMs in High-File-Count Environments."

How FSIDs Operate with Lenovo Snapshot Copies

When a Lenovo Snapshot™ copy of a volume is created, a copy of a file's inodes is preserved in the file system for access later. The file theoretically exists in two locations.

With NFSv3, even though there are two copies of essentially the same file, the FSIDs of those files are not identical. FSIDs of files are formulated by using a combination of Lenovo WAFL inode numbers, volume identifiers, and Snapshot IDs. Because every Snapshot copy has a different ID, every Snapshot copy of a file has a different FSID in NFSv3, regardless of the setting of the `-v3-fsid-change` option. The NFS RFC specification does not require FSIDs for a file to be identical across file versions.

FSID Changes with Storage Virtual Machine Disaster Recovery (SVM DR)

ONTAP 8.3.1 introduced a new feature to enable disaster recovery for entire SVMs called SVM DR. This feature is covered in TR-4015: SnapMirror Configuration and Best Practices Guide.

When SVM DR is used with NFS exports in versions prior to ONTAP 9.0, the FSID of those exports changes, and clients have to remount the exports on the destination system. Otherwise, the clients show `stale` for NFS operations on those mounts. If the mount's FSID should be preserved by the SVM DR relationship, then the destination SVM would need to be created with the `-is-msid-preserve` option set to `true` in diag privilege mode. When this option is set, SnapMirror relationships used in SVM DR show `-msid-preserve` as `true` in their `snapmirror show` output. This should be used with caution, because SVM DR updates are asynchronous. The source SVM should be confirmed as down before attempting to write to the destination SVM with the same FSID.

NFSv3 Security Considerations

NFSv3 is considered less secure than NFSv4.x, but that does not mean there are not security measures you can put into place. The following sections cover some of the NFSv3 security considerations as the apply to ONTAP.

NFSv3 Export Rules

One way to secure NFS mounts is through the export rules. Export rules allow storage administrators to fence access to volumes or qtrees to specific clients in the environment. For example, if an NFS client is not listed in an export rule that allows access, then that client is not able to mount the export.

Export rules also provide ways to limit whether a specific client is allowed to read, write, setUID permissions, and access to change file ownerships. Rules can define which specific NFS versions are allowed to mount, as well as what security flavors are allowed (such as AUTH_SYS or Kerberos).

Security Limitations of Export Rules

Export rules are just one way to secure NFS mounts, but by themselves, export rules do not offer enough security to approach enterprise standards. Export rules depend on clientmatch entries to limit access. This means someone could spoof an IP address in the clientmatch list and gain access to the export without any other authentication requirements, unless other security concepts, such as Kerberos authentication, are also required by the export rule.

NFSv3 Permissions

NFSv3 follow the NFS mode bits definitions in RFC 1813 starting on page 22. offers basic file and folder permissions to control access to end users. These permissions

Security Limitations of NFSv3 Permissions

Mode bit permissions only cover owner, group, and everyone else in the semantics – meaning that there are no granular user access controls in place for basic NFSv3. ONTAP does not support POSIX ACLs, so granular ACLs are only possible in the following scenarios with NFSv3:

- NTFS security style volumes (CIFS server required) with valid UNIX to Windows user mappings.
- NFSv4.x ACLs applied using an admin client mounting NFSv4.x to apply ACLs.

Additionally, mode bits do not provide the same level of granularity in permissions that NTFS or NFSv4.x ACLs provide. Table 6 compares the permission granularity between NFSv3 mode bits and NFSv4.x ACLs. For details on NFSv4.x ACLs, see the NFSv4 Access Control Lists Linux man page.

Table 5) NFSv3 mode bits versus NFSv4.x ACL granularity.

NFSv3 Mode Bits	NFSv4.x ACLs
• Set user ID on execution. • Set group ID on execution. • Save swapped text (not defined in POSIX). • Read permission for owner. • Write permission for owner.	• ACE types (Allow/Deny/Audit) • Inheritance flags – directory-inherit – file-inherit – no-propagate-inherit – inherit-only

NFSv3 Mode Bits	NFSv4.x ACLs
• Execute permission for owner on a file. Or lookup (search) permission for owner in directory. • Read permission for group. • Write permission for group. • Execute permission for group on a file. Or lookup (search) permission for group in directory. • Read permission for others. • Write permission for others. • Execute permission for others on a file. Or lookup (search) permission for others in directory.	• Permissions – read-data (files) / list-directory (directories) – write-data (files) / create-file (directories) – append-data (files) / create-subdirectory (directories) – execute (files) / change-directory (directories) – delete – delete-child – read-attributes – write-attributes – read-named-attributes – write-named-attributes – read-ACL – write-ACL – write-owner – synchronize

Finally, NFS group membership (in both NFSv3 and NFSV4.x) is limited to a maximum of 16 as per the RPC packet limits. Options exist in ONTAP to help extend beyond those limits in "Auxiliary GIDs – Addressing the 16 GID Limitation for NFS."

NFSv3 User and Group IDs

NFSv3 user and group IDs come across the wire as numeric IDs, rather than names. ONTAP does no name resolution for these numeric IDs in UNIX security style volumes and qtrees with no NFSv4.x ACLs present. With NTFS security styles and NFSv4.x ACLs, ONTAP attempts to resolve a numeric ID to a valid Windows user to negotiate access. For more information, see "Viewing and Managing NFS Credentials."

Security Limitations of NFSv3 User and Group IDs

The client and server never have to confirm that the user attempting a read or write with a numeric ID is actually a valid user. It is just implicitly trusted. This opens the filesystem up to potential breaches simply by spoofing a known numeric ID. To prevent this from creating a security hole with NFSv3, implementing Kerberos for NFS forces users to authenticate with a username and password or keytab file to get a Kerberos ticket to allow access into a mount. Using this in conjunction with NFSv4.x or NTFS ACLs can help mitigate performance risk with NFSv3 numeric IDs.

Advanced NFSv3 Security Concepts

This section covers two advanced security concepts for NFSv3. The term "advanced" refers to the concepts being either non-standard or containing multiple steps for configuration.

Using ACLs with NFSv3

As mentioned above, POSIX ACLs are not supported for use with NFSv3 in ONTAP. However, there are two options for controlling NFSv3 user and group access with ACLs.

NTFS ACLs

ONTAP provides a way to present datasets in the same volume to multiple NAS protocols simultaneously. For example, a volume can host clients through NFS and SMB, while protocol-specific features such as locking and permissions are negotiated by ONTAP to provide data resiliency and security.

NFSv3 client access can be managed through the file and folder permissions as configured from Windows clients when using ONTAP for both CIFS/SMB and NFS access. When a CIFS/SMB server is created and volume security styles are using NTFS permission semantics, NFSv3 clients adheres to those NTFS ACLs by way of UNIX to Windows user-name mapping that ONTAP requires for NFS access to volumes containing NTFS ACLs. For more information on this functionality, see “Multiprotocol NAS.”

NFSv4.x ACLs

NFSv4.x is supported in ONTAP along with NFSv3 and offers granular file and folder permissions in the same vein as NTFS ACLs. ONTAP also allows storage administrators to set NFSv4.x ACLs through an admin client that is honored by NFSv3 clients. When an NFSv3 client attempts access to a mount with NFSv4.x ACLs, ONTAP forces the numeric user ID to resolve to a valid UNIX user through the NFSv4.x name mapping requirements so that the NFSv4.x ACL can be honored properly. For more information on NFSv4.x ACLs, see “NFSv4.x Access Control Lists (ACLs).”

Using NFS Kerberos with NFSv3

You can use Kerberos with NFSv3, but you should consider these caveats:

- NFSv3 is made up of several ancillary protocols in addition to the NFS protocol. When Kerberos is used with NFSv3, only the NFS packets use Kerberos. Mount, portmap, and so on do not.
- Export policy rules with NFSv3 and Kerberos should use both sys and krb5* in their settings to reflect the lack of Kerberos support for ancillary protocols.

2.2 NFSv4.x Considerations

This section covers functionality, known issues, and considerations with NFSv4.x in ONTAP.

Enabling NFSv4.x

To start using NFSv4.x with ONTAP in your environment, consider this list of steps to perform or review:

- Set the NFS ID domain string in `/etc/idmap.conf` to the same value as the `-v4-id-domain` option on the NFS SVM.
- Make sure that the users and groups accessing the NFSv4.x client also exist (or can be queried from) the ONTAP SVM. These users and groups must have the same names and case sensitivity.
  - For example, `john@DOMAIN.COM` on the NFS client needs to match `john@DOMAIN.COM` on the ONTAP NFS server.
- If using LDAP or NIS for UNIX identities, make sure that the user and group lookups return the expected IDs and group members.
- Export policy rules for volumes should be set to allow NFSv4 as a protocol.
- Data LIFs in the environment should have NFS as an allowed protocol (`net int show -fields allowed-protocols`).
- The desired NFSv4.x version(s) should be enabled. If only version 4.1 is desired, enable only that version and disable version 4.0.
- If NFSv4.x ACLs are desired, you must enable them for the specific NFSv4.x version you wish to use them with. (`-v4.0-acl`, `-v4.1-acl`)
- Clients negotiate the highest NFS version the NFS server supports. If some clients require NFSv3, they need to change how they mount.

Advantages of Using NFSv4.x

The following are some advantages to using NFSv4.x in your environment:

- Firewall-friendly because NFSv4 uses only a single port (2049) for its operations
- Advanced and aggressive cache management, like delegations in NFSv4.x
- Strong RPC security choices that employ cryptography
- Internationalization
- Compound operations
- Works only with TCP
- Stateful protocol (not stateless like NFSv3)
- Kerberos configuration for efficient authentication mechanisms
  - AES support
- Migration (for dNFS) using referrals
- Support of access control that is compatible with UNIX and Windows
- String-based user and group identifiers
- Parallel access to data through pNFS (does not apply for NFSv4.0)

It is important that you treat every specific use case differently. NFSv4.x is not ideal for all workload types. Be sure to test for desired functionality and performance before rolling out NFSv4.x en masse.

Note: ONTAP currently does not support NFSv4.x session trunking.

Performance Enhancements for NFSv4.x Operations

Lenovo is constantly striving to improve performance for each ONTAP release. NFSv4.x performance is a priority, as it is the future for NFS. Performance enhancements are listed below, along with the ONTAP release they were introduced. For the best possible performance with NFS, always run the latest patched ONTAP release available.

NFSv4.x Fastpath

NFS fastpath was introduced to potentially improve NFSv4 performance for reads and writes. This improvement is made by bypassing the internal processing of NFSv4 packets into ONTAP-centric packets when the data request is made on a LIF that is local to the node hosting the volume. When combined with other features such as pNFS or referrals, localized data can be guaranteed for each read and write request, thus allowing consistent use of the NFSv4 fastpath. NFSv3 has always had an NFS fastpath concept. NFS fastpath is enabled by default.

NFSv4.x Multithreaded Operations

Multiprocessor support was added for NFSv4.x read and write operations. Metadata operations, however, still use a single threaded approach. In previous releases, NFSv4.x read and write operations were single threaded, thus allowing a potential bottleneck at the CPU for the protocol domain. Using multiple processors for read and write operations can greatly increase throughput on Lenovo systems that contain more than one CPU for NFSv4.x workloads that are read and write heavy.

Note: NFSv3 has always used multiple processors for reads and writes. NFSv3 also uses multiple processors for metadata operations.

NFSv4.x – Performance Improvements for Streaming Workload Types

ONTAP introduced an improvement to how streaming workload types like Vmware, Oracle and SAP HANA performed with NFSv4.1 by adding large I/O support. This allows NFS (both v3 and 4.x) to use up to 1MB for both reads and writes.

NFSv4.x – Performance Improvements for Metadata Workloads (ONTAP 9.5)

Many improvements were added into ONTAP 9.5 to improve metadata workloads, including the following:

- NFSv4.0 cache IO support
- Optimizations of NFSv4.x metadata operations
- Increased caching
- Improved locking performance
- Increased StorePool limits

NFSv4.x FlexGroup Volume Support (ONTAP 9.7)

In addition to the metadata performance improvements added in ONTAP 9.5, FlexGroup volume support can help increase metadata workload performance as well, by way of parallelization of file ingest operations.

2.3 NFSv4.0

The NetApp ONTAP NFSv4.x implementation provides the following features:

- **Write Order.** The implementation provides the capability to write data blocks to shared storage in the same order as they occur in the data buffer.
- **Synchronous Write Persistence.** Upon return from a synchronous write call, ONTAP guarantees that all the data has been written to durable, persistent storage.
- **Distributed File Locking.** The implementation provides the capability to request and obtain an exclusive lock on the shared storage, without assigning the locks to two servers simultaneously.
- **Unique Write Ownership.** ONTAP guarantees that the file lock is the only server process that can write to the file. After ONTAP transfers the lock to another server, pending writes queued by the previous owner fail.

Transitioning from NFSv3 to NFSv4.x: Considerations

This section covers some considerations that need to be addressed when migrating from NFSv3 to NFSv4.x. When choosing to use NFSv4.x after using NFSv3, you cannot simply turn it on and have it work as expected. There are specific items to address, such as:

- Domain strings/ID mapping
- Storage failover considerations
- Name services
- Firewall considerations
- Export policy rule considerations
- Client support
- NFSv4.x features and functionality

For an in-depth look at the NFSv4.x protocol, including information about NFSv4.2, see the SNIA overview of NFSv4.

NFSv4.x ID Domain Mapping

While customers prepare to migrate their existing setup and infrastructure from NFSv3 to NFSv4, some environmental changes must be made before moving to NFSv4. One of them is "id domain mapping."

When NFSv3 clients access a mount, the numeric ID is passed to the NFS server and no further ID lookups are needed, provided all portions of the access are using UNIX security with NFSv3 semantics.

When NFSv4.x clients access a mount, a “name string” is passed from the client to the server that contains the user or group principal (name@DOMAIN.COM).

The server then attempts to verify that it knows about a principal with the same exact name (case sensitive) through its name service and NFS server configuration. If that name string does not exist on both client and server, then the user is “squashed” to a “nobody” user as defined in the client NFSv4.x configuration file. This provides an extra layer of security over NFSv3.

Bypassing the name string – Numeric IDs

In some cases, storage administrators might not want to use NFSv4.x for its security, but instead for its locking mechanisms. Or they might not want to deal with the configuration overhead of client/server name strings.

In those cases, there is an option called `v4-numeric-ids`. With this option enabled, if the client does not have access to the name mappings, numeric IDs can be sent in the user-name and group-name fields. The server accepts them and treats them as representing the same user as would be represented by a v2/v3 UID or GID having the corresponding numeric value. If the client does have a name string that matches, then the client uses the name string rather than the numeric ID.

Essentially, this option makes NFSv4.x behave more like NFSv3. The default value of this option is “enabled.”

Storage Failover Considerations

NFSv4.x uses a completely different locking model than NFSv3. Locking in NFSv4.x is a lease-based model that is integrated into the protocol rather than separated as it is in NFSv3 (NLM). From the ONTAP documentation:

```
In accordance with RFC 3530, Data ONTAP "defines a single lease period for all state held by an
NFS client. If the client does not renew its lease within the defined period, all states
associated with the client's lease may be released by the server." The client can renew its lease
explicitly or implicitly by performing an operation, such as reading a file.
Furthermore, Data ONTAP defines a grace period, which is a period of special processing in which
clients attempt to reclaim their locking state during a server recovery.
```

Table 6) NFSv4.x lock terminology.

Term	Definition (per **RFC 3530**)
Lease	The time period in which ONTAP irrevocably grants a lock to a client
Grace period	The time period in which clients attempt to reclaim their locking state from ONTAP during server recovery
Lock	Refers to both record (byte-range) locks as well as file (share) locks unless specifically stated otherwise

For more information about NFSv4.x locking, see the section in this document on “NFSv4 Locking.” Because of this new locking methodology, as well as the statefulness of the NFSv4.x protocol, storage failover operates differently as compared to NFSv3. For more information, see the section in this document called “Nondisruptive Operations with NFS.”

Name Services

When deciding to use NFSv4.x, it is a best practice to centralize the NFSv4.x users in name services such as LDAP or NIS. Doing so allows all clients and ONTAP NFS servers to leverage the same resources and guarantees that all names, UIDs, and GIDs are consistent across the implementation.

Firewall Considerations

NFSv3 required several ports to be opened for ancillary protocols such as NLM, NSM, and so on in addition to port 2049. NFSv4.x requires only port 2049. If you want to use NFSv3 and NFSv4.x in the same environment, open all relevant NFS ports. These ports are referenced in “Default NFS Ports in ONTAP.” For more information and guidance on firewalls, see “NFS Security Best Practices.”

Volume Language Considerations

In ONTAP, volumes can have specific languages set. This capability is intended to be used for internationalization of file names for languages that use characters not common to English, such as Japanese, Chinese, German, and so on. When using NFSv4.x, RFC 3530 states that UTF-8 is recommended.

```
11.  Internationalization

   The primary issue in which NFS version 4 needs to deal with
   internationalization, or I18N, is with respect to file names and
   other strings as used within the protocol.  The choice of string
   representation must allow reasonable name/string access to clients
   which use various languages.  The UTF-8 encoding of the UCS as
   defined by [ISO10646] allows for this type of access and follows the
   policy described in "IETF Policy on Character Sets and Languages",
   [RFC2277].
```

When changing a volume's language, every file in the volume must be accessed after the change to make sure that they all reflect the language change. Use a simple `ls -lR` to access a recursive listing of files. If the environment is a high file count environment, consider using XCP to scan the files quickly.

Export Policy Rules

If an environment was configured for NFSv3 and the export policy rule option `–protocol` was limited to allow NFSv3 only, then the option needs to be modified to allow NFSv4. Additionally, policy rules could be configured to allow access only to NFSv4.x clients.

Example:

```
cluster::> export-policy rule modify -policy default -vserver NAS -protocol nfs4
```

For more information, consult the product documentation for your specific version of ONTAP.

Client Considerations

When you use NFSv4.x, clients are as important to consider as the NFS server. Contact the OS vendor for specific questions about NFSv4.x configuration.

Follow the client considerations below when implementing NFSv4.x. (Other considerations might be necessary.)

- NFSv4.x is supported.
- The fstab file and NFS configuration files are configured properly. When mounting, the client negotiates the highest NFS version available with the NFS server. If NFSv4.x is not allowed by the client or fstab specifies NFSv3, then NFSv4.x is not used at mount.
- The idmapd.conf file is configured with the proper settings, including the correct NFSv4.x ID domain.
- The client either contains identical users/groups and UID/GID (including case sensitivity) in local passwd and group files or uses the same name service server as the NFS server/ONTAP SVM.
- If using name services on the client, verify that the client is configured properly for name services (nsswitch.conf, ldap.conf, sssd.conf, and so on) and the appropriate services are started, running, and configured to start at boot.

- The NFSv4.x service is started, running, and configured to start at boot.

Note: TR-4073: Secure Unified Authentication covers some NFSv4.x and name service considerations in detail as they pertain to clients.

NFSv4 Features and Functionality

NFSv4.x is the next evolution of the NFS protocol and enhances NFSv3 with new features and functionality, such as referrals, delegations, pNFS, and so on. These features are covered throughout this document and should be factored into any design decisions for NFSv4.x implementations.

NFSv4 User ID Mapping

As previously mentioned in this document (in the "NFSv4.x ID Domain Mapping" section), NFSv4.x clients and servers attempt to map user ID domain strings for added security. If numeric IDs are preferred, ONTAP has an NFS option (`-v4-numeric-ids`) that can avoid requirements for name strings.

NFSv4.x Access Control Lists (ACLs)

The NFSv4.x protocol can provide access control in the form of Access Control Lists (ACLs), which are similar in concept to those found in CIFS. An NFSv4 ACL consists of individual Access Control Entries (ACEs), each of which provides an access control directive to the server. ONTAP defaults to 400 ACEs and supports a maximum of 1,024 ACEs with a configurable NFS option (`-v4-acl-max-aces`).

Benefits of Enabling NFSv4 ACLs

The benefits of enabling NFSv4 ACLs include the following:

- Granular control of user access to files and directories
- Better NFS security
- Improved interoperability with CIFS
- Removal of the NFS limitation of 16 groups per user with AUTH_SYS security
  - ACLs bypass the need for GID resolution, which effectively removes the GID limit.

Compatibility Between NFSv4 ACLs and SMB Clients

NFSv4 ACLs are different from Windows file-level ACLs (NTFS ACLs) but carry similar functionality. However, in multiprotocol NAS environments, if NFSv4.x ACLs are present, clients using SMB2.0 and later will not be able to view ACLs from Windows security tabs – even if the NFS option `ntacl-display-permissive-perms` and .the CIFS/SMB option `is-unix-nt-acl-enabled` are set.

See bug 928026 for details.

How NFSv4 ACLs Work

When a client sets an NFSv4 ACL on a file during a SETATTR operation, the Lenovo storage system sets that ACL on the object, replacing any existing ACL. If there is no ACL on a file, then the mode permissions on the file are calculated from OWNER@, GROUP@, and EVERYONE@. If there are any existing SUID/SGID/STICKY bits on the file, they are not affected.

When a client gets an NFSv4 ACL on a file during the course of a GETATTR operation, the Lenovo system reads the NFSV4 ACL associated with the object, constructs a list of ACEs, and returns the list to the client. If the file has an NT ACL or mode bits, then an ACL is constructed from mode bits and is returned to the client.

Access is denied if a DENY ACE is present in the ACL; access is granted if an ALLOW ACE exists. However, access is also denied if neither of the ACEs is present in the ACL.

A security descriptor consists of a Security ACL (SACL) and a Discretionary ACL (DACL). When NFSv4 interoperates with CIFS, the DACL is one-to-one mapped with NFSv4 and CIFS. The DACL consists of the ALLOW and the DENY ACEs.

If a basic chmod is run on a file or folder with NFSv4.x ACLs set, the ACLs is removed unless the NFS option `v4-acl-preserve` is enabled.

A client using NFSv4 ACLs can set and view ACLs for files and directories on the system. When a new file or subdirectory is created in a directory that has an ACL, the new file or subdirectory inherits all ACEs in the ACL that have been tagged with the appropriate inheritance flags. For access checking, CIFS users are mapped to UNIX users. The mapped UNIX user and that user's group membership are checked against the ACL.

If a file or directory has an ACL, that ACL is used to control access no matter which protocol—NFSv3, NFSv4, or CIFS—is used to access the file or directory. The ACL is also used even if NFSv4 is no longer enabled on the system.

Files and directories inherit ACEs from NFSv4 ACLs on parent directories (possibly with appropriate modifications) as long as the ACEs have been tagged with the correct inheritance flags.

When a file or directory is created as the result of an NFSv4 request, the ACL on the resulting file or directory depends on whether the file creation request includes an ACL or only standard UNIX file access permissions. The ACL also depends on whether the parent directory has an ACL.

- If the request includes an ACL, that ACL is used.
- If the request includes only standard UNIX file access permissions and the parent directory does not have an ACL, the client file mode is used to set standard UNIX file access permissions.
- If the request includes only standard UNIX file access permissions and the parent directory has a noninheritable ACL, a default ACL based on the mode bits passed into the request is set on the new object.
- If the request includes only standard UNIX file access permissions but the parent directory has an ACL, the ACEs in the parent directory's ACL are inherited by the new file or directory as long as the ACEs have been tagged with the appropriate inheritance flags.

Note: A parent ACL is inherited even if `-v4.0-acl` is set to `off`.

NFSv4 ACL Behavior with umask and ACL Inheritance

NFSv4 ACLs provide the ability to offer ACL inheritance. ACL inheritance means that files or folders created beneath objects with NFSv4 ACLs set can inherit the ACLs based on the configuration of the ACL inheritance flag.

Umask is used to control the permission level at which files and folders are created in a directory. For more information, see the section called "Umask."

By default, ONTAP allows umask to override inherited ACLs, which is expected behavior as per RFC 5661. To adjust the behavior ACL inheritance with umask, you can enable the option `-v4-inherited-acl-preserve`.

ACL Formatting

NFSv4.x ACLs have specific formatting. The following is an ACE set on a file:

```
A::ldapuser@domain.lenovo.com:rwatTnNcCy
```

The preceding follows the ACL format guidelines of:

```
type:flags:principal:permissions
```

A type of “A” means allow. The flags are not set in this case, because the principal is not a group and does not include inheritance. Also, because the ACE is not an AUDIT entry, there is no need to set the audit flags. For more information about NFSv4.x ACLs, see http://linux.die.net/man/5/nfs4_acl.

If an NFSv4.x ACL is not set properly, the ACL might not behave as expected, or the ACL change might fail to apply and throw an error.

Sample errors include:

```
Failed setxattr operation: Invalid argument
Scanning ACE string 'A::user@rwaDxtTnNcCy' failed.
```

Explicit DENY

NFSv4 permissions might include explicit DENY attributes for OWNER, GROUP, and EVERYONE. That is because NFSv4 ACLs are `default-deny`, which means that if an ACL is not explicitly granted by an ACE, then it is denied. Explicit DENY attributes override any ACCESS ACEs, explicit or not.

DENY ACEs are set with an attribute tag of `D`.

See the following example:

```
sh-4.1$ nfs4_getfacl /mixed
A::ldapuser@domain.lenovo.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC
```

DENY ACEs should be avoided whenever possible because they can be confusing and complicated. When DENY ACEs are set, users might be denied access when they expect to be granted access. This is because the ordering of NFSv4 ACLs affects how they are evaluated.

The preceding set of ACEs is equivalent to 755 in mode bits. That means the following:

- The owner has full rights.
- Groups have read only.
- Others have read only.

However, even if permissions are adjusted to the 775 equivalent, access can be denied because of the explicit DENY set on EVERYONE.

For an example of explicit DENY, see “NFSv4.x ACL Explicit DENY Example.”

NFSv4 ACL Preservation

By default, NFSv4 ACLs can be affected by setting mode bits on a file or folder. If an NFSv4 ACE has been configured and a `chmod` is used, the ACE is removed. This is controlled through the `v4-acl-preserve` option. For an example, see “NFSv4.x ACL Preservation Example” in this document.

Note: This option is set to enabled by default. Lenovo recommends not modifying this option unless a specific use case arises.

NFSv4 Delegations

NFSv4 introduces the concept of delegations that provide an aggressive local client cache, which is different from the ad hoc caching that NFSv3 provides. There are two forms of delegations: read and write. Delegations value cache correctness over improving performance. For delegations to work, a supported UNIX client is required along with the NFS 4.x version-specific delegation options enabled on the Lenovo controller. These options are disabled by default.

When a server determines to delegate a complete file or part of the file to the client, the client caches it locally and avoids additional RPC calls to the server. This reduces GETATTR calls in the case of read delegations because there are fewer requests to the server to obtain the file’s information. However, delegations do not cache metadata, which means that high file count workloads do not see as much of a benefit with delegations as a streaming file workload might see.

Read delegations can be granted to numerous clients, but write delegations can be granted only to one client at a time, because any new write to a file invalidates the delegation. The server reserves the right to recall the delegation for any valid reason. The server determines to delegate the file under two scenarios: a confirmed call-back path from the client that the server uses to recall a delegation if needed and when the client sends an OPEN function for a file.

Why Use Read or Write Delegations?

Delegations can be used to improve the read and write performance of certain applications. For example, web applications that have numerous readers of one or more files on the same client and across clients that also generate copious amounts of metadata operations like GETATTRs and LOOKUPs could request read delegations from the storage system for local access to improve performance and response time. Delegating the whole file or certain ranges of bytes to the client's local memory avoids additional RPC calls over the wire for metadata operations.

If the file or byte offset is rewritten by any client during the delegation, the delegation is recalled. Although this process is necessary to acquire updates, the delegation recall can affect read performance. Therefore, write delegations should be used for single writer applications. Read and write delegations can improve I/O performance, but that depends on the client hardware and operating system. For instance, low-memory client platforms do not handle delegations very well.

How can I tell if I am using delegations?

If you enable delegations and the client supports them, they are used. You can verify that delegations are being used by checking `vserver locks show -type delegation`.

```
cluster::*> vserver locks show -type delegation

Vserver: DEMO
Volume   Object Path               LIF         Protocol  Lock Type   Client
-------- ------------------------- ----------- --------- ----------- ----------
flexgroup_16
         /flexgroup_16/files/topdir_82/subdir_268/file3
                                   data        nfsv4.1   delegation  -
                Delegation Type: write
         /flexgroup_16/files/topdir_27/subdir_249/file2
                                   data        nfsv4.1   delegation  -
                Delegation Type: write
```

You can also review delegations with performance statistics.

```
cluster::*> statistics show -object nfsv4_1 -counter *del*

Object: nfsv4_1
Instance: DEMO
Start-time: 4/9/2020 15:05:44
End-time: 4/9/2020 15:31:13
Elapsed-time: 1529s
Scope: cluster
Number of Constituents: 2 (complete_aggregation)
    Counter                                                     Value
    -------------------------------- --------------------------------
    delegreturn_avg_latency                                    1366us
    delegreturn_percent                                            4%
    delegreturn_success                                        311465
    delegreturn_total                                          311465
```

NFSv4 Locking

For NFSv4 clients, ONTAP supports the NFSv4 file-locking mechanism, maintaining the state of all file locks under a lease-based model. In accordance with RFC 3530, ONTAP "defines a single lease period for all state held by an NFS client. If the client does not renew its lease within the defined period, all state associated with the client's lease might be released by the server." The client can renew its lease explicitly or implicitly by performing an operation, such as reading a file. Furthermore, ONTAP defines a grace period, which is a period of special processing in which clients attempt to reclaim their locking state during a server recovery.

Locks are issued by ONTAP to the clients on a lease basis. The server checks the lease on each client by default every 30 seconds. In the case of a client reboot, the client can reclaim all the valid locks from the server after it has restarted. If a server reboots, then upon restarting it does not issue any new locks to the clients for a default grace period of 45 seconds (tunable in ONTAP to a maximum of 90 seconds). After that time the locks are issued to the requesting clients. The lease time of 30 seconds can be tuned based on the application requirements. For information on managing NFS locks, see Managing file locks in the product documentation.

Table 7) NFS lease and grace periods.

Term	Definition (See RFC 3530 for More Information)
Lease	The time period in which ONTAP irrevocably grants a lock to a client
Grace period	The time period in which clients attempt to reclaim their locking state from ONTAP during server recovery

Specifying the NFSv4 Locking Lease Period

To specify the NFSv4 locking lease period (the time period in which ONTAP irrevocably grants a lock to a client), you can modify the `-v4-lease-seconds` option. By default, this option is set to 30. The minimum value for this option is 10. The maximum value for this option is the locking grace period, which you can set with the `locking.lease_seconds` option.

NFSv4.x Referrals

An NFS referral directs a client to another LIF in the SVM upon initial NFS mount request. The NFSv4.x client uses this referral to direct its access over the referred path to the target LIF from that point forward. Referrals are issued when there is a LIF in the SVM that resides on the node where the data volume resides. In other words, if a cluster node receives an NFSv4.x request for a nonlocal volume, the cluster node can refer the client to the local path for that volume by means of the LIF. Doing so allows clients faster access to the data using a direct path and avoids extra traffic on the cluster network.

How They Work

When a mount request is sent, the request acts as a normal NFSv4.x mount operation. However, after the DH LOOKUP call is made, the server (Lenovo cluster) responds with the GETFH status of `NFS4ERR_MOVED` to notify the client that the volume being accessed does not live where the LIF being requested lives. The server then sends a LOOKUP call to the client, notifying it of the IP (using the `fs_location4` value) on the node where the data volume lives. This process works regardless of whether a client is mounting using a DNS name or IP. However, the client reports that it is mounted to the IP specified rather than the IP returned to the client from the server.

If a volume moves to another aggregate on another node, the NFSv4.x clients must unmount and remount the file system manually if volume locality is desired. Remounting makes sure that the client is referred to the new location of the volume. If the manual mount/unmount process is not followed, the client can still access the volume in its new location, but I/O requests would then take a remote path. However, remote I/O requests might not be impactful enough to an environment to matter enough to remount clients, which is a disruptive and potentially involved operation. The decision to remount clients should be made on a case-by-case basis.

Note: NFSv4.x referrals were introduced in RHEL as early as 5.1 (2.6.18-53), but Lenovo recommends using no kernel older than 2.6.25 with NFS referrals and no version earlier than 1.0.12 of nfs-utils.

If a volume is junctioned below other volumes, the referral uses the volume being mounted to refer to as the local volume. For example:

- A client wants to mount vol2.
- Vol2's junction is /vol1/vol2.
- Vol1 lives on node1; vol2 lives on node2.
- A mount is made to cluster:/vol1/vol2.
- The referral returns the IP address of a LIF that lives on node2, regardless of what IP address is returned from DNS for the host name "cluster."
- The mount uses the LIF local to vol2 on node2.

In a mixed client environment, if any of the clients do not support referrals, then the `-v4.0-referrals` option should not be enabled. If the option is enabled and a client that does not support referrals gets a referral from the server, that client is unable to access the volume and experiences failures. See RFC 3530 for more details about referrals.

NFSv4.x Stateless Migration – Oracle dNFS

NFSv4 referrals also brought NFSv4 stateless migration support in ONTAP 8.1 and later and only includes support for Oracle dNFS.

Migration is an NFSv4.x feature that allows a file system to move from one server to another without client disruption. Migration enablement requires enabling referrals and the option `-v4-fsid-change` on the NFS server. Migration is a diag-level option. Enabling migration assumes the following about the solution:

- All clients accessing the NFSv4.x server on the SVM are stateless.
- All clients accessing the NFSv4.x server on the SVM support migrations.
- The NFSv4.x clients do not use the following:
  - Locking
  - Share reservations
  - Delegations
  - OPEN for file access
- The NFSv4.x clients do use the following:
  - READ, WRITE, and SETATTR with special stateid of all bits 0
  - OPEN only to create a file and close it right away
- The NFSv4.x clients do not have a state established on the NFS server.

NFS migration support can be useful in the following scenarios in ONTAP:

- Volume moves
- LIF migration/failover

Table 8) Referrals versus migration versus pNFS.

	Referrals	Stateless Migration	pNFS
When does redirect take place?	At mount	Any operation (I/O and metadata)	I/O only (READ, WRITE)

	Referrals	Stateless Migration	pNFS
Traffic that is redirected	All traffic	All traffic	I/O only (READ, WRITE)
Use case	Automounter	Oracle dNFS	Guaranteed data locality for I/O
Drawback	Only on mount	Only stateless operations (no lock state)	Non-I/O traffic is not redirected

Snapshot Copies with NFSv4.x

In NFSv3, the .snapshot directory is visible to clients by default. (For information on hiding the snapshot directory in NFSv3, see “Hiding Snapshot Copies.”) Since NFSv4.x does not use the MOUNT protocol, the .snapshot directory is not visible. However, it is accessible from anywhere in the NFSv4.x mount.

To access Snapshot copies using NFSv4.x, simply navigate to the .snapshot directory manually.

```
# ls -la /nfs3 | grep snapshot
drwxrwxrwx  16 root                  root                   4096 Mar 31 14:05 .snapshot

# ls -la /nfs4 | grep snapshot
#
# ls -la /nfs4/.snapshot
total 64
drwxrwxrwx 16 root root   4096 Mar 31 14:05 .
drwxr-xr-x 14 root root   4096 Apr 11  2018 ..
drwx--x--x  8 root daemon 4096 Jan 19  2017 base
drwxrwxrwx 11 root root   4096 Jul 10  2017 clone_home_clone.0
drwxr-xr-x 14 root root   4096 Apr 11  2018 daily.2020-03-30_0010
drwxr-xr-x 14 root root   4096 Apr 11  2018 daily.2020-03-31_0010
drwxr-xr-x 14 root root   4096 Apr 11  2018 hourly.2020-03-31_0905
drwxr-xr-x 14 root root   4096 Apr 11  2018 hourly.2020-03-31_1005
```

NFSv4.x Behavior with Load Sharing Mirrors

Load sharing (LS) mirrors in ONTAP are read-only copies of volumes that are replicated through SnapMirror (with LS type specified) to multiple nodes in a cluster to provide resiliency for volumes. LS mirrors are not supported for use with data volumes, but instead are used to protect the SVM root volume, which contains the root of the namespace. This is covered in more detail in “Protecting Your Namespace.”

However, NFSv4.x operations do not leverage LS mirror destinations when mounting volumes and instead use the source volume file handle. As a result, LS mirrors do not provide protection against SVM root volume failures.

2.4 NFSv4.1

NFSv4.1 is considered a minor version update of NFSv4. This section covers the specifics of NFSv4.1. The previous section covered NFSv4.0, as well as topics that apply to both NFSv4.0 and NFSv4.1 (denoted by NFSv4.x in this document).

It is possible to enable NFSv4.1 and disable NFSv4.0. This is recommended if you wish to prevent clients from using NFSv4.0 for any reason.

To mount a client using NFSv4.1, there must be client support for NFSv4.1. Check with the client vendor for support for NFSv4.1. Mounting NFSv4.1 is generally done with the `minorversion` mount option, but newer Linux kernels autonegotiate the highest supported NFS version.

Example:

```
# mount -o nfsvers=4,minorversion=1 NFSSERVER:/unix /unix
```

NFSv4.1 Features

NFSv4.1 introduced a number of new features to the NFSv4 protocol standard, as covered in RFC-5661. These differences are covered in Section 1.8 of the RFC.

Some features are listed as REQUIRED, which means that the feature must be implemented in or supported by the NFS server to claim RFC standard. Other features are listed as RECOMMENDED or OPTIONAL features and are supported ad hoc by the NFS server but are not required to claim RFC compliance. For example, pNFS is listed as an OPTIONAL feature for NFSv4.1 and is supported by ONTAP, but NFS session trunking and directory delegations (also OPTIONAL features) are not currently supported by ONTAP.

Parallel Network File System (pNFS)

Parallel NFS (pNFS) is a part of NFS version 4.1 standards. With traditional NFS versions 3, 4, and 4.1, the metadata and data shared the same I/O path. pNFS handles metadata and data on different I/O paths. A metadata server handles all the metadata activities from the client, while the data servers provide a direct path for data access.

As is explained in RFC 5661:

“Parallel data access is controlled by recallable objects known as ‘layouts,’ which are integrated into the protocol locking model. Clients direct requests for data access to a set of data servers specified by the layout using a data storage protocol which might be NFSv4.1 or might be another protocol.”

For pNFS, Lenovo supports all clients that support pNFS and follow the RFC specifications. By default, the pNFS option is enabled, but it is only active if NFSv4.1 support also is enabled.

How pNFS Works

pNFS defines the notion of a device that is generated by the server (that is, an NFS server running on ONTAP) and sent to the client. This process helps the client locate the data and send requests directly over the path local to that data. ONTAP generates one pNFS device per flexible volume. The metadata path does not change, so metadata requests might still be remote. In an ONTAP pNFS implementation, every data LIF is considered an NFS server, so pNFS only works if each node owns at least one data LIF per NFS SVM. Doing otherwise negates the benefits of pNFS, which is data locality regardless of which IP address a client connects to.

The pNFS device contains information about the following:

- Volume constituents
- Network location of the constituents

The device information is cached to the local node for improved performance.

To see pNFS devices in the cluster, use the following command in advanced privilege:

```
cluster::> set diag
cluster::*> vserver nfs pnfs devices cache show
```

pNFS Components

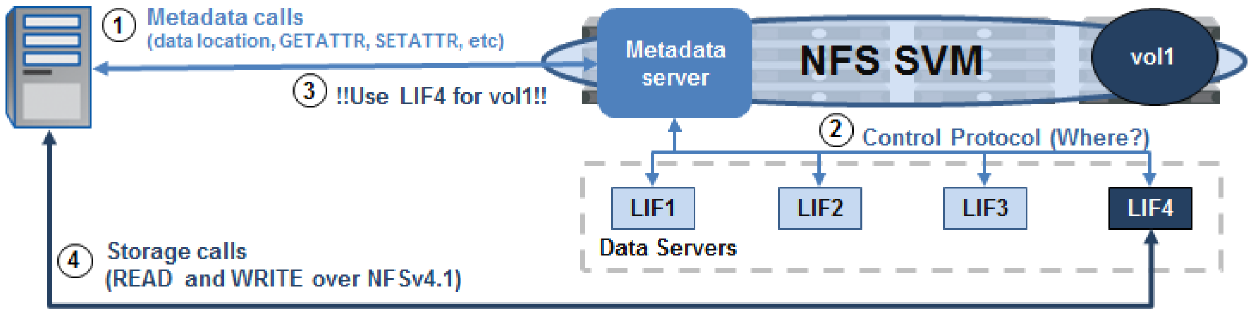
There are three main components of pNFS:

- Metadata server
  - Handles all nondata traffic such as GETATTR, SETATTR, and so on
  - Responsible for maintaining metadata that informs the clients of the file locations
  - Located on the Lenovo NFS server
- Data server

  - Stores file data and responds to READ and WRITE requests
  - Located on the Lenovo NFS server
  - Inode information also resides here
- Clients

These components leverage three different protocols. The control protocol is the way the metadata and data servers stay in sync. The pNFS protocol is used between clients and the metadata server. pNFS supports file, block, and object-based storage protocols, but Lenovo currently only supports file-based pNFS.

Figure 8) pNFS data workflow.



1. The client makes a data request to the cluster.
2. The metadata server works to find the location of the data if the location is not already cached.
3. The location of the data is returned to the client via the control path.
4. The client begins operations over the specified data LIF returned from the metadata server.

How can I tell pNFS is being used?

To check if pNFS is in use, you can run statistics counters to check for `pnfs_layout_conversions` counters. If the number of `pnfs_layout_conversions` are incrementing, then pNFS is in use.

```
cluster::*> statistics show -object nfsv4_1_diag -counter pnfs_layout_conversions

Object: nfsv4_1_diag
Instance: nfs4_1_diag
Start-time: 4/9/2020 16:29:50
End-time: 4/9/2020 16:31:03
Elapsed-time: 73s
Scope: node1

    Counter                                                     Value
    -------------------------------- --------------------------------
    pnfs_layout_conversions                                      4053
```

•

NFSv4.1 Delegations

NFSv4.1 delegations are very similar to NFSv4.0 delegations, but are part of the v4.1 protocol rather than v4.0. The following is a table that covers the new additions to NFSv4.1 and how they benefit an environment over NFSv4.0. These additions are covered in detail in RFC 5661, Section 10.2.

Table 9) NFSv4.1 delegation benefits.

NFSv4.1 Delegation Feature	Benefit Versus NFSv4.0 Delegation
EXCHANGE_ID is used	In NFSv4.0, SETCLIENTID was used. EXCHANGE_ID replaces SETCLIENTID and enables a client ID to be assigned before any other client operations take place. As per RFC 5661, "The only NFSv4.1 operations possible before a client ID is established are those needed to establish the client ID.”
Callbacks use the same TCP connection as the forechannel	In NFSv4.0, callbacks use different TCP connections than the forechannel. Using the same TCP connection for callbacks provides better performance for delegations and is more firewall friendly.
New OPEN request options: • OPEN4_SHARE_ACCESS_WANT_DELEG_MASK • OPEN4_SHARE_ACCESS_WANT_NO_PREFERENCE • OPEN4_SHARE_ACCESS_WANT_READ_DELEG • OPEN4_SHARE_ACCESS_WANT_WRITE_DELEG • OPEN4_SHARE_ACCESS_WANT_ANY_DELEG • OPEN4_SHARE_ACCESS_WANT_NO_DELEG	NFSv4.1 provides more precise control to clients for acquisition of delegations than NFSv4.0. These new options enable more OPEN scenarios to be covered to prevent problems issuing or reclaiming delegations.

NFSv4.1 Sessions

As per RFC 5661:

A session is a dynamically created, long-lived server object created by a client and used over time from one or more transport connections. Its function is to maintain the server's state relative to the connection(s) belonging to a client instance. This state is entirely independent of the connection itself, and indeed the state exists whether or not the connection exists. A client might have one or more sessions associated with it so that client-associated state can be accessed by using any of the sessions associated with that client's client ID, when connections are associated with those sessions. When no connections are associated with any of a client ID's sessions for an extended time, such objects as locks, opens, delegations, layouts, and so on, are subject to expiration. The session serves as an object representing a means of access by a client to the associated client state on the server, independent of the physical means of access to that state.

A single client can create multiple sessions. A single session MUST NOT serve multiple clients.

From SNIA:

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 are never 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.

NFSv4.1 Session Trunking

The ONTAP NFSv4.1 server currently does not offer support for the session trunking (or, multipathing) feature. This feature is commonly asked for in VMware environments.

2.5 NFSv4.2

NFSv4.2 is the latest NFSv4.x version available and is covered in RFC-7862. Currently there is no support for NFSv4.2 in any ONTAP release.

3 Name Services

In enterprise NAS environments, thousands of clients, users and groups are interacting with storage systems every day. These clients, users and groups require easy management that is consistent across all NAS clients. You don't want "user1" in client A to be different than "user1" in client B, and you also don't want client A and client B to use the same hostnames or IP addresses.

That's where name services come in.

3.1 Domain Name System (DNS)

DNS servers provide a centralized way to create and manage IP addresses, hostnames and business-critical service records. When all clients and storage systems point to the same DNS configurations, then there is consistency in hostname <-> IP mapping without needing to manage thousands of local files.

DNS is critical in many applications and network services, such as:

- Kerberos
- LDAP
- Active Directory

Use of DNS in NFS environments is highly recommended – especially when Kerberos and LDAP are involved.

Dynamic DNS

In DNS, it's possible to have clients send DNS updates to the DNS servers when IP addresses are added, deleted or modified. This feature reduces the amount of management overhead is needed for DNS, but is only possible if the DNS server supports it.

ONTAP provides a method for data LIFs to send updates to DNS servers through dynamic DNS. This is managed with the `vserver services name-service dns dynamic-update` commands.

DNS Load Balancing

In some cases, a host name might not just be a single IP address, but might be a front-end for multiple IP addresses. In ONTAP, an SVM might have multiple data LIFs. NFS clients generally access these network interfaces with a DNS hostname that load balances connections across the multiple IP addresses. ONTAP supports a few methods to load balance connections with DNS.

- Off-box DNS (round robin through A records)
- On-box DNS (DNS forwarding/delegation to the ONTAP DNS server)
- 3rd party load balancer (hardware or software gateway)

On-Box DNS or Off-Box DNS?

ONTAP provides a method to service DNS queries through an on-box DNS server. This method factors in a node's CPU and throughput to help determine which available data LIF is the best one to service NAS access requests.

- Off-box DNS is configured by way of the DNS administrator creating multiple A name records with the same name on an external DNS server that provides round-robin access to data LIFs.
- For workloads that create mount-storm scenarios, the ONTAP on-box DNS server cannot keep up and balance properly, so it's preferable to use off-box DNS.

For more information, see TR-4523: DNS Load Balancing in ONTAP.

3.2 Identity Management Name Services

For identity management, LDAP and NIS provide a central repository for users and groups, as well as netgroup functionality. These centralized services offer a way for clients and servers to maintain the same information to provide predictable, consistent identities when accessing NAS file systems.

ONTAP supports both LDAP and NIS for name services, but LDAP is recommended over NIS for its security and replication support.

Lightweight Directory Access Protocol (LDAP)

The recommended method for identity management for users, groups and netgroups is to use an LDAP server. Not only does LDAP centralize the source for name services across NFS clients and servers, it also provides a way to secure communication through secure binds and searches using SSL or Kerberos to encrypt the LDAP packets. NIS servers don't offer support for this by default.

Additionally, LDAP servers offer easier ways to replicate their information across multiple servers – particularly when using Active Directory for UNIX identity management.

Network Information Services (NIS)

NIS databases can also be used for users, groups and netgroup name services with ONTAP. ONTAP uses the standard NIS *.byname functionality for lookups using ypserv calls. Any NIS server that leverages standard NIS functionality can be used for lookups – including Windows Active Directory.

ONTAP can also enable local NIS group.byname and netgroup.byname functionality (similar to NIS slave) for chatty NIS environments. This helps reduce the overall load on the network and NIS servers when enabled.

3.3 Local Files

Local files (such as passwd, group and netgroup) are also supported as name service sources. With ONTAP, a storage administrator can either build the files with ONTAP commands, through the GUI (UNIX user and group creation), or they can import flat files from servers using the load-from-uri commands. By default, ONTAP SVMs support up to 64,000 entries for local UNIX users and groups.

If local files are to be the primary name service and there needs to be more than 64,000 entries, then enabling scaled/file-only mode would be an option.

Scaled Mode/File-Only Mode

Scaled mode/file-only mode for local users and groups in ONTAP 9.1 and later allows storage administrators to expand the limits of local users and groups by enabling a diag-level name service option and then using the load-from-uri functionality to load files into the cluster to provide higher numbers of users and groups. Scaled mode/file-only mode also can add performance improvements to name service lookups, because there is no longer a need to have external dependencies on name service servers, networks, and so on. However, this performance comes at the expense of ease of management of the name services because file management adds overhead to the storage management and introduces more potential for human error. Additionally, local file management must be done per cluster, adding an extra layer of complexity.

To enable this option for users and groups, use the `vserver services name-service unix-user file-only` and `vserver services name-service unix-group file-only` commands:

After the mode is enabled, use the following command to load the user and group file from URI:

```
cluster::*> vserver services name-service unix-user load-from-uri
```

Note: If loading files larger than 10MB for users and 25MB for groups, use the `–skip-file-size-check` option.

When using file-only mode, individual operations on users and groups are not allowed. This configuration is not currently supported in Lenovo MetroCluster™ or SVM disaster recovery (SVM DR) scenarios.

Can You Still Use External Name Services When Using File-Only Mode?

File-only mode does not mean you cannot use LDAP or NIS as a name service; it means that management of local users and groups is done with files only (as opposed to replicated database entries). LDAP and NIS lookups still work properly when file-only mode is enabled.

Default Local Users

When an SVM is created using vserver setup or Storage Manager, default local UNIX users and groups are created, along with default UIDs and GIDs.

The following example shows these users and groups:

```
cluster::> vserver services unix-user show -vserver vs0
               User            User   Group  Full
Vserver        Name            ID     ID     Name
-------------- --------------- ------ ------ --------------------------------
nfs            nobody          65535  65535  -
nfs            pcuser          65534  65534  -
nfs            root            0      0      -

cluster::> vserver services unix-group show -vserver vs0
Vserver        Name                ID
-------------- ------------------- ----------
nfs            daemon              1
nfs            nobody              65535
nfs            pcuser              65534
nfs            root                0
```

Note: When using file-only mode, be sure the preceding users exist in the files being used to manage the cluster. After file-only is enabled, the default users are removed if the uploaded file does not include them.

Local User Effect

When file-only is enabled, the default local users of root, pcuser, and nobody are removed if the file being loaded does not have the users. Be sure to include the local users and groups in your passwd/group files when using file-only.

Limits

This section covers the limits for using local users and groups in ONTAP. These limits are cluster-wide.

Table 10) Limits on local users and groups in ONTAP.

	Local UNIX Users/Groups	Scaled-Mode Users/Groups
Local users and groups maximum entries	65,536	Users: 400K Groups: 15k Group memberships: 3000k SVMs: 6
Scaled-mode users and groups maximum file sizes	N/A	Passwd file size (users): 10MB* Group file size: 25MB* *group and passwd file sizes can be overridden with `–skip-file-size-check` but larger file sizes have not been tested

As previously mentioned, the local UNIX user and group limits are cluster-wide and affect clusters with multiple SVMs. Thus, if a cluster has four SVMs, then the maximum number of users in each SVM must add up to the maximum limit set on the cluster.

For example:

- SVM1 has 2,000 local UNIX users.
- SVM2 has 40,000 local UNIX users.
- SVM3 has 20 local UNIX users.
- SVM4 would then have 23,516 local UNIX users available to be created.

Any attempted creation of any UNIX user or group beyond the limit would result in an error message.

See the following example:

```
cluster::> unix-group create -vserver NAS -name test -id 12345

Error: command failed: Failed to add "test" because the system limit of {limit number}
       "local unix groups and members" has been reached.
```

4 Multiprotocol NAS

ONTAP supports multiprotocol NAS access to the same datasets in an SVM. With ONTAP multiprotocol NAS access, users and groups can use CIFS/SMB and NFS to access volumes or qtrees and leverage ACLs for users and groups, along with setting file and folder ownership as desired. This section contains an overview of the functionality and basic concepts of multiprotocol NAS access in ONTAP.

Basic Multiprotocol Functionality

The following covers the basic process for multiprotocol NAS functionality when users attempt access. These steps are covered in more detail in the sections following.

- User attempts to mount the NAS share (NFS export or CIFS/SMB share).
- ONTAP receives the request and checks the initial share access to see if the client and/or user can open the share.
  - If initial access to open the share is allowed, then the user authenticates into ONTAP based on the protocol + volume security style of the object being accessed.
- Depending on the protocol and security style of the volume, ONTAP attempts to map the incoming user numeric name or ID to a corresponding UNIX or Windows users.

  - The UNIX user mapping leverages name services configured in the ns-switch databases for the SVM.
- The list of credentials (group memberships, name mappings, and so on) is fetched and cached.
- If the user maps to a valid Windows or UNIX user, then the Access Control List is checked to see if the user has file permissions to the object being accessed.
- If the user has permissions, access is granted.

Volume Security Styles

CIFS/SMB and NFS use very different permission models for user and group access. As a result, ONTAP must be configured to honor the desired permission model for protocol access. For NFS-only environments, the decision is simple – use UNIX security styles.

If NFS and CIFS/SMB are required, then the decision should be made based on two main concepts:

- What protocol will users manage permissions from the most?
- What is the desired permission management endpoint? (ie, do users require the ability to manage permissions from NFS clients or Windows clients? Both?)

When we say “volume security styles,” what is really meant is “permission styles.”

ONTAP Volume/Qtree Security Styles

ONTAP offers three volume security styles to choose from for volumes and qtrees:

- **UNIX.** UNIX security style provides UNIX-style permissions, such as basic mode-bits (Owner/Group/Everyone access with standard Read/Write/Execute permissions, such as 0755) and NFSv4.x ACLs. POSIX ACLs are not supported.

NTFS. NTFS security style provides identical functionality as Windows SMB permissions provide, with granular user and groups in Access Control Lists (ACLs) and detailed security/audit permissions.

Mixed. Mixed security style takes the concepts of UNIX and NTFS security styles and applies one as an “effective” style based on the protocol that last modified the ACL. For instance, if a Windows SMB client changes permissions of a file or folder in a mixed security style volume, then that file or folder takes on NTFS as the effective security style and apply the necessary ACLs. If an NFS client changes the permissions on that same file or folder later, then the effective security style changes to UNIX. This provides the ability for multiple clients to manage permissions and works best for applications that require this functionality.

As a best practice, a mixed security style is not recommended unless your application has a direct requirement.

Table 11) Limitations of existing security styles.

Security Style	Limitations
UNIX	• Windows clients cannot set UNIX permission attributes through SMB. • NFSv4.x ACLs don’t have GUI management.
NTFS	• UNIX clients cannot set attributes through NFS. • NFS clients show only approximated permissions when viewing ACLs.
Mixed	• Both Windows and UNIX clients can set attributes. • Only one style of ACL can be honored on an object. – Applying UNIX-style ACLs drops NTFS-style ACLs. – Applying NTFS-style ACLs drops UNIX-style ACLs.

	Last protocol used to modify the ACL determines the file's effective security style.

Name Mapping

When accessing NAS shares, users might require name mappings to properly authenticate into ONTAP and translate permissions accurately. Name mapping is the way ONTAP resolves a user name from one protocol method to another to enable proper multiprotocol NAS access.

Name mappings take place in the following order:

- ONTAP checks for a 1:1 (symmetric) name mapping. For example, UNIX user `lenovo` maps to Windows user `DOMAIN\lenovo`.
- If no 1:1 mapping exists, the namemap ns-switch database is consulted for name service sources for name mapping. By default, local files are used (through `vserver name-mapping rule` entries), but LDAP can also be used for namemap entries.
- If no name mapping rules exist for the user, then ONTAP attempts to use the default user name set on the CIFS/SMB or NFS server. By default, CIFS/SMB uses `pcuser` as a default UNIX user (`-default-unix-user`). NFS servers have no default Windows user (`-default-win-user`) set.
- If no user can be mapped, the NAS request fails.

Name Mapping Functionality Based on Security Style

The direction a name mapping occurs (Windows to UNIX or UNIX to Windows) depends which protocol is being used, but also on which security style is applied to a volume. While a Window client always needs a Windows to UNIX name mapping, whether or not that user is applied to review permissions depends on the security style. Conversely, an NFS client only needs to use a UNIX to Windows name mapping if NTFS security style is in use.

Table 13 summarizes name mapping direction and security styles.

Table 12) Name mappings and security styles.

Protocol	Security Style	Name Mapping Direction	Permissions Applied
CIFS/SMB	UNIX	Windows to UNIX	UNIX (mode-bits or NFSv4.x ACLs)
CIFS/SMB	NTFS	Windows to UNIX	NTFS ACLs (based on Windows SID accessing share)
CIFS/SMB	Mixed	Windows to UNIX	Depends on effective security style
NFSv3	UNIX	None	UNIX (mode-bits or NFSv4.x ACLs*)
NFSv4.x	UNIX	Numeric ID to UNIX user name	UNIX (mode-bits or NFSv4.x ACLs)
NFS	NTFS	UNIX to Windows	NTFS ACLs (based on mapped Windows user SID)
NFS	Mixed	Depends on effective security style	Depends on effective security style

* NFSv4.x ACLs can be applied using an NFSv4.x administrative client and honored by NFSv3 clients.

Permissions

This section describes how ONTAP handles permissions based on the security style in use.

ACL Interaction with Different Security Styles

The security semantics of a volume are determined by its security style and its ACL (NFSv4 or NTFS).

For a volume with UNIX security style:

- NFSv4 ACLs and mode bits are effective.
- NTFS ACLs are not effective.
- Windows clients cannot set attributes.

For a volume with NTFS security style:

- NFSv4 ACLs are not effective.
- NTFS ACLs and mode bits are effective.
- UNIX clients cannot set attributes.

For a volume with mixed security style:

- NFSv4 ACLs and mode bits are effective.
- NTFS ACLs are effective.
- Both Windows and UNIX clients can set attributes.

Displaying NTFS Permissions from NFS Clients

When you use NTFS security style volumes or qtrees, NFS clients display the mode bits or NFSv4 ACLs for the object as having wide open permissions (777) by default. This can be problematic for users and storage administrators for two primary reasons:

- Applications might depend on the ACLs or mode bits displaying properly for functionality.
- Users who see the mode bits as `wide open` might become alarmed, which can result in support tickets and cycles spent on troubleshooting.

Even though an ACL or mode bit shows `777` in NTFS security style volumes, it does not mean that the object allows everyone full access. In ONTAP, NTFS security style volumes control access based on NTFS security and ACLs. Therefore, an NFS client must have a valid UNIX user that maps to a valid Windows user to be able to access the volume at all (authentication). After the initial authentication, the mapped user is then used to determine access based on the granular NTFS ACLs.

ONTAP 8.3.1 and later introduced an option called `ntacl-display-permissive-perms`. The default value for the option is `disabled`, which allows the approximation of interpreted NTFS ACLs on NFS clients mounting NTFS objects, thereby displaying permissions based on minimum access, more closely approximating the real NTFS permissions of the current user in UNIX terms. This helps alleviate concerns and address application compatibility.

The option allows the user accessing the NTFS security–style volume to see the approximate permissions provided based on the user accessing the share. Therefore, users accessing the object might see differing results based on the NTFS security access.

Also, because of the vast difference between NTFS and UNIX-style ACLs, the approximation of permissions might not be exact. For example, if a user has a granular permission provided only in NTFS security semantics, then the NFS client cannot interpret that properly.

Viewing Permissions and Security Styles from the ONTAP CLI

In some cases, storage administrators might want to see the effective security styles of a volume, user ownership or permissions of a file, folder or volume from the ONTAP CLI. To accomplish this, use the following command:

```
cluster::> vserver security file-directory show -vserver vs0 -path /junction-path
```

5 Nondisruptive Operations with NFS

This section covers nondisruptive operations (NDO) with NFS in ONTAP and scenarios with NDO behavior for NFS clients. In some cases, even NFSv3 can be disrupted by specific planned and unplanned events. The reason for this happening is that, even though NFSv3 is a stateless protocol, there are still underlying mechanisms such as locking and NFS server-side caches that can come into play during disruptive events.

5.1 Replay/Reply Cache

The replay (or reply) cache in ONTAP is crucial to preventing NFS requests from trying nonidempotent requests twice. Nonidempotent requests are requests that can change data structures. For example, reading a document twice at the same time is an idempotent operation because it is harmless. Editing that document twice at the same time is a nonidempotent operation and can be harmful if the document does not have locking in place to protect it. The replay/reply cache in ONTAP helps keep track of what operations have arrived on the storage in case a network issue causes a client to resend the same operation. The cache is used to reply to the operation rather than retrying in the storage layer.

This cache is stored at the data layer with the volumes. When this cache is lost, CREATE operations can fail with E_EXIST and REMOVE operations can fail with E_NOENT. Table 14 shows different scenarios in which replay cache is kept or lost, which determines the disruptive-ness of the operation.

Table 13) Replay/reply cache NDO behavior.

Operation	Result (NFSv3 and NFSv4.x
Volume move	Replay cache is moved with volume.
Aggregate relocation or storage giveback operation	Replay cache is lost.
LIF migrate (same node)	Replay cache remains intact.
LIF migrate (different node)	Replay cache is lost.
Unplanned takeover	Replay cache is lost.
Planned takeover	Replay cache is lost.

5.2 File Locking

File locking mechanisms were created to prevent a file from being accessed for write operations by more than one user or application at a time. NFS leverages file locking either using the NLM process in NFSv3 or by leasing and locking, which is built in to the NFSv4.x protocols. Not all applications leverage file locking, however; for example, the application vi does not lock files. Instead, it uses a file swap method to save changes to a file.

When an NFS client requests a lock, the client interacts with the ONTAP system to save the lock state. Where the lock state is stored depends on the NFS version being used. In NFSv3, the lock state is stored at the data layer. In NFSv4.x, the lock states are stored in the NAS protocol stack.

In NFSv3 environments, locks are managed by the NLM protocol, which is ancillary to the NFS protocol. As a result, when locks are used in NFSv3, there might be stale locks leftover after failovers that need to be cleaned up manually. NFSv4.x locks are reclaimed based on a lease model and do not need manual cleanup. For more information on NFS file locking, see “File Locking Concepts.”

To view or remove file locks in an SVM, use the following commands in advanced privilege:

```
cluster::> set advanced
cluster::*> vserver locks
    break show
```

When potentially disruptive operations occur, lock states do not transfer in some instances. As a result, delays in NFS operations can occur as the locks are reclaimed by the clients and reestablished with their new locations. Table 15 covers the scenarios in which locks are kept or lost.

Table 14) Lock state NDO behavior.

Operation	NFSv3	NFSv4.x
Volume move	Lock state is moved with volume.	Lock state is moved with volume.
Aggregate relocation or storage giveback operation	Lock state is not moved; up to 45s outage when locks are in use.	Lock state is not moved; up to 90s outage when locks are in use.
LIF migrate (same node)	Lock state is not stored in NAS protocol stack; no disruption.	Lock state remains intact; still on local node; no disruption.
LIF migrate (different node)	Lock state is not stored in NAS protocol stack; nothing to move; no disruption.	Lock state is not moved; up to 90s outage when locks are in use.
Unplanned takeover	Lock state is not moved; up to 45s outage when locks are in use.	Lock state is not moved; up to 90s outage when locks are in use.
Planned takeover	Lock state is not moved; up to 45s outage when locks are in use.	Lock state is not moved; up to 90s outage when locks are in use.

5.3 NFSv4.1 Sessions

In ONTAP, NFSv4.1 sessions are supported. With NFSv4.1 sessions, LIF migrations can be disruptive to NFSv4.1 operations, but they are less disruptive than with NFSv4.0. For more information, see RFC-5661, section 2.10.13.

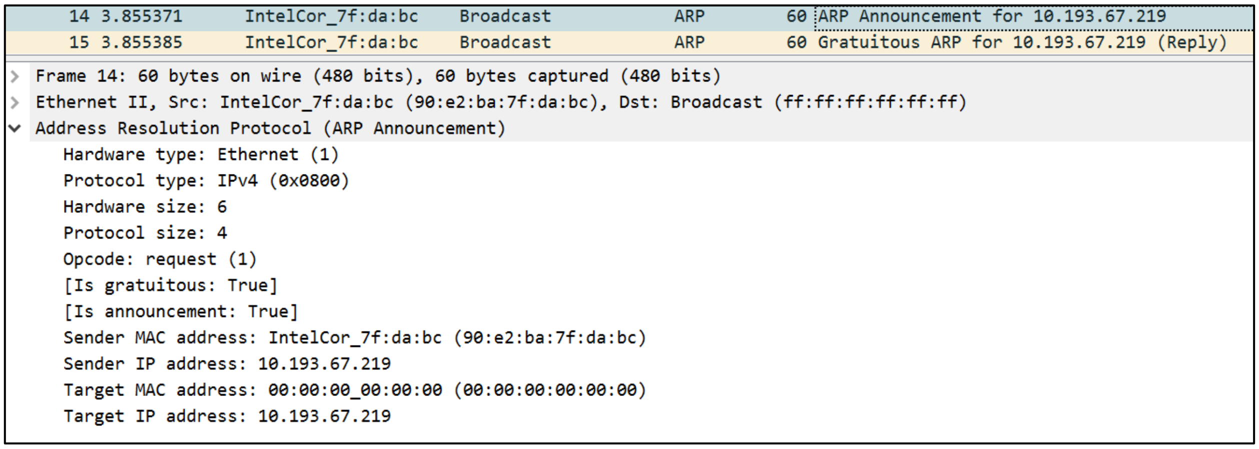
5.4 What Happens During LIF Migrations in NFSv4.x?

When a data LIF hosting NFSv4.x traffic is migrated in ONTAP, existing NFSv4.x traffic must be quiesced until a safe point in the process to move the LIF. After the NFS server is determined to be safe to allow the migration, the LIF is then moved to the new location and lock states are reclaimed by NFS clients. Lock state reclamation is controlled by the NFS option `-v4-grace-seconds` (45 seconds by default). With NFSv4.1 sessions, this grace period is not needed, because the lock states are stored in the NFSv4.1 session. Busier systems cause longer latency in LIF migrations because the system must wait longer for the operations to quiesce and the LIF waits longer to migrate. However, disruptions occur only during the lock reclamation process.

5.5 LIF Migrations with NFSv3

When a LIF migrates due to storage failover or port failure, ONTAP broadcasts an ARP announcement over the network for the IP address to inform clients that the MAC address has changed for the data LIF. As a result, the clients update their ARP tables to reflect that change. If a client cannot update the ARP entry (such as if a client firewall like AppArmor or SELinux is blocking ARP broadcasts), the NFS access still attempts to use the old MAC address, which causes access failures/mount hangs until the LIF fails back to the original MAC address or the ARP cache is updated.

Figure 9) Gratuitous ARP during LIF migration



```
14 3.855371    IntelCor_7f:da:bc    Broadcast    ARP    60 ARP Announcement for 10.193.67.219
15 3.855385    IntelCor_7f:da:bc    Broadcast    ARP    60 Gratuitous ARP for 10.193.67.219 (Reply)

> Frame 14: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: IntelCor_7f:da:bc (90:e2:ba:7f:da:bc), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
v Address Resolution Protocol (ARP Announcement)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    [Is gratuitous: True]
    [Is announcement: True]
    Sender MAC address: IntelCor_7f:da:bc (90:e2:ba:7f:da:bc)
    Sender IP address: 10.193.67.219
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.193.67.219
```

You can view the ARP caches on clients and compare the MAC entries with the port information on the Lenovo cluster.

For example, this is what the client sees:

```
# arp -a x.x.x.a
demo.ntap.local (x.x.x.a) at 90:e2:ba:7f:d4:bc [ether] on eno16780032
```

This is what ONTAP sees:

```
cluster::*> net int show -vserver DEMO -address x.x.x.a -fields curr-port,curr-node
vserver lif   curr-node          curr-port
------- ----- ------------------ ---------
DEMO    data2 node2              e2a

cluster::*> net port show -node node2 -port e2a -fields mac
node               port mac
------------------ ---- -----------------
Node2              e2a  90:e2:ba:7f:d4:bc
```

When the LIF is migrated, the client's ARP cache gets updated with the new port's MAC address:

```
cluster::*> net int migrate -lif data2 -destination-node node1 -destination-port e2a

cluster::*> net port show -node node1 -port e2a -fields mac
node               port mac
------------------ ---- -----------------
node1              e2a  90:e2:ba:7f:da:bc

# arp -a x.x.x.a
demo.ntap.local (x.x.x.a) at 90:e2:ba:7f:da:bc [ether] on eno16780032
```

5.6 Direct Connect NFS

Direct connect NFS refers to using a direct connection from an NFS client to the storage system. Because NFS is an ethernet-based protocol, there is no technical reason this souldn't work. However, in a Lenovo cluster, there are multiple nodes used for NFS connectivity. In the event of a storage failover, cable failure, port failure, and so on, then the directly connected client has no way to communicate with the NFS export.

Lenovo highly recommends using NFS over a full-fledged network for the best reliability and performance.

6 Volume Styles: FlexGroup or FlexVol?

When deploying volumes to use with NFS workloads, you have two choices of volume styles available.

- **FlexVol volumes.** The standard volume type available in ONTAP and span a single node's hardware.
- **FlexGroup volumes.** Volumes that are made of up multiple FlexVol member volumes spanning multiple hardware domains in a cluster that provide a number of advantages over FlexVol volumes including:
  - Volume sizes greater than 100TB (20PB tested).
  - File counts greater than 2 billion (400 billion tested).
  - Multi-threaded metadata operations that provide 2-6x performance for high-ingest workloads.
  - Ability to use multiple nodes in a cluster to automatically balance workloads.
  - FlexVol-like management for ease of use.
  - Nondisruptive expansion when a volume reaches capacity.

In most NFS workloads, a FlexGroup volume provides more benefits relative to FlexVol volumes. The main caveat when deciding is to check feature parity between the volume styles to see whether or not the necessary features in your environment are supported.

7 NFS Auditing

This section covers the setup and use of NFS auditing, which can use either NFSv4.x audit ACEs (UNIX security styles) or Windows audit ACEs (NTFS security styles).

7.1 NFS Audit Setup

The main requirement for setting up NFS auditing is to have an audit ACE on the volume that requires auditing. The audit ACE can be Windows or NFSv4.x, but in NFS-only environments, the ACE must be NFSv4.x. As a result, NFSv4.x must be enabled in the NFS server, and an NFSv4.x admin client needs to be used to set up the auditing.

Enabling Auditing on System

For more information about NFSv4.x ACLs, see the "NFSv4.x Access Control Lists (ACLs)."

After NFSv4.x and NFSv4.x ACLs are enabled, enable NFS auditing with the following command:

```
cluster::> vserver audit create -vserver nfs -destination /unix -rotate-size 100MB
```

This command enables auditing for NFS and CIFS access on the junction path `/unix` for the SVM named `nfs`. After auditing is enabled on the ONTAP system, the AUDIT ACEs should be created.

Note: If you are using inheritable audit ACEs, be sure to create at least one inheritable allow or deny ACE on the parent directory to avoid access issues.

Creating NFSv4 AUDIT ACEs

To create an NFSv4 AUDIT ACE, mount the volume on which auditing was enabled using NFSv4.x. After the volume is mounted, create an AUDIT ACE on the volume, files, and/or directories where auditing is required.

An AUDIT ACE can be used to track ALLOW or DENY for a variety of operations, including the following:

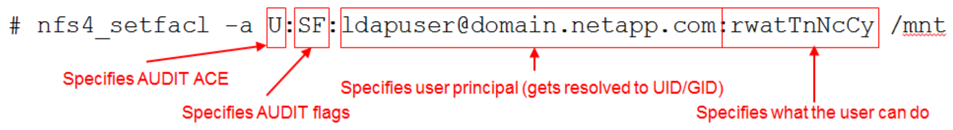
- Read
- Write
- Execute
- Append
- Delete

For information about all of the ACE permissions in NFSv4, see the nfs4_acl(5) - Linux man page.

Each Linux client uses a different method of assigning NFSv4.x ACEs. In RHEL/CentOS/Fedora, the commands `nfs4_setacl` and `nfs4_getacl` are used.

An AUDIT ACE leverages flags to specify if auditing should be for successes, failures, or both. AUDIT ACEs use the ACE type of U.

Figure 10) Example of setting NFSv4 audit ACE.



```
# nfs4_setfacl -a U:SF:ldapuser@domain.netapp.com:rwatTnNcCy /mnt
```

Specifies AUDIT ACE
Specifies AUDIT flags
Specifies user principal (gets resolved to UID/GID)
Specifies what the user can do

After the AUDIT ACE is applied and the user that is being audited attempts access, the events get logged to an XML file on the volume.

For an example of a logged NFS audit event, see "NFS Audit Event Example."

8 NFS Best Practices

This section covers best practices with NFS environments. Best practices are simply recommendations that should be considered when using and deploying NFS. Best practices are not hard requirements and in many cases are dependent on a variety of factors in the environment.

8.1 General ONTAP Best Practices

The following list covers general ONTAP best practices:

- Upgrade to the latest ONTAP version for bug fixes and newest NFS features.
- Use ThinkSystem Intelligent Monitoring for upgrade recommendations and proactive remediation.
- Enable AutoSupport to allow for a seamless and efficient support experience.
- Configure LS mirrors for the vsroot volume in NFSv3 environments.
- For the best performance, use Lenovo All-Flash Hybrid systems.
- Enable all storage efficiencies for maximum storage utilization.
- Follow best practices for aggregate and storage configurations as per the product documentation.
- In most cases, leave defaults unchanged unless there is a specific reason to change the settings.
- Install ThinkSystem Intelligent Monitoring Unified Manager and configure to monitor your ONTAP cluster.
- Set up proactive alerts for events such as storage failovers, volume capacity alerts, etc.
- Consider deploying FlexGroup volume for your NFS workloads – particularly in high file count environments.

8.2 ONTAP Data LIF Best Practices with NFS Environments

ONTAP allows storage administrators to provide the following benefits:

- Seamless scale-out storage

- Multiprotocol unified access (NFS, CIFS, and SAN)
- Nondisruptive operations

This is done by way of a secure multitenant architecture with SVMs.

What are SVMs?

SVMs are logical storage containers that own storage resources such as flexible volumes, logical interfaces (LIFs), exports, CIFS shares, and so on. Think of them as a storage blade center in your cluster. These SVMs share physical hardware resources in the cluster with one another, such as network ports/VLANs, aggregates with physical disk, CPU, RAM, switches, and so on. Data access can occur on any data network interface owned by SVMs, regardless of the location in the cluster. As a result, load for SVMs can be balanced across a cluster for maximum performance and efficiency or to leverage SaaS functionality, among other benefits. Alternately, a single SVM can be used to present a monolithic storage device to an environment.

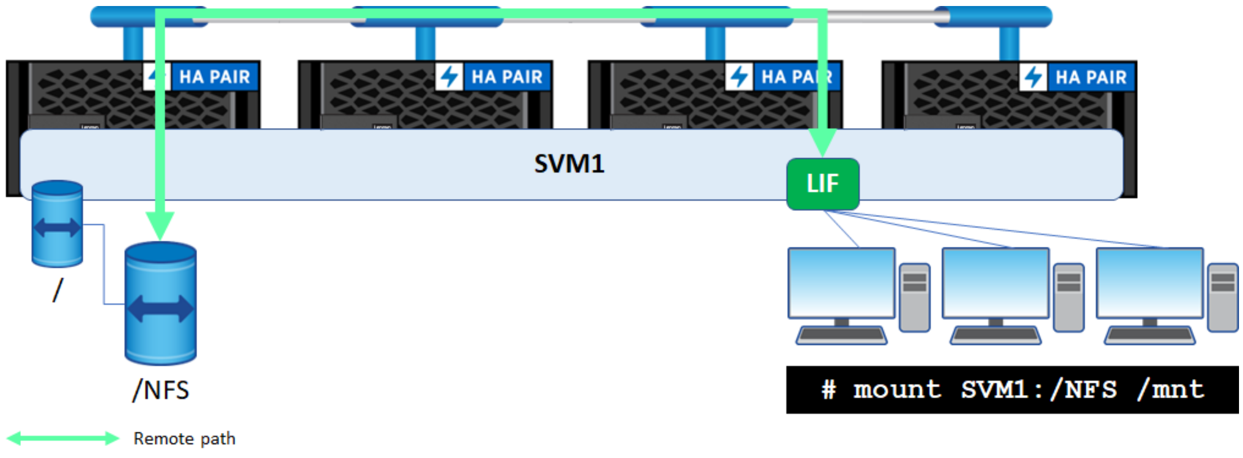
Data Logical Interface Considerations

Data logical interface (LIFs) can live on any physical port in a cluster that is added to a valid broadcast domain. These data LIFs are configured with SVM-aware routing mechanisms that allow the correct pathing of Ethernet traffic in an SVM, regardless of where a valid data LIF lives in the cluster. When designing a network for NAS interaction, one of two approaches can be taken.

Option #1: Simplicity Approach - Single LIF per SVM

Essentially, all it takes to access NAS data in ONTAP is a single network IP address that is routable to network clients. In many environments, a single network interface suffices for NAS workloads. If the underlying physical network port fails, or if a storage node is taken over by its HA partner, then the network IP address migrates to another working port in the cluster. Using a single network interface reduces the number of IP addresses needed, but it also limits the amount of potential network bandwidth that would be available to a workload. Sending all NAS traffic to a single node in the cluster also limits the number of resources (such as CPU and RAM) available, so if a workload is expected to require high throughput or connects hundreds to thousands of clients, then option #2 might be a better choice.

Figure 11) Single LIF NAS interaction.



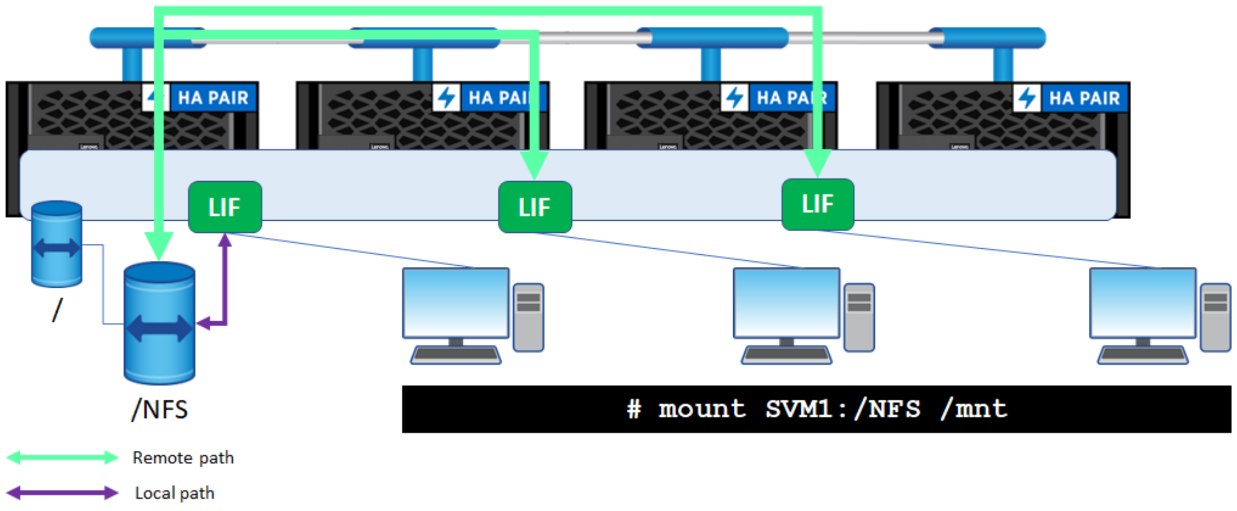
```
# mount SVM1:/NFS /mnt
```

Option #2: Performance Approach - Multiple Data LIFs per SVM

In an ONTAP cluster, multiple nodes can be made available for NAS connectivity and storage. Remember, ONTAP clusters using NAS only can scale up to 24 nodes. Multiple nodes mean multiple physical resources, such as CPU/RAM/network interfaces. As a result, having more than one data LIF in an SVM can add considerable performance to your NAS workloads. Spreading network connections across nodes alleviates CPU and network port contention, as well as avoiding scenarios where a node can have too many TCP connections. For network load balancing of NAS connections, round robin DNS, on-box DNS, or standard load balancing hardware can be leveraged.

In situations where the best possible performance is required, or where many clients are accessing a NAS device at the same time, creating multiple data LIFs per SVM is a sound approach. Additionally, use of load balancing NAS features such as NFS referrals, CIFS auto-location, and pNFS require a data LIF on each node where data resides.

Figure 12) Multiple LIF NAS interaction.



```
# mount SVM1:/NFS /mnt
```

Data LIF Locality Recommendations

In ONTAP, you can use data locality features such as NFS referrals, CIFS auto-location, and pNFS for NAS traffic regardless of where the volumes live in a cluster. For NFS referrals and CIFS auto-location, the initial TCP connection is automatically redirected to a network interface that is local to the requested volume. If the volume being used is a FlexGroup volume, then NFS referrals and CIFS auto-location should not be used.

pNFS provides a metadata path on the initial mount request, but all reads and writes are automatically redirected to local volumes through pNFS layout calls. pNFS is only available for the NFSv4.1 protocol and only with NFS clients that support it. FlexGroup volumes can only use pNFS in ONTAP 9.7 and later. For more information on pNFS, see “Parallel Network File System (pNFS).”

Without auto-location features, managing data LIF locality to avoid the cluster network adds management complexity that might not be worth the effort, because the performance effect for most NFS workloads is negligible. Ideally, NAS connections connect to data LIFs that are local to the volumes, but, with FlexGroup volumes/scale-out NAS and larger cluster backend networks, this becomes less important.

Data Locality Benefits and Considerations

This section describes benefits and considerations of data locality in ONTAP and how to approach these concepts with simplicity in mind.

The ability to spread the load across nodes and leverage all the available hardware in a cluster
When creating volumes and network interfaces, consider deploying workloads across multiple nodes in the cluster to maximize the performance headroom. Why pay for hardware you do not use?

- **Simplicity approach:** ONTAP provides automated provisioning of storage when using ThinkSystem Storage Manager for DM Series. This takes into account available performance headroom and attempts to place new volumes on nodes that are less utilized. Additionally. FlexGroup volumes provision across multiple nodes in a cluster and automatically balance workloads to a single namespace.

The ability to balance network connections across multiple cluster nodes

Clusters are single entities, as are SVMs. But they do have underlying hardware that has its own maximums, such as the number of connections and so on.

- **Simplicity approach:** Create multiple data LIFs per SVM and mask those interfaces behind a DNS round robin name or DNS forwarding zone leveraging ONTAP's on-box DNS feature. In addition, leverage FlexGroup volumes to spread workloads across multiple nodes.

The ability to enable data locality in the event of a volume move

If you move a volume to another node, you can be certain you still have a local path to the data if every node has a data LIF for the SVM. When moving volumes in ONTAP to new nodes, existing TCP connections remain in place for NAS clients. As a result, these NAS operations traverse the cluster network.

- **Simplicity approach:** Do nothing. In most cases, NAS clients notice no difference in performance to these NAS shares. If using NFSv4.1, consider using pNFS.

Network Port Exhaustion with a Large Number of NFS Clients

In environments with a large number of clients connecting through NFS, it is important to keep in mind that, by default, the number of mount and NFS ports are limited to 1,024.

This number is controlled with the following options:

```
mount-rootonly
nfs-rootonly
```

By default, `mount-rootonly` is set to `enabled` and `nfs-rootonly` is set to `disabled`.

In some circumstances, the number of ports used to mount or for NFS operations might be exhausted, which then causes subsequent mount and NFS operations to hang until a port is made available.

If an environment has thousands of clients that are mounted through NFS and generating I/O, it is possible to exhaust all ports on an NFS server. For example, one scenario seen was with ESX using NFS datastores, because some legacy best practices would call for a data LIF/IP address per datastore. In environments with many volumes or datastores, this created a situation where NFS ports were overrun. The remediation for this situation would be to disable the `mount-rootonly` and/or the `nfs-rootonly` options on the NFS server. This removes the 1 to 1,024 port range limit and allows up to 65,534 ports to be used in a NFS server. For more information on these options, see "The rootonly Options—nfsrootonly and mountrootonly."

This situation affects the source port (client-side) only. The server-side mountd, portmapper, NFS, and nlm ports are designated by ONTAP.

NFS Behind Network Address Translation

NFS maintains a reply cache to keep track of certain operations to make sure that they have been completed. This cache is based on the source port and source IP address. When Network Address Translation (NAT) is used in NFS operations, the source IP or port might change in flight, which could lead to data resiliency issues. If NAT is used, static entries for the NFS server IP and port should be added to make sure that data remains consistent.

In addition, NAT could also lead to issues with NFS mounts hanging due to how NAT handles idle sessions. If using NAT, the configuration should take idle sessions into account and leave them open indefinitely to prevent issues. NAT can also create issues with NLM lock reclamation.

Ultimately, the best practice for NAT with NFS would be to avoid using it if possible and instead create a data LIF on the SVM. If NAT is necessary, work with the NAT vendor to configure it properly for NFS operations.

8.3 NFS Security Best Practices

While NFS stands for Network File System, it has playfully been referred to as Not for Security. NFSv3 and prior were limited in security scope, because client IP addresses, the numeric user, group IDs, and other identifying information could be passed over networks in clear text.

Given that the default security mechanisms for NFSv3 security are export rules and basic user mode-bit permissions, spoofing IP addresses and user identities for access is fairly simple and straightforward. As a result, the following suggestions can be used to better secure your NFS environments.

Choosing Your NFS Version

When deploying an NFS environment, the NFS version being used is one of the many things to consider. NFSv3 is by far the most popular NFS version, but NFSv4.x is picking up steam as more and more businesses are requiring more stringent security requirements.

- If security is the most important factor in your environment, you should at the very least test NFSv4.x, because it is designed for security.
- If raw performance is the priority, then NFSv3 would be the prudent choice, as because statelessness provides distinct performance advantages over NFSv4.x.
- If locking is important, then NFSv4.x offers the most robust and resilient options.
- If a blend of functionality is required, then it is best to test which NFS version makes the most sense for your workloads.

The section called “NFS Version Considerations” breaks down the details of all NFS versions, advantages, features, and migration considerations.

Securing NFSv3

While NFSv4.x is superior to NFSv3 for security, there are still some steps we can take to better secure the protocol.

Export Policy Rule Considerations

Export policies and rules are the first gateway for securing an NFS export. These rules allow you to specify one or more clients and define the level of access they receive when they mount an export. These rules override file-level permissions. For example, if you set rw to `never`, then, even if the UNIX permissions are 777, that client cannot write to the export. For more information on NFS export policies and rules, see “Export Concepts.”

The way you set up your export policies and rules also affect if a client is allowed to mount the export at all. The following considerations should be considered when securing NFSv3 exports:

- If an export policy has no rules and is applied to a volume or qtree, then no one is allowed access to mount.
- If an export policy rule doesn’t include a specific client (either through IP/hostname, netgroup, or subnet), that client is unable to mount.

- If the export policy rule on the vsroot volume does not allow read access to a specific client, then that client is unable to mount the export. Read access is require on an export policy rule to allow clients to traverse the paths to the volume. Vsroot is "/" in the export path.
  - Vsroot uses the default export policy when it is created. By default, the default policy has no export rules.

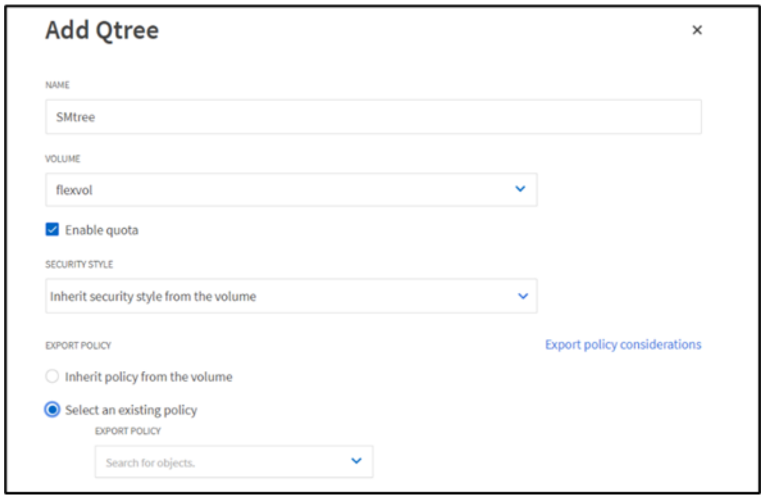
When setting the clientmatch for the vsroot's export policy rule to allow path traversal, consider setting it by netgroup, subnet or a list of clients. Avoid setting the entry to 0.0.0.0/0 or 0/0; this allows all clients to traverse/read the export. For more information, see "Qtree Exports."

In ONTAP, it is possible to set export policies and rules for volumes, as well as underlying qtrees. This offers a way to restrict/allow client access to storage-manage In ONTAP, it is possible to set export policies and rules for volumes, as well as underlying qtrees. This offers a way to restrict/allow client access to storage-managed directories in ONTAP, which can help storage administrators more easily manage workloads such as home directories.

By default, qtrees inherit the export policy of the parent volume. You can explicitly choose or create an export policy and rule when creating qtrees in ThinkSystem Storage Manager for DM Series, or by using the -export-policy CLI option.d directories in ONTAP, which can help storage administrators more easily manage workloads such as home directories.

By default, qtrees inherit the export policy of the parent volume. You can explicitly choose or create an export policy and rule when creating qtrees in ThinkSystem Storage Manager for DM Series, or by using the -export-policy CLI option.

Figure 6) Qtree export specification – ThinkSystem Storage Manager for DM Series.



Add Qtree

NAME

SMtree

VOLUME

flexvol

[✓] Enable quota

SECURITY STYLE

Inherit security style from the volume

EXPORT POLICY

Export policy considerations

○ Inherit policy from the volume

● Select an existing policy

EXPORT POLICY

Search for objects.

Qtree IDs and Rename Behavior

After a noninherited export policy is applied to a qtree, NFS file handles change slightly when dealing with operations between qtrees. ONTAP validates qtree IDs in NFS operations, which affects things like file renames or moves when moving to or from a qtree in the same volume as the source folder or qtree. This is considered a security feature, which helps prevent unwanted access across qtrees, such as in-home directory scenarios. However, simply applying export policy rules and permissions can achieve similar goals.

For example, a move or rename to or from a qtree in the same volume results in "Access denied." The same move or rename to or from a qtree in a different volume results in the file being copied. With larger files, the "copy" behavior can make it seem like a move operation is taking an unusually long time, where most move operations are near-instantaneous, because they are simple file renames when in the same file system/volume.

This behavior is controlled by the advanced privilege option and is covered in detail in the Lenovo knowledge base article Permission denied while moving files between qtrees when NFS option '-validate-qtree-export' is enabled.

From that KB, these are the behaviors of different operations:

Note: Assuming that file permissions allow and that client is allowed by export policies to access both source and destination volume/qtree, these are the current permutations with the 'validate-qtree-export' flag enabled or disabled:

- Enabled:
  - Rename in same volume and qtree: SUCCESS
  - Rename in same volume, different qtrees: EACCESS
  - Rename between volumes where qtree IDs differ: EACCESS
  - Rename between volumes where qtree IDs match: XDEV
- Disabled:
  - Rename in same volume and qtree: SUCCESS
  - Rename in same volume, different qtrees: SUCCESS
  - Rename between volumes where qtree IDs differ: XDEV
  - Rename between volumes where qtree IDs match: XDEV

  Note: NFS3ERR_XDEV and NFS3ERR_ACCESS are defined in RFC-1813.

To change the behavior of renames/moves across qtrees, modify `-validate-qtree-export` to disabled.

Note: There are no known negative effects caused by disabling the -validate-qtree-export option outside of allowing renames across qtrees.

File Handle Effect for qtree Exports

Normally, the NFS export file handles that are handed out to clients are 32 bits or less in size. However, with qtree exports, extra bits are added to create 40-bit file handles. In most clients, this is not an issue. However, older clients (such as HPUX 10.20, introduced in 1996) might have problems mounting these exports. Be sure to test older client connectivity in a separate test SVM before enabling qtree exports, because there is currently no way to change the file handle behavior after qtree exports have been enabled.

- Access Control to vsroot.
- To allow (or prevent) root access for a client, use the superuser export policy rule option. For more information, see “Mapping All UIDs to a Single UID (squash_all).”
- In multiprotocol NAS environments (CIFS/SMB and NFS access), then the way ONTAP responds to permission or owner changes on NFS mounts of volumes with NTFS security depends on the setting of the export policy. The options are `fail` (send an error) or `ignore` (fail silently with no error).
- Setting `-rw-rule`, `-ro-rule` or `-superuser` to `none` squashes a client’s access to the user set with `-anon`. In most cases, `-anon` should be set to `65534`. Exceptions include setting `-anon` to `0` to allow root access to an admin host or setting it to an explicit user to control file ownership for a specific client/set of clients.
- Rather than setting hostnames and IP addresses in the export clientmatch, consider using netgroups to mask host/client lists in the `-clientmatch` field. Netgroups can live in local files on an SVM, in LDAP or NIS.
- For clients that are doing read/write access, set `-superuser` to `none` and `-anon` to 65534 to prevent root access.

- Consider setting up a group of clients that are designated for administrative/root-level tasks and set the export policy to allow root access. See “” for details on how to allow root access.
- If you are using Kerberos for NFSv3 mounts, make sure that you have included both `sys` and `krb5*`” in your rules. NFSv3 uses ancillary protocols and only uses Kerberos for the NFS portion. Limiting access to export policy rules in NFSv3 to only `krb5*` results in failed mounts, as sys access is needed for portmapper, mount, and so on.
- Avoid using `any` for `-rw-rule`, `-ro-rule`, `-superuser`; allowing `any` reduces the security effectiveness of the policy rule. Instead, specify the authentication method you require in those options.
- Avoid using `any` for `-protocol`, as this allows access to protocols you might not want to use. If you require both NFSv3 an NFSv4.x access to mounts, set the protocol to `nfs`. If you only want NFSv3 or only NFSv4.x to access a mount, specify either `nfs3` or `nfs4`.

Client Firewalls

By default, NFS clients enable firewalls (such as SELinux in RHEL or AppArmor in Ubuntu). In some cases, those firewalls can affect NFS operations. Always check the client firewall rules if NFS access issues occur, and, if necessary during troubleshooting, stop the firewall service.

Firewall Port Rules

For NFSv3, there are a number of ports that need to be allowed by firewalls to provide full access to all NFSv3 operations. In general, firewalls have built-in rules, such as `NFS` that keeps a list of well-known NFS ports. However, many security environments recommend changing well-known ports for better security. The ports listed in “Default NFS Ports in ONTAP” show which ports ONTAP uses by default for NFSv3, which ports can be modified and which NFS version they apply to.

The following contains a set of general recommendations for firewall rules as guidance. These are not requirements by Lenovo; they are simply some ideas to help secure NFS environments.

- Consider changing the default port numbers for NFS ancillary protocols to ports that are not well-known and adjusting firewall ports as needed.
- Consider allowing only the NFSv3 ports that are needed by your environment. For example, if rquota is not being used, do not allow it though a firewall. There are several ports that are required for NFS mounts, however, so you should always allow those port numbers.
  - Portmapper (111), mountd (635), and NFS (2049) are required to allow NFS access. Other ancillary ports (NLM, NSM, or rquota) are only needed if an application or client needs them. Test in your environment accordingly.
- NFS mounts generate a source port on the client. By default, ONTAP sets the range of allowed source ports between 1-1024 (`-mount-rootonly`). In some cases, this range of port numbers might not be large enough to accommodate the number of clients that are mounting. If more source port numbers are needed, set `-mount-rootonly` to `disabled` and modify the firewall rules to accommodate that change. See “The rootonly Options—nfsrootonly and mountrootonly” for more information.
- NFS operations also use a range of source ports that can be controlled through `-nfs-rootonly`. By default, this value is set to `disabled`, which means the port range is 1-65536. If a low number of NFS clients is used (100 or less), consider setting this option to `enabled` to reduce the port range to 1-1024 for better security.
- Avoid opening up NFS ports outside of your local network. If NFS over a WAN is required, strongly consider using NFS Kerberos with krb5p for end-to-end encryption. Alternately, consider using Lenovo FlexCache volumes to localize NFS traffic to a remote site. Lenovo FlexCache volumes use TLS 1.2 to encrypt communications.

- Export policy rules and name services for UNIX identity management might depend on external name servers (such as DNS, LDAP, NIS, Active Directory, and so on); make sure that the firewall rules allow traffic for those name servers.
- Some firewalls might drop idle TCP connections after a set amount of time. For example, if a client has an NFS mount connected, but doesn't use it for a while, it's considered `idle`. When this happens, client access to mounts might hang, as the network connection has been severed by the firewall. Keepalives can help avoid this, but it is better to address this either by configuring firewalls to actively reject packets from stale sessions or configuring the ONTAP NFS server options `-idle-connection-timeout` and `-allow-idle-connection`. These options were introduced in ONTAP 9.5.

Permissions

While it is possible to set read and write permissions for clients in export policies, you cannot set permissions for specific users and groups with just export rules. In addition, you cannot get more granular than read or write with export policies. As a result, file and folder permissions are needed. NFSv3 by default uses a fairly basic permission model for files and folders through mode-bits, which means that permissions can only be explicitly controlled for the file/folder owner and everyone else. Additionally, the level of permissions is limited to what is defined in RFC-1813.

Table 16 shows what each permission mode bit does from 0–7.

Table 15) UNIX mode bit levels.

Symbolic Notation	Mode Bit Value	Access Level
-	0	No access
rwx	7	All access
rw-	6	Read & write
r-x	5	Read & execute
r--	4	Read
-wx	3	Write & execute
-w-	2	Write
---x	1	Execute

It is also possible to leverage an NFSv4.x admin client to mount a filesystem and add NFSv4.x ACLs, which are then honored by NFSv3 clients. This provides more granular permissions for NFSv3 environments.

The following additional recommendations can help secure NFSv3 a bit better. These are general recommendations and do not apply to all environments, so be sure to test them in your environment.

- For the vsroot volume, limit the ower and group to `read and execute` and everyone else to `execute` only privileges (551). Vsroot is generally small, so you want to prevent users from writing to it when possible. Limiting everyone other than the owner and group to `execute` permissions only allows traversal of the path and does not allow file and directory listings in the vsroot volume.
- Avoid setting permissions to 7 on volumes/qtrees, unless you want to allow root access to the volume owner or group.
- Follow standard UNIX permission best practices when managing access to NFS filesystems.
- Consider using NFSv4.x ACLs or NTFS security style volume permissions (if multiprotocol NAS is in use) for more robust permission management.

- If you are using LDAP or NIS for UNIX identities, make sure that the user and group lookups are returning the proper users and group memberships from the NFS client and server.

User Authentication

When an NFS mount is accessed, the authentication method is negotiated between the client and server. For standard AUTH_SYS access, the user credentials are passed over the wire in plaintext, which means anyone sniffing the network could see those credentials and use them to gain access to the filesystem. With standard AUTH_SYS, there is no username and password gateway to safeguard against unintended access.

For the best security result with authentication in NFSv3, consider enabling NFS Kerberos for your NFS mounts. AUTH_GSS (or Kerberos) configuration provides a few levels of protection for access, including:

- Username and password interaction with a Kerberos Key Distribution Center server (such as Windows Active Directory, FreeIPA, and so on)
- Security settings protecting against changing the time on a client to re-use an expired Kerberos credential.
- Encryption of the NFS operations, ranging from logins (krb5), data integrity checks (krb5i) and end-to-end NFS encryption (krb5p).
- An increase of the maximum number of user group membership from 16 in AUTH_SYS to 32 in AUTH_GSS. ONTAP provides a way to increase this number to 1024 in the “Auxiliary GIDs – Addressing the 16 GID Limitation for NFS” section.

Encryption

ONTAP provides in-flight encryption for NFS through krb5p, but it also offers a few options for encrypting data at rest. These options include the following:

- SED and LSE drives
- Lenovo Volume Encryption (LVE)
- Lenovo Aggregate Encryption (LAE)

Showmount

Showmount on NFS clients has historically been how users can see exported file systems on an NFS server. By default, NFS servers in ONTAP enables `showmount` functionality to show exported paths but does not list the allowed client access. Instead, `showmount` displays that (everyone) has access.

The / root path does not get shown in `showmount` commands by default. To control the behavior of `showmount` and show the root path, use the following NFS server options:

```
-showmount
-showmount-rootonly
```

This functionality might cause security scanners to flag the NFS server has having a vulnerability, because these scanners often use `showmount` to see what is being returned. In those scenarios, you might want to disable `showmount` on the NFS server.

However, some applications make use of `showmount` for functionality, such as Oracle OVM. In those scenarios, inform the security team of the application requirements.

Securing NFSv4.x

NFSv4.x offers more robust security than NFSv3 and should be used in any NFS environment that prioritizes security over everything else.

NFSv4.x security features include the following:

- Single port for all NFS operations
- ID domain string matching
- NFSv4.x ACLs
- Integrated Kerberos support for username and password authentication and end-to-end encryption
- Client ID checks to help prevent client spoofing

Export Policy Rule Considerations

The export policy rule considerations for NFSv4.x generally mirror those covered in “Securing NFSv3.” Review those considerations for guidance.

Client Firewalls

NFS clients, by default, enable firewalls (such as SELinux in RHEL or AppArmor in Ubuntu). In some cases, those firewalls might affect NFS operations. Always check the client firewall rules if NFS access issues occur, and, if necessary during troubleshooting, stop the firewall service.

Firewall Port Rules

Unlike NFSv3, NFSv4.x combines all NFS operations into compound calls that all leverage the same NFS port – 2049. This port is not modifiable in ONTAP, so if you plan on using NFSv4.x, firewall port rules should be modified to allow port 2049.

Permissions

NFSv4.x provides permissions through mode-bits, as well as NFSv4.x ACLs. The advantages of ACLs are covered in “Benefits of Enabling NFSv4 ACLs,” but for the most robust and granular security with NFSv4.x, use ACLs.

User Authentication

For user authentication in NFSv4.x, the same guidance discussed in “Securing NFSv3” should be followed – use Kerberos whenever the highest level of security is required.

Encryption

For encryption in-flight and at-rest in NFSv4.x, the same guidance discussed in “Securing NFSv3” should be followed.

8.4 Name Services Best Practices

ONTAP provides support for external name service servers for DNS, LDAP/NIS, Kerberos and Active Directory. In NAS environments, name services can provide a wide array of benefits for performance, resiliency and manageability.

- **Centralize name services whenever possible.** In environments with hundreds or thousands of users, groups, host names, and so on, trying to keep local copies of files in sync across the organization can result in inconsistencies, duplicate hostnames or user credentials, and general confusion across platforms. Pointing your clients and ONTAP cluster to the same set of name services vastly reduces the management overhead in these environments and simplifies access permissions and troubleshooting.
- **Always configure DNS.** In the same vein as centralization of name services, having a centralized service to keep track of hostnames, IP addresses, DNS zones and domains and service records rather than trying to maintain these manually is essential. DNS in particular is critical to the

functionality of many NAS services – particularly Windows Active Directory, Kerberos KDCs, LDAP (for service record lookups and load balancing) and NFSv4.x.

- **In NFSv4.x environments, use LDAP for UNIX identity management.** Because NFSv4.x leverages name strings for user authentication, and because user name matching across server and clients controls how NFSv4.x functions, having a centralized name service for UNIX identities for NFSv4.x clients and ONTAP to point to makes using NFSv4.x that much simpler.
- **When using name service servers, have more than one.** Name service servers might receive thousands, hundreds of thousands, or even millions of requests every day. Those requests have a performance cost, both on the network and on the name service servers. ONTAP does a good job of caching requests, but name service servers can still get overwhelmed unless you have set up multiple servers and added them to the client and ONTAP configuration. Even better, load balancing several servers behind the same DNS name allows for easier configuration, reducing the need to re-configure clients and servers if name service addresses change, and masking the number of servers that are being used to load balance requests.

8.5 NFS Client Best Practices

NFS client best practices are generally dependent on the applications being used. When determining how to configure NFS clients, always involve the application vendor in those discussions. The following best practice recommendations are not set in stone and can be overridden by application recommendations or by workload testing.

Client and NFS utility versions

For NFS client versions, the best practice is usually to run the latest OS patch version available, as well as updating the NFS utilities to their latest releases to get the newest bug fixes and feature functionality. In environments where OS and NFS utility versions are qualified and standardized, this might not be an option.

As a general rule, ONTAP NFS supports and any all NFS clients that comply with RFC standards. This does not include NFS clients that have custom kernel compilations.

RPC Slot Tables

RPC Slot Table refers to the maximum allowed threads on a single TCP connection that is allowed by the NFS client and server. These values are controlled through sunrpc configuration on NFS clients. The latest NFS client versions default to a dynamic slot table value of 65536, which means that the client attempts to use as many slot tables as it can in a single TCP connection. ONTAP, however, supports only 128 slot tables per TCP connection. If a client exceeds that value, ONTAP enacts NAS flow control and pauses client operations until resources are freed up.

As a best practice, set the slot table values on NFS clients to a static value no higher than 128. In environments with many clients, this value might need to be set as low as 16. For more information, including details on the performance impact, see “Network Connection Concurrency: NFSv3.”

Mount Options

NFS mount option recommendations depend solely on the workload and application being used. There is general advice for specific mount options, but the decision on which NFS options to use is dependent on the client OS administrators and application vendor recommendations. There is no catch all recommendation for mount options. The following sections covers only a subset of NFS mount options. For a complete list of supported NFS mount options, use `man nfs` on your NFS client.

Default Mount Options

Linux clients set default mount options out of the box. These defaults depend on client OS version and NFS configuration files found on the clients. Default mount options are what get set during a mount operation where no options are specified with the -o flag.

In some cases, mount options are negotiated with the NFS server. Specifically, in newer Linux kernels, the rsize/wsize values and the NFS versions are based on what the NFS server is set to.

For example, if NFSv4.1 is enabled and no NFS version is specified in configuration files or with the mount command, then the client use NFSv4.1 because it is the highest supported NFS version enabled.

The following shows output from a mount command that issued no specific options. The ONTAP NFS server was set to 1MB TCP max transfer size (`-tcp-max-xfer-size`) and has NFSv4.1 enabled.

```
# mount DEMO:/flexgroup_16 /flexgroup
# mount | grep flexgroup
DEMO:/flexgroup_16 on /flexgroup type nfs4
(rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,port=0,timeo=600,retr
ans=2,sec=sys,clientaddr=10.x.x.x,local_lock=none,addr=10.x.x.y)
```

Wsize/rsize

The mount options `wsize` and `rsize` determine how much data is sent between the NFS client and server for each packet sent. This can help optimize performance for specific applications but should be set as per application vendor best practices, as what is best for one application might not be best for other applications.

Newer NFS clients autonegotiate the `wsize` and `rsize` values to what the `-tcp-max-xfer-size` value is set to on the ONTAP NFS server if the mount command does not explicitly set the values. ONTAP defaults `-tcp-max-xfer-size` to 64K and can be set to a maximum of 1MB.

Note: The general recommendation for `-tcp-max-xfer-size` is to increase that value in ONTAP to 262144 (256K) and then specify explicit mount options as the applications require it.

For examples of some performance testing with different workload types and different wsize/rsize values, see “Performance Examples for Different TCP Max Transfer Window Sizes.”

Actimeo and Nocto

NFSv3 manages shared file system consistency and application performance by utilizing cached file/directory data and cached file/directory attributes. It is a loose consistency, because the application doesn’t have to go to shared storage and fetch data every time to use it. That can have tremendous effect to application performance. The cached information has timers that set the period to trust the cache data and at timeout, a light weight, fast getattr/access call to revalidate the data till the next time out.

There are two mechanisms that manage this process:

- Close to open consistency assures getting the latest data for a file, regardless of cache. (cto)
- Attribute cache timers. (actimeo; default 3s for file, 30s for directory)

If the client has complete ownership of data, ie it is not shared, there is guaranteed consistency. You can reduce getattr/access operations to storage and speed up the application by turning off cto consistency (nocto as a mount option) and by turning up the timeouts for the attribute cache management (actimeo=600 as a mount option changes the timer to 10m versus the defaults mentioned above). In some testing, nocto reduces ~65-70% of the getattr/access calls and actimeo reduces another ~20-25%.

There are other cases that can benefit from a similar set of mount options, even though there is not complete ownership by the clients. For applications that use grids of clients like EDA, web hosting and movie rendering and have relatively static data sets (like the tools/libraries in EDA, like the web content for the web hosting, like the textures for rendering), the typical behavior is the data set is largely cached on the clients (very few reads; no writes). There are many getattr/access calls coming back to storage in those cases. These data sets are typically updated through either another client mounting the file systems and periodically pushing content updates or in some case SnapMirror relationships to multiple file systems for update.

In these cases, there is a known lag in picking up new content and the application still works with potentially out of date data. For those scenarios, nocto and actimeo can be used to control the time period where out of data date can be managed. For example, in EDA with tools and libraries and other static content, actimeo=600 works well because this data is typically updated infrequently. For small web hosting where clients need to see their data updates timelier as they are editing their sites, actimeo=10 might be acceptable. For large scale web sites where there is content pushed to multiple file systems, actimeo=60 might be more effective. As always, test with your individual environmets.

Using this mount options reduce the workload to storage significantly in these cases (for example, a recent EDA experience reduced IOPs to the tool volume from >150K to ~6K) and applications can run significantly faster because they can trust the data in memory, rather than needing to query the NFS storage. This also helps reduce overall CPU % and load on the ONTAP nodes.

Actimeo

The actimeo mount option controls the attribute cache timeout on NFS clients. The actimeo option covers the entire range of available attribute caches, including:

```
acregmin=n
The minimum time (in seconds) that the NFS client caches attributes of a regular file before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 3-second minimum.

acregmax=n
The maximum time (in seconds) that the NFS client caches attributes of a regular file before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 60-second maximum.

acdirmin=n
The minimum time (in seconds) that the NFS client caches attributes of a directory before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 30-second minimum.

acdirmax=n
The maximum time (in seconds) that the NFS client caches attributes of a directory before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 60-second maximum.
```

Attribute caching provides some relief on networks by reducing the number of metadata calls. This also helps reduce latency to some workloads, as these metadata operations can now occur locally on the client. Attribute caching generally has no effect to the number of overall operations, unless all operations to the storage were metadata – specifically ACCESS calls.

For example, in our Customer Proof of Concept labs, actimeo was set to 10 minutes (600 seconds) and saw latency cut in half with an EDA workload generated by vdbench (from ~2.08ms to ~1.05ms).

Figure 13) Default actimeo latency - vdbench.

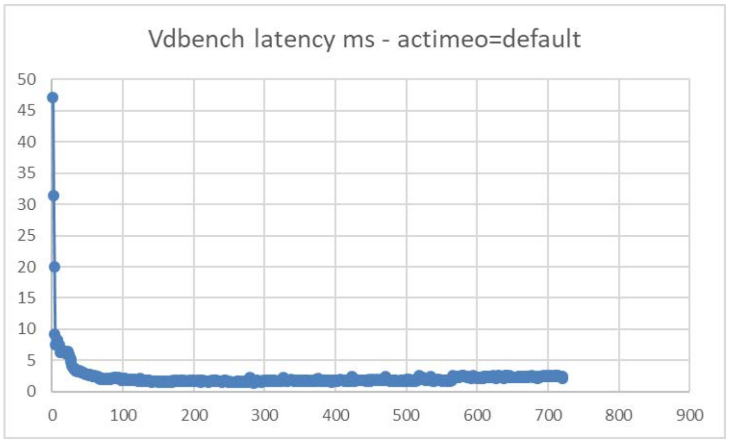
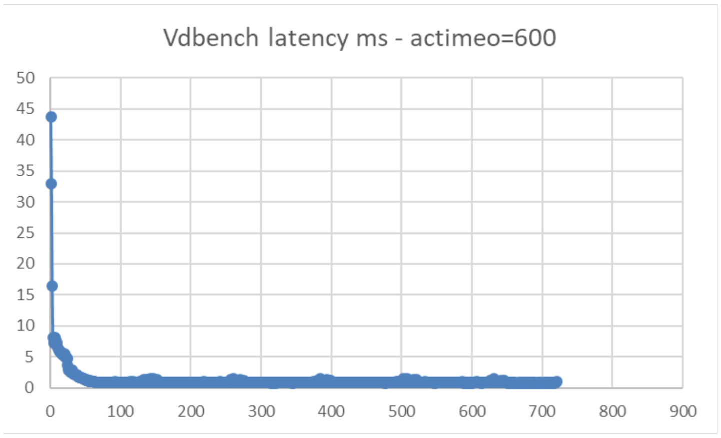


Figure 14) Actimeo=600 latency - vdbench.



The downside of setting the actimeo value too high is that attributes that change might not be reflected properly until the cache timeout occurs, which could result in unpredictable access issues.

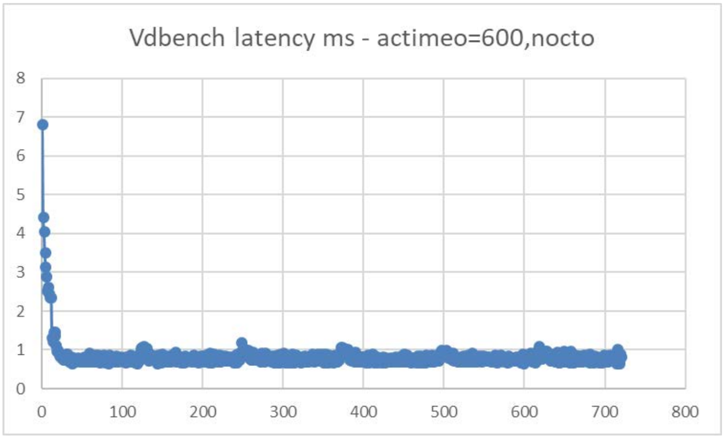
Note: The recommendation for attribute caching is to leave the defaults, unless otherwise directed by the application vendor or if testing shows a vast improvement in performance.

Nocto

Nocto stands for "no close-to-open," which means that a file can close before a write has completed to save time. What this means in NFS environments is that other clients that have that file open for reading will not get consistent updates to the file. By default, nocto is not set on NFS mounts, which means that all files wait to finish writes before allowing a close.

The nocto option is used primarily to increase raw performance. For example, in the same vdbench tests run in our Customer Proof of Concept Labs, the nocto mount option reduced latency by an additional .35ms to .7ms.

Figure 15) Actimeo=600,nocto latency - vdbench.



Note: The recommendation for use of the *nocto* option is to use only with read-heavy/read-mostly workloads.

Clientaddr

By default, the clientaddr mount option gets set automatically by using the NFS client IP address. However, in some cases, the option might need to be specified in NFS mounts.

Two scenarios where it might be necessary to specify clientaddr would be:

- **Multi-NIC clients** might need to specify this option to make sure NFS uses the desired IP address for connectivity.
- **NFSv4.x clients** might need to specify this option if two clients with the same hostname (but different IP addresses) try to access NFS exports in the same SVM. NFSv4.x sends a client ID to the NFS server based on the hostname. ONTAP responds with `CLID_IN_USE` and prevent the second client from mounting if it uses the same client ID. Specifying `clientaddr` can force the client to increment the client ID on subsequent mount attempts.

  Note: In most cases, `clientaddr` does not need to be specified.

Nconnect

A new NFS option called nconnect is in its nascent stages for use with NFS mounts. Currently, nconnect is only available on newer SLES, Ubuntu, and Fedora clients, but is targeted for other Linux kernels in future OS versions. The purpose of nconnect is to provide multiple transport connections per TCP connection/mount point on a client. This helps increase parallelism and performance for NFS mounts.

Note: The recommendation for using nconnect depends on client OS and application needs. Testing with this new option is highly recommended before deploying in production.

Hard/soft

`hard` or `soft` mount options specify whether the program using a file using NFS should stop and wait (`hard`) for the server to come back online if the NFS server is unavailable or if it should report an error (`soft`).

If `hard` is specified, processes directed to an NFS mount that is unavailable cannot be terminated unless the `intr` option is also specified.

If `soft` is specified, the `timeo=<value>` option can be specified, where `<value>` is the number of seconds before an error is reported.

Note: For business-critical NFS exports, Lenovo recommends using `hard` mounts.

Intr/nointr

`intr` allows NFS processes to be interrupted when a mount is specified as a `hard` mount. This policy is deprecated in new clients such as RHEL 6.4 and is hardcoded to "nointr." Kill -9 is the only way to interrupt a process in newer kernels.

Note: For business-critical NFS exports, Lenovo recommends using `intr` with `hard` mounts with NFS clients that support it.

9 Logging, Monitoring and Statistics

This section covers ways to view logs and set up monitoring for NFS-specific environments, as well as managing NFS operations.

9.1 Managing NFS Locks

ONTAP NFS servers keep track of locks when they are established by a NAS client. If desired, storage administrators can view and break locks using the following command from advanced privilege:

```
cluster::*> vserver locks
break show
```

Additionally, to specify NFSv4.x locks, use the following command from diag privilege:

```
cluster::*> vserver locks nfsv4 show
```

9.2 NFS Events – Event Messaging System (EMS)

ONTAP provides a messaging system that keeps track of events within the cluster. These offer summaries of informational, warning and error conditions across all subsystems.

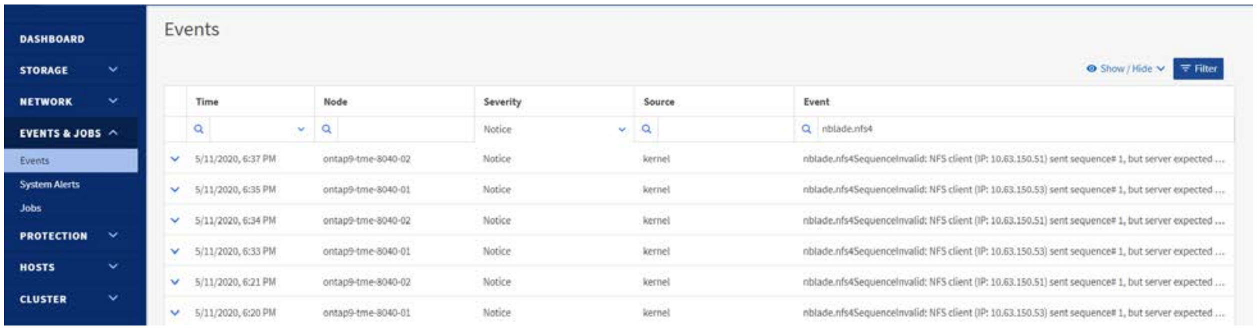
EMS logs can be viewed using ThinkSystem Storage Manager for DM Series, as well as from the CLI with the following command:

```
cluster::> event log show
```

NFS events cover a wide array of subsystems, from general NFS events (such as exports or other errors) to name service and authentication issues. Event names can be filtered by portions of the event message name in the CLI and in the GUI.

```
cluster::*> event log show -message-name nblade.nfs*
Time                Node             Severity      Event
------------------- ---------------- ------------- ---------------------------
5/11/2020 18:37:50  node2
                                     NOTICE        nblade.nfs4SequenceInvalid: NFS client (IP:
x.x.x.x) sent sequence# 1, but server expected sequence# 2. Server error: OLD_STATEID.
5/11/2020 18:35:52  node1
                                     NOTICE        nblade.nfs4SequenceInvalid: NFS client (IP:
x.x.x.y) sent sequence# 1, but server expected sequence# 2. Server error: OLD_STATEID.
5/11/2020 18:34:23  node2
                                     NOTICE        nblade.nfs4SequenceInvalid: NFS client (IP:
x.x.x.x) sent sequence# 1, but server expected sequence# 2. Server error: OLD_STATEID.
```

Figure 16) Filtering events in GUI of ThinkSystem Storage Manager for DM Series.



Time	Node	Severity	Source	Event
		Notice		nblade.nfs4
5/11/2020, 6:37 PM	ontap9-tme-8040-02	Notice	kernel	nblade.nfs4SequenceInvalid: NFS client (IP: 10.63.150.51) sent sequence# 1, but server expected ...
5/11/2020, 6:35 PM	ontap9-tme-8040-01	Notice	kernel	nblade.nfs4SequenceInvalid: NFS client (IP: 10.63.150.53) sent sequence# 1, but server expected ...
5/11/2020, 6:34 PM	ontap9-tme-8040-02	Notice	kernel	nblade.nfs4SequenceInvalid: NFS client (IP: 10.63.150.51) sent sequence# 1, but server expected ...
5/11/2020, 6:33 PM	ontap9-tme-8040-01	Notice	kernel	nblade.nfs4SequenceInvalid: NFS client (IP: 10.63.150.53) sent sequence# 1, but server expected ...
5/11/2020, 6:21 PM	ontap9-tme-8040-02	Notice	kernel	nblade.nfs4SequenceInvalid: NFS client (IP: 10.63.150.51) sent sequence# 1, but server expected ...
5/11/2020, 6:20 PM	ontap9-tme-8040-01	Notice	kernel	nblade.nfs4SequenceInvalid: NFS client (IP: 10.63.150.53) sent sequence# 1, but server expected ...

EMS events are triggered with a severity level, which lets you know which messages are important (such as ERROR and EMERGENCY) and which messages are simply informative (such as INFORMATIONAL and NOTICE). In general, DEBUG level messages can be ignored unless there are other noticeable issues occurring in the environment. If you see messages that are ERROR, ALERT or EMERGENCY, it might be worth opening a support case.

You can filter based on severity, message name and other variables. The following event log commands can show NFS/Multiprotocol NAS-relevant EMS events.

```
cluster::> event log show -message-name dns*
cluster::> event log show -message-name *export*
cluster::> event log show -message-name ldap*
cluster::> event log show -message-name mgmt.nfs*
cluster::> event log show -message-name nameserv*
cluster::> event log show -message-name nblade*
cluster::> event log show -message-name netgroup*
cluster::> event log show -message-name *nfs*
cluster::> event log show -message-name secd*
```

9.3 NFS Statistics

Statistics for NFS are being collected within the ONTAP Counter Manager archives and can be viewed using ThinkSystem Storage Manager for DM Series for up to a year. Additionally, general NFS statistic captures can be done from ActiveIQ Performance Manager.

If individual performance counters for NFS are desired, the `statistics start` command can be used to enable captures for a specific counter object or for multiple objects. After the statistics counters are started, they run until you stop them. These are most useful to run during periods where performance might be suffering, or if you wish to see the workload trends for a specific volume.

9.4 Determining What Type of Virtual Machine Is Hosted on NFS

It is possible to determine what type of virtual machine is accessing the storage using statistics captured in the counter manager. Use the following commands to show the statistics. These are available in diagnostic privilege.

```
cluster::> set diag
cluster::*> statistics start -object wafl -counter wafl_nfs_application_mask
cluster::*> statistics show -object wafl -counter wafl_nfs_application_mask -raw
```

The output of these statistics shows masks for specific virtual machine types.

Table 16) Virtual machine statistic masks.

VM Type	Mask
None	0

VM Type	Mask
ESX/ESXi	1
Citrix Xen	2
Red Hat KVM	4

If more than one virtual machine application is being used, then the masks are added together to determine which ones are in use. For example, if ESX/ESXi and Red Hat KVM are in use, then the masks would be 1 + 4 = 5.

9.5 Determining If Oracle Is in Use

Additionally, the wafl counters can determine if Oracle data is hosted on NFS.

```
cluster::> set diag
cluster::*> statistics start -object wafl -counter wafl_nfs_oracle_wcount
cluster::*> statistics show -object wafl -counter wafl_nfs_oracle_wcount -raw
```

9.6 Performance Monitoring Enhancements

ONTAP have introduced some performance monitoring enhancements that can aid storage administrators in performance monitoring and proactive performance management.

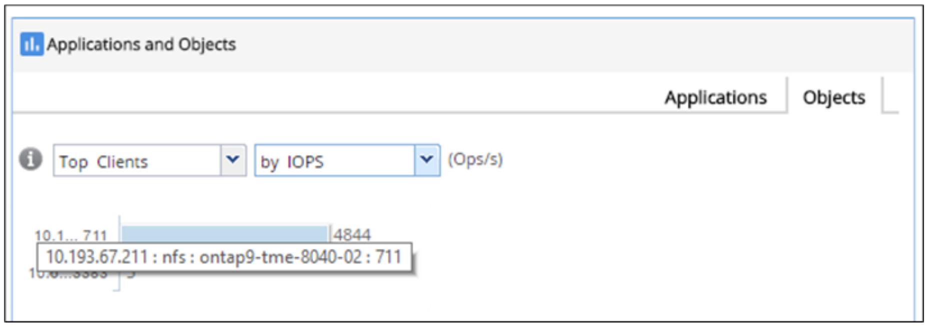
Note: For a complete list of performance counters, objects, and instances, use the `statistics catalog` command, found at advanced privilege.

Top Clients

ONTAP provides the ability to track the top NAS clients writing data to a cluster using the new ONTAP 9 feature called top clients. This feature tracks the overall incoming operations and lists the client IP address, NAS protocol being used, total IOPS, node being accessed, and the SVM to which the client is connecting. This information can be accessed from the CLI or the ThinkSystem Storage Manager for DM Series GUI.

In ThinkSystem Storage Manager for DM Series, the splash screen you see when you first log in shows the top clients at the bottom of the page. Simply select the drop box and select Top Clients and then the Objects tab for a point-in-time look.

Figure 17) Top NAS clients view in ThinkSystem Storage Manager for DM Series.



10.193.67.211 : nfs : ontap9-tme-8040-02 : 711

You can also watch the top clients in real time using the command line with the `admin privilege` command statistics top client show. This command allows you to specify a minimum of 30-second intervals, number of iterations to show, as well as the maximum number of clients that should be displayed. In the following example, a Python file create script was run from two clients to a Lenovo FlexGroup volume over NFSv3 to show an example of what to expect from the command's output.

```
cluster::> statistics top client show -interval 30 -iterations 5 -max 10

cluster : 6/26/2017 17:42:27
*Estimated
     Total
      IOPS Protocol               Node Vserver Client
---------- -------- ----------------- ------- -------------
     23010      nfs              node01 DEMO   x.x.x.b
     20006      nfs              node02 DEMO   x.x.x.c
        17     cifs              node02 CIFS
                                               x.x.x.d
```

By default, the CLI orders by IOPS. It’s also possible to order by throughput by using the `-sort-key` option.

For example:

```
cluster::> statistics top client show -interval 30 -iterations 5 -max 10 -sort-key write_data

cluster : 6/26/2017 18:04:53
*Estimated
     Write
Data (Bps) Protocol               Node Vserver Client
---------- -------- ----------------- ------- -------------
    154968      nfs              node01 DEMO   x.x.x.b
    150400      nfs              node02 DEMO   x.x.x.c
```

Hot Files

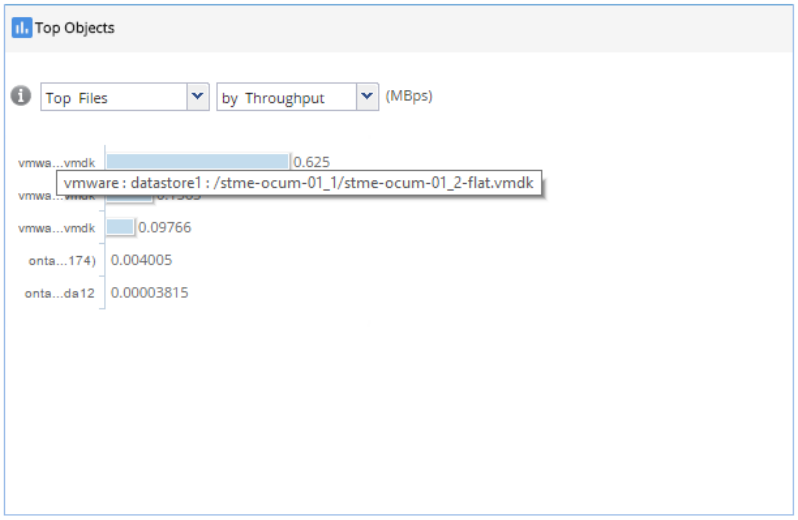
In addition to exposing the top clients accessing a cluster, ONTAP can also show the top files in use on a cluster. This can be particularly useful when dealing with ESX/virtualized environments, where a single ESX server might be hosting hundreds of virtual machines on a single NFS mounted datastore. In that scenario, the “top clients” feature would not be as useful as knowing which files are doing the most work. ONTAP can expose that information all the way down to the VMDK level.

In the following example, this ONTAP cluster is hosting a number of ESXi 6.0 VMs on an NFS mounted datastore. Again, we can see this information in ThinkSystem Storage Manager for DM Series, as well as in the CLI.

In ThinkSystem Storage Manager for DM Series, the top files can be accessed in the bottom graph, by selecting the appropriate drop-down option and the Objects tab.

In Figure 18, the ActiveIQ Unified Manager VM (stme-ocum-01) is using up the most throughput.

Figure 18) Top files view for an ESXi environment in ThinkSystem Storage Manager for DM Series.



From the CLI, the command is `statistics top file show` at admin privilege. In the following example, we still see ActiveIQ Unified Manager VMDK as the top consumer of write throughput and IOPS:

```
cluster::> statistics top file show -interval 30 -iterations 1

cluster : 6/26/2017 18:01:21
*Estimated
     Total
      IOPS              Node Vserver     Volume File
---------- ----------------- ------- ---------- ----------------------------------------
        48            node03  vmware datastore1 /stme-ocum-01_1/stme-ocum-01_2-flat.vmdk
        31            node03  vmware datastore1 /OCUM722-MP/OCUM722-MP_2-flat.vmdk

cluster::> statistics top file show -interval 30 -iterations 1 -max 10 -sort-key write_data

cluster : 6/26/2017 18:05:04
*Estimated
     Write
Data (Bps)              Node Vserver     Volume File
---------- ----------------- ------- ---------- --------------------------------------------
    685600            node03  vmware datastore1 /Parisi OCUM 7.2/Parisi OCUM 7.2_2-flat.vmdk
    159475            node03  vmware datastore1 /stme-ocum-01_1/stme-ocum-01_2-flat.vmdk
```

Note: The above is a lab environment. Results vary depending on workload.

Viewing NFS Usage

In ONTAP 8.3.x and later, you can see how many RPC calls have been issued per NFS version on a local node. The values are persistent and clear only when the node reboots. This command is available only from diagnostic privilege level and must be issued on the local node.

```
cluster::> set diag
cluster::*> diag nblade nfs usage show
       Node: node2
                  v3: 120
                  v4: 2076
```

Viewing Active NFS Connections in the Cluster

In ONTAP, it is possible to view active NFS connections across all SVMs and nodes in the cluster using the `network connections active show` command. This command allows filtering of IPs, services, and other features to provide more useful and granular information. The command can be used in place of classic `netstat` commands found in 7-Mode.

Example:

```
cluster::> network connections active show
    show           show-clients   show-lifs      show-protocols show-services

cluster::> network connections active show -node node1 -service nfs*
              Vserver    Interface          Remote
      CID Ctx Name       Name:Local Port    Host:Port            Protocol/Service
--------- --- --------- ----------------- -------------------- ----------------
Node: node1
286571835   6 vs0          data:2049         x.x.x.z:763    TCP/nfs

cluster::> network connections active show -node node2 -service nfs*
There are no entries matching your query.
```

Additionally, it is possible to view network connections in a LISTEN state with `network connections listening show`.

Mapping NFS Clients to Specific Volumes

ONTAP 9.7 introduced a new command that allows storage administrators to see which clients are mounted through NFS to specific volumes in the cluster.

The use cases are varied, but usually fall into:

- Need to discover who's using a volume before performing migrations, cutovers, etc.
- Troubleshooting issues.
- Load distribution.

To view connected NFS clients:

```
cluster::> nfs connected-clients show ?
  [ -instance | -fields <fieldname>, ... ]
  [[-node] <nodename>]                                              Node Name
  [[-vserver] <vserver>]                                            Vserver
  [[-data-lif-ip] <IP Address>]                                     Data LIF IP Address
  [[-client-ip] <IP Address>]                                       Client IP Address
  [[-volume] <volume name>]                                         Volume Accessed
  [[-protocol] <Client Access Protocol>]                            Protocol Version
  [ -idle-time <[<integer>d][<integer>h][<integer>m][<integer>s]> ] Idle Time (Sec)
  [ -local-reqs <integer> ]                                         Number of Local Reqs
  [ -remote-reqs <integer> ]                                        Number of Remote Reqs
```

For sample output, see "NFS Connected-Clients Sample Output."

10 Advanced NFS Concepts

This section covers NFS concepts that extend outside of the realm of basic configuration.

10.1 Unmask

In NFS operations, permissions can be controlled through mode bits, which leverage numerical attributes to determine file and folder access. These mode bits determine read, write, execute, and special attributes. Numerically, these are represented as:

- Execute = 1
- Read = 2
- Write = 4

Total permissions are determined by adding or subtracting a combination of the preceding.

For example:

```
4 + 2 + 1 = 7 (can do everything)
4 + 2 = 6 (rw) and so on…
```

For more information about UNIX permissions, see the UNIX permissions help page.

Umask is a functionality that allows an admin to restrict the level of permissions allowed to a client. By default, the umask for most clients is set to 0022, which means that files created from that client are assigned that umask. The umask is subtracted from the base permissions of the object. If a volume has 0777 permissions and is mounted using NFS to a client with a umask of 0022, objects written from the client to that volume have 0755 access (0777 – 0022).

```
# umask
0022
# umask -S
u=rwx,g=rx,o=rx
```

However, many operating systems do not allow files to be created with execute permissions, but they do allow folders to have the correct permissions. Thus, files created with a umask of 0022 might end up with permissions of 0644.

The following is an example using RHEL 6.5:

```
# umask
0022
# cd /cdot
# mkdir umask_dir
# ls -la | grep umask_dir
drwxr-xr-x.  2 root     root         4096 Apr 23 14:39 umask_dir

# touch umask_file
# ls -la | grep umask_file
-rw-r--r--.  1 root     root            0 Apr 23 14:39 umask_file
```

10.2 NFS User nfsnobody

In some cases, NFS clients might see file owner/group information show up in file listings as nfsnobody.

```
# ls -la | grep newfile
-rwxrwxrwx   1 nfsnobody nfsnobody          0 May 19 13:30 newfile.txt
```

When you list the file with numerics, you find that the owner:group is 65534:

```
# ls -lan | grep newfile
-rwxrwxrwx   1 65534 65534          0 May 19 13:30 newfile.txt
```

The user 65534 on most Linux clients is `nfsnobody`, but in ONTAP, that user is `pcuser`."

```
cluster::*> unix-user show -vserver DEMO -id 65534
               User            User   Group  Full
Vserver        Name            ID     ID     Name
-------------- --------------- ------ ------ --------------------------------
DEMO           pcuser          65534  65534
```

It's also the default "anonymous" user in export-policy rules:

```
cluster::*> export-policy rule show -vserver DEMO -policyname default -fields anon
vserver policyname ruleindex anon
------- ---------- --------- -----
DEMO    default    1         65534
DEMO    default    2         65534
DEMO    default    3         65534
```

When you look at the file permissions from the ONTAP cluster, you might see that the UNIX owner is indeed 65534, but that there are also Windows ACLs and owners that are different:

```
cluster::*> vserver security file-directory show -vserver DEMO -path /data/newfile.txt

                Vserver: DEMO
              File Path: /data/newfile.txt
      File Inode Number: 7088
         Security Style: ntfs
        Effective Style: ntfs
         DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
           UNIX User Id: 65534
          UNIX Group Id: 65534
         UNIX Mode Bits: 777
 UNIX Mode Bits in Text: rwxrwxrwx
                   ACLs: NTFS Security Descriptor
                         Control:0x8004
                         Owner:NTAP\ntfs
                         Group:NTAP\DomainUsers
                         DACL - ACEs
                           ALLOW-Everyone-0x1f01ff-(Inherited)
```

When you see “nfsnobody” or 65534 in NFS listings, one of two things are probably happening:

- The volume that is being exported to NFS clients is also used by Windows SMB clients and the Windows users writing to the shares don’t map to valid UNIX users and/or groups.
- The volume that is being exported to NFS clients has the anonymous user set to 65534 and something is causing the NFS user to “squash” to the anonymous user. See “The Anon User” for more details on user squashing.

You can view Windows user to UNIX user mapping with the following command in advanced privilege:

```
cluster::*> access-check name-mapping show -vserver DEMO -direction win-unix -name ntfs
'ntfs' maps to 'pcuser'

cluster::*> access-check name-mapping show -vserver DEMO -direction win-unix -name prof1
'prof1' maps to 'prof1'
```

10.3 NFSv4.x – nobody:nobody

One of the most common issues seen with NFSv4.x configuration is when a file or folder is shown in a listing using “ls” as being owned by the user:group combination of nobody:nobody.

For example:

```
sh-4.2$ ls -la | grep prof1-file
-rw-r--r-- 1 nobody nobody    0 Apr 24 13:25 prof1-file
```

And the numeric ID is 99:

```
sh-4.2$ ls -lan | grep prof1-file
-rw-r--r-- 1 99 99    0 Apr 24 13:25 prof1-file
```

On the cluster (and using NFSv3), that file ownership appears to have a proper UID/GID:

```
cluster::*> vserver security file-directory show -vserver DEMO -path /home/prof1/prof1-file

                Vserver: DEMO
              File Path: /home/prof1/prof1-file
      File Inode Number: 9996
         Security Style: unix
        Effective Style: unix
```

```
         DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
           UNIX User Id: 1002
          UNIX Group Id: 10002
         UNIX Mode Bits: 644
 UNIX Mode Bits in Text: rw-r--r--
                   ACLs: -
```

In some cases, the file might show the correct owner, but “nobody” as the group:

```
sh-4.2$ ls -la | grep newfile1
-rw-r--r-- 1 prof1  nobody    0 Oct  9  2019 newfile1
```

Who is “nobody” anyway?

The “nobody” user in NFSv4.x is different from the “nfsnobody” user mentioned in “NFS User nfsnobody.” You can view how an NFS client sees each user with the “id” command.

```
# id nobody
uid=99(nobody) gid=99(nobody) groups=99(nobody)
# id nfsnobody
uid=65534(nfsnobody) gid=65534(nfsnobody) groups=65534(nfsnobody)
```

With NFSv4.x, the “nobody” user is the default user defined by the `idmapd.conf` file and can be defined as any user you want to use.

```
# cat /etc/idmapd.conf | grep nobody
#Nobody-User = nobody
#Nobody-Group = nobody
```

Why does this happen?

Because security through name string mapping is a key tenet of NFSv4.x operations, the default behavior when a name string does not match properly is to squash that user to one that will not normally have any access to files and folders owned by users and groups.

When you see “nobody” for the user and/or group in file listings, that generally means something in NFSv4.x is misconfigured. Case sensitivity can come into play here.

For example, if user1@NTAP.LOCAL (uid 1234, gid 1234) is accessing an export, then ONTAP must be able to find user1@NTAP.LOCAL (uid 1234, gid 1234). If the user in ONTAP is USER1@NTAP.LOCAL, then it will not match. In many cases, you can see the following in the messages file on the client:

```
May 19 13:14:29 centos7 nfsidmap[17481]: nss_getpwnam: name 'root@defaultv4iddomain.com' does not
map into domain 'NTAP.LOCAL'
May 19 13:15:05 centos7 nfsidmap[17534]: nss_getpwnam: name 'nobody' does not map into domain
'NTAP.LOCAL'
```

The client and server must both agree that a user is indeed who they are claiming to be, so the following should be checked to make sure that the user that the client sees has the same information as the user that ONTAP sees.

- NFSv4.x ID domain (Client: `idmapd.conf` file; ONTAP: `-v4-id-domain` option)
- User name and numeric IDs (Name service switch configuration – client: `nsswitch.conf` and/or local passwd and group files; ONTAP: `ns-switch` commands)
- Group name and numeric IDs (Name service switch configuration – client: `nsswitch.conf` and/or local passwd and group files; ONTAP: `ns-switch` commands)

In almost all cases, if you see “nobody” in user and group listings from clients but ONTAP reports the correct user and group information (through vserver security file-directory show), the issue is user or group name domain ID translation.

You can also make use of the ONTAP option `-v4-numeric-ids`, covered in “Bypassing the name string – Numeric IDs.”

10.4 Hiding Snapshot Copies

A Snapshot copy is a read-only, point-in-time image of a volume. The image consumes minimal storage space and incurs negligible performance overhead because it records only changes to files since the last Snapshot copy was made.

Users can access Snapshot copies to restore individual files from NFS clients. In some workloads however, applications might list the contents of the NFS mount. In some cases, the .snapshot directory gets included, which can add considerable time to these scans – especially in high file count environments. In those cases, the Snapshot directory can be hidden from NFS mounts using the NFS server option `-v3-hide-snapshot`.

As evidenced by the command, this only affects NFSv3 clients, as the .snapshot directory is already hidden for NFSv4.x.

Even when the `.snapshot` directory is hidden, it is still accessible. To remove access to the .snapshot directory to clients, use the volume option `-snapdir-access`.

Note: These options don’t take effect until the export is remounted.

10.5 Viewing and Managing NFS Credentials

In ONTAP 9.3, a global cache for name services was implemented to offer better performance, reliability, resilience and supportability for NAS credentials and name service server management.

Part of those changes was the implementation of the NFS credential cache, which stores NFS user and group information in ONTAP when NFS exports are accessed.

These caches can be viewed and managed through the advanced privilege `nfs credentials` commands.

```
cluster::*> nfs credentials ?
  count                        *Count credentials cached by NFS
  flush                        *Flush credentials cached by NFS
  show                         *Show credentials cached by NFS
```

Cache entries populate the node where the TCP connection for the NFS mount exists. This information can be seen with the following command on the cluster:

```
cluster::*> nfs connected-clients show -vserver DEMO -client-ip x.x.x.x -fields data-lif-ip -
volume scripts
node               vserver data-lif-ip   client-ip     volume  protocol
------------------ ------- ------------- ------------- ------- --------
Node1              DEMO    x.x.x.y       x.x.x.x       scripts nfs3
```

From the command above, we know the client IP x.x.x.x is connected to a data LIF on node1. That helps us narrow down which node to focus on for cache entries.

The `nfs credentials count` command allows you to see how many credentials are currently stored in the NFS credential cache. This can be useful in understanding the effect of clearing the cache.

```
cluster::*> nfs credentials count -node node1
Number of credentials cached by NFS on node "node1": 4
```

If a user traverses into an ONTAP NFS export, user IDs, group IDs, etc. are all added to the NFS credential cache. For example, we have a user named `prof1`.

```
# id prof1
```

```
uid=1102(prof1) gid=10002(ProfGroup) groups=10002(ProfGroup),10000(Domain
Users),1202(group2),1101(group1),1220(sharedgroup),1203(group3)
```

That user has 8 different entries – a numeric UID and 7 group memberships. Then, the user `prof1` accesses an NFS export. Our credential cache increases by 8.

```
cluster::*> nfs credentials count -node node1
Number of credentials cached by NFS on node "node1": 12
```

That count is for the entire node, not just per SVM. If you have multiple SVMs in your environment, the count might not be useful for troubleshooting.

Viewing the NFS Credential Cache

In addition to showing how many credentials are in the NFS credential cache, we can also view individual cache entries for users and/or groups. If a user in your environment complains about having access issues, you can look for that user in the cache.

Note: You cannot view the contents of entire credential cache.

In our example, `prof1` accessed the mount. We can view that cache entry, as well as the flags that tell us more about the cache entry.

```
cluster::*> nfs credentials show -node node1 -vserver DEMO -unix-user-name prof1

Credentials
-----------
                     Node: node1
                  Vserver: DEMO
                Client IP: -
                    Flags: unix-extended-creds-present, id-name-mapping-present
  Time since Last Refresh: 52s
   Time since Last Access: 44s
                Hit Count: 4

  UNIX Credentials:
                    Flags: 1
                Domain ID: 0
                      UID: 1102
              Primary GID: 10002
          Additional GIDs: 10002
                           10000
                           1101
                           1202
                           1203
                           1220

  Windows Credentials:
                    Flags: -
                 User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: user
                       ID: 1102
                     Name: prof1
```

We can also see an entry for the user's primary group.

```
cluster::*> nfs credentials show -node node1 -vserver DEMO -unix-group-name ProfGroup

Credentials
-----------
                     Node: node1
                  Vserver: DEMO
                Client IP: -
                    Flags: id-name-mapping-present
```

```
  Time since Last Refresh: 64s
   Time since Last Access: 6s
                Hit Count: 2

  UNIX Credentials:
                    Flags: -
                Domain ID: -
                      UID: -
              Primary GID: -
          Additional GIDs: -

  Windows Credentials:
                    Flags: -
                 User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: group
                       ID: 10002
                     Name: ProfGroup
```

You can also view credential cache entries for users and groups down to the client IP that tried the access.

```
cluster::*> nfs credentials show -node node1 -vserver DEMO -client-ip x.x.x.x -unix-user-id 1102

Credentials
-----------
                          Node: node1
                       Vserver: DEMO
                     Client IP: x.x.x.x
                         Flags: unix-extended-creds-present, id-name-mapping-present
       Time since Last Refresh: 35s
        Time since Last Access: 34s
                     Hit Count: 2
               Reference Count: 4
 Result of Last Update Attempt: no error

  UNIX Credentials:
                    Flags: 1
                Domain ID: 0
                      UID: 1102
              Primary GID: 10002
          Additional GIDs: 10002
                           10000
                           1101
                           1202
                           1203
                           1220

  Windows Credentials:
                    Flags: -
                 User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: user
                       ID: 1102
                     Name: prof1
```

The credential cache also keeps negative entries (entries that could not be resolved) in cache. Negative entries occur when ONTAP can’t resolve the numeric UID to a valid user. In this case, the UID 1236 cannot be resolved by ONTAP, but attempted access to the NFS export.

```
# su cifsuser
bash-4.2$ cd /scripts/
bash: cd: /scripts/: Permission denied
```

```
bash-4.2$ id
uid=1236(cifsuser) gid=1236(cifsuser) groups=1236(cifsuser)

cluster::*> nfs credentials show -node node1 -vserver DEMO -unix-user-id 1236

Credentials
-----------
                     Node: node1
                  Vserver: DEMO
                Client IP: -
                    Flags: no-unix-extended-creds, no-id-name-mapping
  Time since Last Refresh: 33s
   Time since Last Access: 7s
                Hit Count: 15

  UNIX Credentials:
                    Flags: -
                Domain ID: -
                      UID: -
              Primary GID: -
          Additional GIDs: -

  Windows Credentials:
                    Flags: -
                 User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: -
                       ID: -
                     Name: -
```

NFS Credential Cache with NFSv4.x and Multiprotocol NAS

The NFS credential cache entries also store Windows credentials and NFSv4 ID mapping credentials.

When a user traverses NFSv4.x exports and maps into the ID domain correctly, we'd see the `ID-Name Information` field populated.

```
Credentials
-----------
                          Node: node
                       Vserver: DEMO
                     Client IP: x.x.x.x
                         Flags: unix-extended-creds-present, id-name-mapping-present
       Time since Last Refresh: 12s
        Time since Last Access: 9s
                     Hit Count: 2
               Reference Count: 4
 Result of Last Update Attempt: no error

  UNIX Credentials:
                    Flags: 1
                Domain ID: 0
                      UID: 1102
              Primary GID: 10002
          Additional GIDs: 10002
                           10000
                           1101
                           1202
                           1203
                           1220

  Windows Credentials:
                    Flags: -
                 User SID: -
        Primary Group SID: -
              Domain SIDs: -
  ID-Name Information:
                     Type: user
                       ID: 1102
                     Name: prof1
```

If the user accesses an export that has NTFS permissions/security style, we'd see the flag `cifs-creds-present`, as well as the domain SID information under `Windows Credentials`:

```
Credentials
-----------
                          Node: node1
                       Vserver: DEMO
                     Client IP: x.x.x.x
                         Flags: ip-qualifier-configured, unix-extended-creds-present, cifs-creds-
present
       Time since Last Refresh: 19s
        Time since Last Access: 1s
                     Hit Count: 9
               Reference Count: 2
 Result of Last Update Attempt: no error

  UNIX Credentials:
                    Flags: 0
                Domain ID: 0
                      UID: 1102
              Primary GID: 10002
          Additional GIDs: 10002
                           10000
                           1101
                           1202
                           1203
                           1220

  Windows Credentials:
                    Flags: 8320
                 User SID: S-1-5-21-3552729481-4032800560-2279794651-1214
        Primary Group SID: S-1-5-21-3552729481-4032800560-2279794651-513
              Domain SIDs: S-1-5-21-3552729481-4032800560-2279794651
                           S-1-18
                           S-1-1
                           S-1-5
                           S-1-5-32

  ID-Name Information:
                     Type: -
                       ID: -
                     Name: -
```

NFS Credential Cache Settings

The timeout value of the NFS credential cache is controlled by the following NFS server options.

Table 17) NFS credential cache settings.

Option	What It Does	Default Value (ms)
`-cached-cred-negative-ttl`	This optional parameter specifies the age of the negative cached credentials after which they are cleared from the cache. The value specified must be between 60,000 and 604,800,000.	7,200,000
`-cached-cred-positive-ttl`	This optional parameter specifies the age of the positive cached credentials after which they are cleared from the cache. The	86,400,000 (24 hours)

Option	What It Does	Default Value (ms)
	value specified must be between 60,000 and 604,800,000.	
`-cached-cred-harvest-timeout`	This optional parameter specifies the harvest timeout for cached credentials. The value specified must be between 60,000 and 604,800,000.	86400000 (24 hours)

Cache entries maintain the time since last access/refresh (as seen the the “show” command). If an entry stays idle for a period of time, it eventually is removed from the cache. If the entry is active, it gets refreshed and stays in cache.

These values can be modified to longer or shorter timeout values, depending on the desired effects.

- **Longer cache timeout values** reduce network load and provide faster lookups of users, but can produce more false positives/false negatives as the cache entries are not always in sync with name services.
- **Shorter cache timeout values** increase load on the network and name servers, and can add some latency to name lookups (depending on name service source), but offer more accurate and up-to-date entries.

The best practice is to leave the values as is. If you need to change the values, be sure to monitor the results and adjust as needed.

Flushing the NFS Credential Cache

In cases where a user has been added or removed from a group and does not have the desired access, the credential cache entry can be flushed manually, rather than waiting for the cache entry to timeout.

The command can be run for a UNIX user or numeric ID or a UNIX group or numeric ID. Additionally, the command can be run as granularly as down to the client IP address having the issue.

```
cluster::*> nfs credentials flush -node node1 -vserver DEMO -client-ip x.x.x.x -unix-user-id 1102

Number of matching credentials flushed: 2
```

Note: You can only flush one NFS credential cache entry at a time.

The NFS credential cache is separate from the name-service cache.

10.6 Using NFSv4.x ACLs with NFSv3 mounts

NFSv3, by default, has a fairly limited way to manage permissions. However, it is possible in ONTAP to set NFSv4.x ACLs on files and folders and have them apply to NFSv3 exports.

The method for doing this is pretty straightforward. It involves these steps:

1. Configure and enable NFSv4.x in ONTAP and on your client
2. Enable NFSv4.x ACL support in ONTAP
3. Mount the export through NFSv4.x
4. Apply the NFSv4.x ACLs
5. Unmount and then remount the export using NFSv3 and test it out.

In my environment, I mounted a `homedir` volume and then set up an ACL on a file owned by root for a user called `prof1` using `nfs4_setfacl -e` (which allows you to edit a file rather than have to type in a long command).

The file lives in the root user’s homedir. The root homedir is set to 755, which means anyone can read them, but no one but the owner (root) can write to them.

```
drwxr-xr-x 2 root root 4096 Jul 13 10:42 root
```

That is, unless I set NFSv4.x ACLs to allow a user full control.

```
[root@centos7 mnt]# nfs4_getfacl /mnt/root/file
A::prof1@ntap.local:rwaxtTnNcCy
A::OWNER@:rwaxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy
```

I can also see those permissions from the ONTAP CLI.

```
cluster::*> vserver security file-directory show -vserver DEMO -path /home/root/file

Vserver: DEMO
 File Path: /home/root/file
 File Inode Number: 8644
 Security Style: unix
 Effective Style: unix
 DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
 UNIX User Id: 0
 UNIX Group Id: 1
 UNIX Mode Bits: 755
 UNIX Mode Bits in Text: rwxr-xr-x
 ACLs: NFSV4 Security Descriptor
 Control:0x8014
 DACL - ACEs
 ALLOW-user-prof1-0x1601bf
 ALLOW-OWNER@-0x1601bf
 ALLOW-GROUP@-0x1200a9-IG
 ALLOW-EVERYONE@-0x1200a9
```

In the above example, I gave `prof1` full control over the file. Then, I mounted through NFSv3. When I become a user that isn’t on the NFSv4.x ACL, I can’t write to the file or remove the file (the expected behavior).

```
[root@centos7 /]# su student1
sh-4.2$ cd /mnt/root
sh-4.2$ ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul 13 10:42 .
drwxrwxrwx 11 root root 4096 Jul 10 10:04 ..
-rwxr-xr-x 1 root bin 0 Jul 13 10:23 file
-rwxr-xr-x 1 root root 0 Mar 29 11:37 test.txt

sh-4.2$ touch file
touch: cannot touch ‘file’: Permission denied
sh-4.2$ rm file
rm: remove write-protected regular empty file ‘file’? y
rm: cannot remove ‘file’: Permission denied
```

When I change to the `prof1` user, I have access to do whatever I want, even though the mode bit permissions in v3 say I shouldn’t be able to. That’s because the NFSv4.x ACL is working.

```
[root@centos7 /]# su prof1
sh-4.2$ cd /mnt/root
sh-4.2$ ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul 13 10:42 .
drwxrwxrwx 11 root root 4096 Jul 10 10:04 ..
-rwxr-xr-x 1 root bin 0 Jul 13 10:23 file
-rwxr-xr-x 1 root root 0 Mar 29 11:37 test.txt
```

```
sh-4.2$ vi file
sh-4.2$ cat file
NFSv4ACLS!
```

When I do a chmod, however, nothing seems to change from the NFSv4 ACL for the user. I set 700 on the file, which shows up in NFSv3 mode bits.

```
sh-4.2$ chmod 700 file
sh-4.2$ ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul 13 10:42 .
drwxrwxrwx 11 root root 4096 Jul 10 10:04 ..
-rwx------ 1 root bin 11 Aug 11 09:58 file
-rwxr-xr-x 1 root root 0 Mar 29 11:37 test.txt
```

But notice how the `prof1` user still has full control.

```
cluster::*> vserver security file-directory show -vserver DEMO -path /home/root/file

Vserver: DEMO
 File Path: /home/root/file
 File Inode Number: 8644
 Security Style: unix
 Effective Style: unix
 DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
 UNIX User Id: 0
 UNIX Group Id: 1
 UNIX Mode Bits: 700
 UNIX Mode Bits in Text: rwx------
 ACLs: NFSV4 Security Descriptor
 Control:0x8014
 DACL - ACEs
 ALLOW-user-prof1-0x1601bf
 ALLOW-OWNER@-0x1601bf
 ALLOW-GROUP@-0x120088-IG
 ALLOW-EVERYONE@-0x120088
```

That’s because NFSv4.x ACL preserve is enabled. If that option is disabled, chmod would wipe the ACL.

10.7 Dynamic NAS TCP Autotuning test

ONTAP introduces dynamic NAS TCP autotuning, which enables the NAS protocol stack to adjust buffer sizes on the fly to the most optimal setting. This capability is needed because static methods to set TCP buffer sizes do not consider the dynamic nature of networks nor the range of different types of connections made to a system at one time. Autotuning is used to optimize the throughput of NAS TCP connections by computing the application data read rate and the rate of the data being received by the system to compute optimal buffer size. The feature is not configurable and only increases buffer sizes; buffers never decrease. The starting value for this is 32K. Autotuning applies to individual TCP connections, rather than on a global scale.

This not the same as the max transfer size value (`-tcp-max-xfer-size`) included under the NFS server options.

Max Transfer Size Settings

In Lenovo ONTAP, `-v3-tcp-max-read-size` and `-v3-tcp-max-write-size` have been deprecated. The recommendation is to leverage the option `-tcp-max-xfer-size` instead. This change also allows TCP transfer sizes of 1MB for both reads and writes. ONTAP versions prior to ONTAP 9 only allowed 1MB for reads.

Note: If these values are adjusted, they affect only new mounts. Existing mounts maintain the block size that was set at the time of the mount. If the sizes are changed, existing mounts can experience rejections of write operations or smaller responses for reads than requested. Whenever you change block size options, make sure that clients are unmounted and remounted to reflect those changes.

Why Dynamic Window Size?

Most environments do not benefit from static TCP window sizes, because window sizes are generally considered in the context of a single host or connection. On a server, such as NFS running on ONTAP, there are multiple connections to multiple hosts. Each connection has its own uniqueness and requires varying degrees of throughput. With a static window, a server becomes extremely limited in how it can handle the diversity of inbound connections. Participants in network infrastructures often change and rarely are static; thus, the TCP stack needs to be able to handle those participants in an efficient and effective manner. Dynamic window sizes help prevent the caveats seen in static window environments, such as overutilizing a network and creating a throughput collapse or underutilizing a network and experiencing less operating efficiency over time.

10.8 Network Connection Concurrency: NFSv3

In addition to the preceding considerations, it's worth noting that ONTAP has a limit of 128 concurrent operations per TCP connection for NFSv3 operations. This limit means that for every IP address, the system can handle only up to 128 concurrent operations. Therefore, it's possible that an NFSv3 client would not be able to push the storage system hard enough to reach the full potential of the FlexGroup technology. Clients can be configured to control the number of concurrent operations (by using RPC slot tables) that are sent through NFSv3, which can help avoid hard-to-track performance issues.

Identifying Potential Issues with RPC Slot Tables

Many modern NFSv3 clients use dynamic values for RPC slot tables, which means that the client sends as many concurrent operations on a single TCP thread as possible—up to 65,336. However, ONTAP allows only 128 concurrent operations per TCP connection, so if a client sends more than 128, ONTAP enacts a form of flow control on NFSv3 operations to prevent rogue clients from overrunning storage systems by blocking the NFS operation (exec contexts in ONTAP) until resources free up. This flow control can manifest as performance issues that cause extra latency and slower job completion times that might not have a readily apparent reason from the general storage system statistics. These issues can appear to be network related, which can send storage administrators down the wrong troubleshooting path.

To investigate whether RPC slot tables might be involved, use the ONTAP performance counter. You can check whether the number of exec contexts blocked by the connection being overrun is incrementing.

To gather those statistics, run the following command:

```
statistics start -object cid -instance cid
```

Then, review the statistics over a period of time to see if they are incrementing.

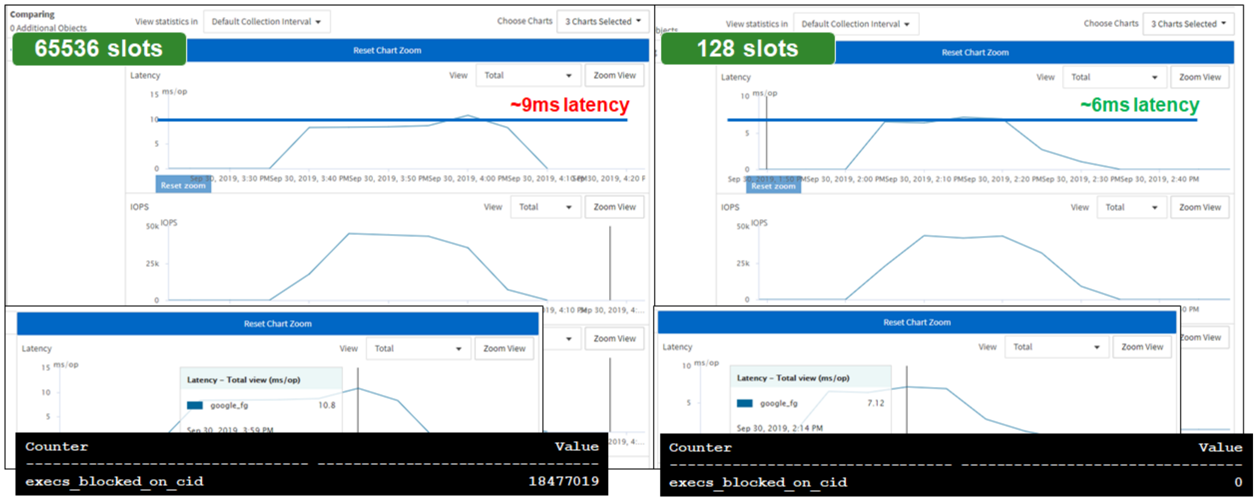
```
statistics show -object cid -instance cid -counter execs_blocked_on_cid
```

Example #1: RPC Slot Table Effect on Performance – High File Count Workload

In the following example, a script was run to create 18 million files across 180,000 subdirectories. This load generation was done from three clients to the same NFS mount. The goal was to generate enough NFS operations with clients that had the default RPC slot table settings to cause ONTAP to enter a flow-control scenario. Then the same scripts were run again on the same clients—but with the RPC slot tables set to 128.

The result was that the default slot tables (65,536) generated 18 million `execs_blocked_on_cid` events and added 3ms of latency to the workload versus the run with the lower RPC slot table setting (128).

Figure 19) Effect of RPC slot tables on NFSv3 performance.



```
Counter                                                   Value
-------------------------------- ------------------------------
execs_blocked_on_cid                                   18477019
```

```
Counter                                                   Value
-------------------------------- ------------------------------
execs_blocked_on_cid                                          0
```

Although 3ms might not seem like a lot of latency, it can add up over millions of operations, considerably slowing down job completion.

Example #2: RPC Slot Table Effect on Performance – Sequential I/O Workload

In the section called "Performance Examples for Different TCP Max Transfer Window Sizes," we show a number of tests that illustrate NFSv3 vs. NFSv4.1 performance differences, along with different wsize/rsize mount option values. While running these tests, we also saw the negative effects of RPC slot tables increasing the number of execs blocked on CIDs causing performance bottlenecks that added 14.4ms of write latency to some of the performance runs, which in turn added 5.5 minutes to the overall job completion times.

The tests were run across two clients on a 10GB network, using a script that runs multiple dd operations in parallel. Overall, 8 folders with two 16GB files each were created and then read and deleted.

- When the RPC slots were set to the maximum dynamic value of 65536, the dd operations took 20 minutes, 53 seconds.
- When the RPC slots were lowered to 128, the same script took just 15 minutes, 23 seconds.

With the 1MB wsize/rsize mount options and 65536 RPC slots, the `execs_blocked_on_cid` incremented to ~1,000 per node.

```
cluster::*> statistics show -object cid -counter execs_blocked_on_cid

Scope: node1

    Counter                                                   Value
    -------------------------------- ------------------------------
    execs_blocked_on_cid                                       1001

Scope: node2

    Counter                                                   Value
    -------------------------------- ------------------------------
    execs_blocked_on_cid                                       1063
```

Figure 20 shows the side-by-side comparison of latency, IOPS, and throughput for the jobs using a 1MB wsize/rsize mount value.

Figure 20) Parallel dd performance – NFSv3 and RPC slot tables; 1MB rsize/wsize.

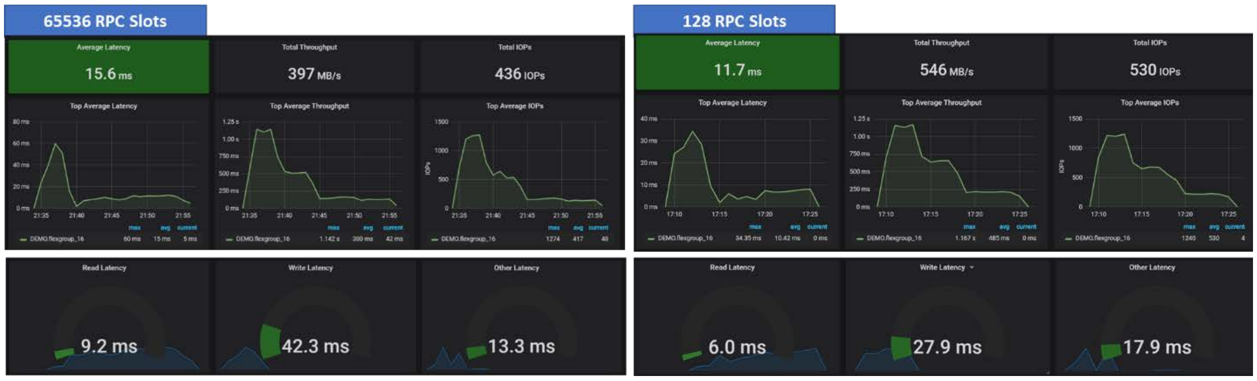


Figure 21 shows the side-by-side comparison of latency, IOPS, and throughput for the jobs using a 256K wsize/rsize mount value.

Figure 21) Parallel dd performance – NFSv3 and RPC slot tables; 256K rsize/wsize.

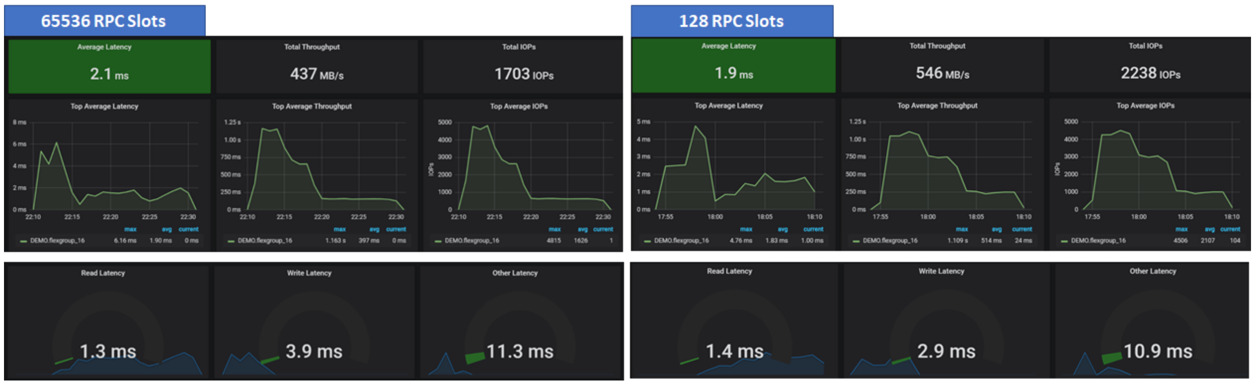


Table 19 shows the comparison of job times and latency with 65,536 and 128 RPC slots.

Table 18) Job comparisons - parallel dd with 65,536 and 128 RPC slots.

Test	Average Read Latency (ms)	Average Write Latency (ms)	Completion Time
NFSv3 – 1MB wsize/rsize; 65 536 slot tables	9.2 (+3.2ms)	42.3 (+14.4ms)	20m53s (+5m30s)
NFSv3 – 1MB wsize/rsize; 128 slot tables	6	27.9	15m23s
NFSv3 – 256K wsize/rsize; 65536 Slot tables	1.3 (-.1ms)	3.9 (+1ms)	19m12s (+4m55s)
NFSv3 – 256K wsize/rsize; 128 slot tables	1.4	2.9	14m17s
NFSv3 – 64K wsize/rsize; 65536 slot tables	.2 (0ms)	3.4 (+1.2ms)	17m2s (+2m14s)
NFSv3 – 64K wsize/rsize; 128 slot tables	.2	2.2	14m48s

Resolving Issues with RPC Slot Tables

ONTAP cannot control the number of slot tables a client sends per TCP connection for NFSv3 operations. Therefore, clients must be configured to limit the maximum slot tables sent through NFS to no more than 128. The configuration of this setting varies depending on the client OS version. Contact the client vendor for more information.

Additionally, the value to set the RPC slot tables to might vary on workload and number of clients. For example, in the tests above, 128 RPC slots using a 1MB wsize/rsize achieved a job completion time of 15 minutes, 23 seconds. Setting the RPC slots to 64 lowered that job completion time to 14 minutes, 2 seconds. However, setting the value too low can have the opposite effect, because the clients can be throttled unnecessarily. Test different values in your environment for the best possible results.

Table 19) Job comparisons - parallel dd with 65,536, 128, and 64 RPC slots.

Test	Average Read Latency (ms)	Average Write Latency (ms)	Completion Time
NFSv3 – 1MB wsize/rsize; 65536 slot tables	9.2 (+3.2ms)	42.3 (+19.4ms)	20m53s (+6m51s)
NFSv3 – 1MB wsize/rsize; 128 slot tables	6 (-.3ms)	27.9 (+5ms)	15m23s (+1m21s)
NFSv3 – 1MB wsize/rsize; 64 slot tables	6.3	22.9	14m2s

It is possible to get more performance out of a client’s NFS connectivity by connecting more mount points to different IP addresses in the cluster on the same client, but that approach can create complexity. For example, rather than mounting a volume at `SVM:/volumename`, multiple mount points on the same client across different folders and IP addresses in the volume could be created.

For example:

```
LIF1:/volumename/folder1
LIF2:/volumename/folder2
LIF3:/volumename/folder3
```

Another possible option is to use the nconnect option available for some Linux distributions that can perform multiplexing of NFSv3 over the same TCP connection. This option provides more available concurrent sessions and better overall performance.

Does the RPC Slot Table Limit Affect Other NAS Protocols?

RPC slot table limits affect only NFSv3 traffic:

- SMB clients use different connection methodologies for concurrency, such as SMB multichannel, SMB multiplex, and SMB credits. The SMB connection methodology depends on client/server configuration and protocol version. For example, SMB 1.0 uses SMB multiplex (mpx), whereas SMB2.x uses SMB credits.
- NFSv4.x clients do not use RPC slot tables—instead, they use state IDs and session tables to control the flow of concurrent traffic from clients.

Table 21 shows the test run results from NFSv3 and NFSv4.1 using the same 65,536 slot table values.

Table 20) Job comparisons - parallel dd – NFSv3 and NFSv4.1 with 65536 RPC slots.

Test	Average Read Latency (ms)	Average Write Latency (ms)	Completion Time
NFSv3 – 1MB wsize/rsize; 65,536 slot tables	9.2 (+2.7ms)	42.3 (+5.5ms)	20m53s (+5m47s)
NFSv4.1 – 1MB wsize/rsize; 65,536 slot tables	6.5	36.8	15m6s
NFSv3 – 256K wsize/rsize; 65,536 slot tables	1.3 (-.1ms)	3.9 (+.7ms)	19m12s (+7m2s)
NFSv4.1 – 256K wsize/rsize; 65,536 slot tables	1.4	3.2	12m10s
NFSv3 – 64K wsize/rsize; 65,536 slot tables	.2 (+.1ms)	3.4 (+2.2ms)	17m2s (+1m54s)
NFSv4.1 – 64K wsize/rsize; 65,536 slot tables	.1	1.2	15m8s

10.9 NFSv4.x Client IDs/NFS4ERR_CLID_INUSE

When NFSv4.x clients mount NFS exports in ONTAP, each client sends its hostname as part of the RPC packet in the NFS mount packet. The NFS server responds with a client ID to keep track of the NFSv4.x session.

If another NFSv4.x client with the same hostname (but a different IP address), then the NFS server responds with an error:

```
50 11.856227 10.x.x.x 10.x.x.y NFS V4 Call (Reply In 51) EXCHANGE_ID
51 11.856407 10.x.x.y 10.x.x.x NFS V4 Reply (Call In 50) EXCHANGE_ID Status: NFS4ERR_CLID_INUSE
```

As per the NFSv4.1 RFC standard, this is a security mechanism.

(https://tools.ietf.org/html/rfc5661#section-2.10.8.3)

```
2.10.8.3.  Protection from Unauthorized State Changes

As described to this point in the specification, the state model of NFSv4.1 is vulnerable to an
attacker that sends a SEQUENCE operation with a forged session ID and with a slot ID that it
expects the legitimate client to use next.  When the legitimate client uses the slot ID with the
same sequence number, the server returns the attacker's result from the reply cache, which
disrupts the legitimate client and thus denies service to it.  Similarly, an attacker could send
a CREATE_SESSION with a forged client ID to create a new session associated with the client
ID.  The attacker could send requests using the new session that change locking state, such as
LOCKU operations to release locks the legitimate client has acquired. Setting a security policy
on the file that requires RPCSEC_GSS credentials when manipulating the file's state is one
potential work around, but has the disadvantage of preventing a legitimate client from releasing
state when RPCSEC_GSS is required to do so, but a GSS context cannot be obtained (possibly
because the user has logged off the client).

…

The SP4_MACH_CRED state protection option uses a machine credential where the principal that
creates the client ID MUST also be the principal that performs client ID and session maintenance
operations. The security of the machine credential state protection approach depends entirely on
safe guarding the per-machine credential. Assuming a proper safeguard using the per-machine
credential for operations like CREATE_SESSION, BIND_CONN_TO_SESSION, DESTROY_SESSION, and
DESTROY_CLIENTID will prevent an attacker from associating a rogue connection with a session, or
associating a rogue session with a client ID.
```

From RFC-5661, this is what that error means:

https://tools.ietf.org/html/rfc5661#section-15.1.13.2

```
 15.1.13.2.  NFS4ERR_CLID_INUSE (Error Code 10017)
While processing an EXCHANGE_ID operation, the server was presented with a co_ownerid field that
matches an existing client with valid leased state, but the principal sending the EXCHANGE_ID
operation differs from the principal that established the existing client. This indicates a
collision (most likely due to chance) between clients.  The client should recover by changing the
co_ownerid and re-sending EXCHANGE_ID (but not with the same slot ID and sequence ID; one or both
MUST be different on the re-send).
```

In these cases, the NFSv4.x client should retry the mount with a new client ID. In some cases, use of the NFS mount option `-clientaddr` might be needed. For more information, see RedHat Bugzilla #1821903.

10.10 NFS Silly Renames

With NFS, when a file is locked by a client, if the file is deleted, then NFS performs what's called a silly rename, where the file is renamed to an `.nfs*` name. In these scenarios, if an application needs to recursively delete directories, then these deletes fail, because the directory are not technically empty.

The NFSv4.1 RFC offers a new result flag called `OPEN4_RESULT_PRESERVE_UNLINKED` that addresses silly renames.

```
The use of the OPEN4_RESULT_PRESERVE_UNLINKED result flag allows a client to avoid the common
implementation practice of renaming an open file to ".nfs<unique value>" after it removes the
file.  After the server returns OPEN4_RESULT_PRESERVE_UNLINKED, if a client sends a REMOVE
operation that would reduce the file's link count to zero, the server SHOULD report a value of
zero for the numlinks attribute on the file.
```

This flag is currently not supported in the ONTAP NFSv4.1 implementation.

10.11 NAS Flow Control

ONTAP also provides NAS-level flow control. This flow control mechanism is separate from the TCP flow control enabled on the NICs and switches of the data network. It is always on and implemented at the NAS protocol stack to prevent rogue clients from overloading a node in the cluster and creating a denial of service (DoS) scenario. This flow control affects all NAS traffic (CIFS and NFS).

How It Works

If a client sends too many packets to a node, the flow control adjusts the window size to 0 and tells the client to wait on sending any new NAS packets until the other packets are processed. If the client continues to send packets during this “zero window,” then the NAS protocol stack flow control mechanism sends a TCP reset to that client.

Contact Lenovo Technical Support if you suspect a performance problem related to NAS flow control.

10.12 Using FlexClone to Quickly Change All UIDs/GIDs in a Volume

In high file count workloads, such as software development environments, occasionally a set of files might be owned by, and access is limited to, a specific user or group. If another developer needs access to these files, the dataset would need to have the owner/permissions changed. If there are millions of files and folders, that operation would take a long time. Additionally, we don’t want to break access for other users in production, so we’d need a way to solve this issue quickly and without disruption.

Lenovo FlexClone volumes provide a way to create an instant snapshot-backed copy of a volume without taking up space in ONTAP, but they also provide a way to quickly change file and folder ownership for everything in that clone. If you want those changes to be permanent, you would split the clone to its own volume (which would then use up capacity in ONTAP). When I ran the following command to clone a volume with over a million files and folders *and* change the UID and GID on all of them, the command took around 10 seconds to complete:

```
cluster::*> vol clone create -vserver DEMO -flexclone clone -parent-vserver DEMO -parent-volume
flexvol -junction-path /clone -uid 1100 -gid 1101
[Job 12606] Job succeeded: Successful
```

10.13 Auxiliary GIDs – Addressing the 16 GID Limitation for NFS

NFS has a specific limitation for the maximum number of auxiliary GIDs (secondary groups) that can be honored in a single NFS request. The maximum for AUTH_SYS/AUTH_UNIX is 16, and for AUTH_GSS (Kerberos) it is 32. This is a known protocol limitation of NFS.

ONTAP provides the ability to increase the maximum number of auxiliary groups to 1024 by avoiding truncation of the group list in the NFS packet by prefetching the requesting user’s group from a name service.

```
auth-sys-extended-groups
extended-groups-limit
```

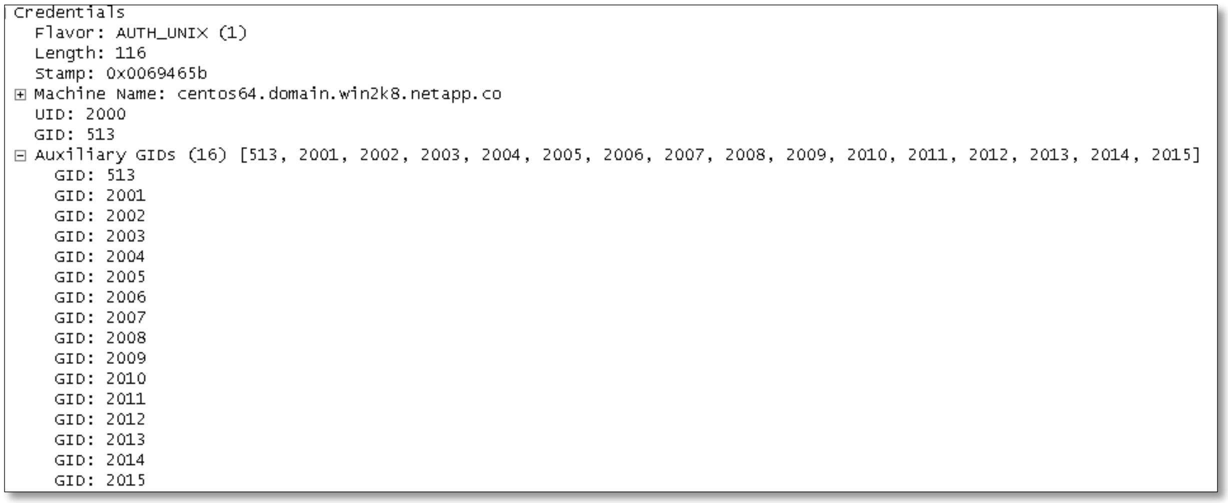
How It Works

The options to extend the group limitation work just the way that the `manage-gids` option for other NFS servers works. Basically, rather than dumping the entire list of auxiliary GIDs a user belongs to, the option does a lookup for the GID on the file or folder and returns that value instead.

From the man page for mountd:

```
-g  or  --manage-gids

Accept requests from the kernel to  map  user  id  numbers  into lists  of group  id  numbers for
use in access control.  An NFS request will normally except when using Kerberos or other
cryptographic  authentication)  contains  a  user-id  and  a list of group-ids.  Due to a
limitation in the NFS protocol, at most  16 groups ids can be listed.  If you use the -g flag,
then the list of group ids received from the client will be replaced by a list of  group ids
determined by an appropriate lookup on the server.
```

When an access request is made, only 16 GIDs are passed in the RPC portion of the packet.



```
Credentials
  Flavor: AUTH_UNIX (1)
  Length: 116
  Stamp: 0x0069465b
⊞ Machine Name: centos64.domain.win2k8.netapp.co
  UID: 2000
  GID: 513
⊟ Auxiliary GIDs (16) [513, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015]
    GID: 513
    GID: 2001
    GID: 2002
    GID: 2003
    GID: 2004
    GID: 2005
    GID: 2006
    GID: 2007
    GID: 2008
    GID: 2009
    GID: 2010
    GID: 2011
    GID: 2012
    GID: 2013
    GID: 2014
    GID: 2015
```

Any GID past the limit of 16 is dropped by the protocol. Extended GIDs can be used with external name services, or locally on the cluster if the users and groups are configured properly. To make sure that a local UNIX user is a member of multiple groups, use the `unix-group adduser(s)` command:

```
COMMANDS
     adduser - Add a user to a local UNIX group

     addusers - Add a list of users to a local UNIX group
```

Performance Effect of Extended GIDs

Extended groups have a minimal performance penalty, generally in the low single digit percentages. Higher metadata NFS workloads would likely have more effect, particularly on the system’s caches. Performance can also be affected by the speed and workload of the name service servers. Overloaded name service servers are slower to respond, causing delays in prefetching the GID.

Considerations for Numeric ID Authentication (NFSv3 and NFSv4.x)

NFSv3 using AUTH_SYS sends numeric ID information for users and groups to perform user authentication to NFS mounts for permission resolution.

NFSv4.x with ONTAP has a feature that allows NFSv4.x mounts to leverage numeric ID strings instead of name strings, which allows NFSv4.x operations without needing centralized name services, matching names/numeric IDs on client/server, matching ID domains, etc. (`-v4-numeric-ids`)

Enabling the `-auth-sys-extended-groups` option causes numeric ID authentication to fail if the UNIX user numeric ID can't be translated into a valid UNIX user name in name services. This counteracts the `-v4-numeric-ids` option, as ONTAP needs to query the incoming numeric user ID to search for any auxiliary groups for authentication. If the incoming numeric ID cannot be resolved to a valid UNIX user or the client's UNIX numeric UID is different than the numeric UID ONTAP knows about, then the lookup fails with `secd.authsys.lookup.failed` in the event log and ONTAP responds to the client with the AUTH_ERROR Client Must Begin a New Session," which appears as Permission Denied.

To use both options, use the following guidance:

- If you require users and groups that either cannot be queried from both NFS client and server or have mismatched numeric IDs, you can leverage NFS Kerberos and NFSv4.x ACLs to provide proper authentication with NFSv4.x, as clients send name strings instead of numeric IDs.
- If you are using `-auth-sys-extended-groups` with AUTH_SYS and without NFSv4.x ACLs, any user that requires access through NFS requires a valid UNIX user in the name service database specified in ns-switch (can also be a local user).

For more information about the `-v4-numeric-ids` option, see "Bypassing the name string – Numeric IDs."

Using Local Files

Using the `-auth-sys-extended-groups` option allows support for >16 auxiliary groups and is best applied with external name services, such as LDAP. However, if an external name service server is unavailable, you can leverage local files to create UNIX users and groups for support with `-auth-sys-extended-groups`. Simply create the users and groups you want to use and use the `unix-group adduser` command to populate the auxiliary group list.

Considerations for Active Directory LDAP

By default, in Microsoft Active Directory LDAP servers, the MaxPageSize attribute is set to a default of 1000. That means groups beyond 1000 would get truncated in LDAP queries. To enable full support with the 1024 value for extended groups, the MaxPageSize attribute must be modified to reflect the 1,024 value.

For more information about how to change that value, see the Microsoft TechNet article titled, "How to view and set LDAP policy in Active Directory by using Ntdsutil.exe."

For concerns about modifying this value, contact Microsoft support and/or review the TechNet library article titled, "MaxPageSize is set too high."

10.14 High File Count Considerations

This section covers general NAS and high file-count environment considerations.

Volume Style Recommendation for High File-Count Environments

High file count refers to a workload that creates and stores hundreds of thousands to upwards of billions of files in a single namespace. Datasets like these generate many metadata operations, which can affect performance on a FlexVol volume. As a result, in high file count environments, a FlexGroup volume is highly recommended.

Inode Count Considerations

An inode in ONTAP is a pointer to any file or folder within the file system, including Snapshot copies. Each FlexVol volume has a finite number of inodes and has an absolute maximum of 2,040,109,451. The default or maximum number of inodes on a FlexVol volume depends on the volume size and has a ratio of one inode to 32KB of capacity. Inodes can be increased after a FlexVol volume has been created and can be reduced starting in ONTAP 8.0.

A FlexGroup volume can handle up to 400 billion files, but the default inode count for 200 FlexVol members in a FlexGroup volume is 4,250,225,200. This count is based on the following formula:

```
200 member volumes * 21,251,126 default inodes per member = 4,250,225,200 total default inodes
```

If the FlexGroup volume requires more inodes than what is presented as a default value, the inodes must be increased by using the `volume modify -files` command.

Table 22 shows a sample of FlexVol sizes, inode defaults, and maximums.

Table 21) Inode defaults and maximums according to FlexVol size.

FlexVol Size	Default Inode Count	Maximum Inode Count
20MB*	566	4,855
1GB*	31,122	249,030
100GB*	3,112,959	24,903,679
1TB	21,251,126	255,013,682
10TB	21,251,126	2,040,109,451
100TB	21,251,126	2,040,109,451

*FlexGroup member volumes should not be any smaller than 100GB in size.

When you use a FlexGroup volume, the total default inode count depends on both the total size of the FlexVol members and the number of FlexVol members in the FlexGroup volume.

Table 23 shows various examples of FlexGroup configurations and the resulting default inode counts.

Table 22) Inode defaults resulting from FlexGroup member sizes and member volume counts.

Member Volume Size	Member Volume Count	Default Inode Count (FlexGroup)
100GB	8	24,903,672
100GB	16	49,807,344
1TB	8	170,009,008
1TB	16	340,018,016
100TB	8	170,009,008
100TB	16	340,018,016

High File Counts, Low Capacity Needs

ONTAP allocates a default inode and maximum inode count based on volume capacity. In Table 22, member volumes smaller than 10TB is not able to achieve the two billion inodes available to a FlexVol volume. To get two billion inodes per member volume, the member volume capacity would need to be

10TB or greater. In a FlexGroup volume with eight member volumes, that would support 16 billion files, but would also provision 80TB of storage.

This can present a challenge to high-file-count environments, because file sizes might be small and will not require that much total capacity. For example, if all files in a workload are 288 bytes each in size, 16 billion files would use up only about 4.6TB. To achieve a maximum of 16 billion files, you'd need at least eight member volumes that are 10TB each, which would take up 80TB. 4.6TB is around 6% of the total capacity needed to contain 16 billion files in ONTAP in an 80TB FlexGroup volume. In these cases, you would need to disable space guarantees to avoid reserving about 75TB of unused space.

When deploying high file counts that use up little capacity, there are two main options for deploying the FlexGroup volume:

- **Deploy the FlexGroup volume with 10TB or greater member volumes with thin provisioning.** Thin provisioning a volume simply means that you are telling ONTAP a volume will be a certain size, but that the size is not guaranteed in the file system. This provides flexibility in the file system to limit storage allocation to physical space. However, other volumes in the aggregate can affect the free capacity, so it's important to monitor available aggregate space when using thin provisioning.
- **Manually create the FlexGroup volume with more member volumes than the default.** If you want to keep space guarantees for volumes, another option for high-file-count/small capacity environments is to create more member volumes in a FlexGroup volume.

  Because inode counts are limited per FlexVol member volume according to capacity, adding more smaller member volumes can provide for higher file counts at the same capacity.

Tips for Managing Slow Directory Listings in High-File-Count Environments

Some workflows in high-file-count environments include running `find`, `ls`, or some other read metadata-heavy operation on an existing dataset. Generally, this process is inefficient and can take a long time to complete. If it's necessary to run these operations, there are a few things you can try to help speed things along.

Generally speaking, the issue with these types of operations are client, protocol, or network related. The storage rarely is the bottleneck for read metadata slowness. ONTAP is able to multithread read metadata operations. With `ls` operations, `getattr` requests are sent one at a time, in serial, which means for millions of `getattr` operations, there might be millions of network requests to the storage. Each network request incurs n milliseconds of latency, which adds up over time. So, there are a few ways to speed these up:

- **Send more `getattr` requests at a time.** By itself, `ls` can't send requests in parallel. But with utilities like the XCP Migration Tool, it is possible to send multiple threads across the network to greatly speed up `ls` operations. Using XCP scan can help with speed, depending on what the `ls` output is being used for later. For example, if you need the user permissions/owners of the files, using `ls` by itself might be a better fit. But for sheer listing of file names, XCP scan is preferable.
- **Add more network hardware (for example, 100GB instead of 10GB) to reduce round-trip time (RTT).** With larger network pipes, more traffic can be pushed over the network, thus reducing load and potentially reducing overall RTT. With million of operations, even shaving off a millisecond of latency can add up to a large amount of time saved for workloads.
- **Run `ls` without unnecessary options, such as highlighting/colors.** When running `ls`, the default behavior is to add sorting, colors, and highlighting for readability. These add work for the operation, so it might make sense to run `ls` with the `-f` option to avoid those potentially unnecessary features.
- **Cache `getattr` operations on the client more aggressively.** Client-side caching of attributes can help reduce the network traffic for operations, as well as bringing the attributes local to the client for operations. Clients manage NFS caches differently, but in general, avoid setting `noac` on NFS mounts for high-file-count environments. Also, keep `actimeo` to a level no less than 30 seconds.

- **Create FlexCache volumes.** Lenovo FlexCache volumes are able to create instant caches for read-heavy workloads. Creating FlexCache volumes for workloads that do a lot of read metadata operations, such as `ls`, can have the following benefits:
  - For local clusters, it can help offload the read metadata operations from the origin volume to the cache volumes, and, as a result, frees the origin volume up for regular reads and writes.
  - FlexCache volumes can reside on any node in a cluster, so creating FlexCache volumes makes the use of cluster nodes more efficient by allowing multiple nodes to participate in these operations, in addition to moving the read metadata operations away from the origin node.
  - For remote clusters across a WAN, FlexCache volumes can provide localized NFS caches to help reduce WAN latency, which can greatly improve performance for read-metadata-heavy workloads.

  When using FlexCache volumes to help read metadata workloads, be sure to disable `fastreaddir` on the nodes that use FlexCache.

```
cluster::> node run “priv set diag; flexgroup set fastreaddir=false persist
```

Note: For this to take effect, a reboot/storage failover is required.

Starting in ONTAP 9.7, FlexGroup volumes can be origins for FlexCache volumes.

Effect of Being Out of Inodes

When a volume runs out of inodes, no more files can be created in that volume until the number of inodes is increased or existing inodes are freed. The effect to the client is that ONTAP reports that the volume has “no space left on device.”

When a volume runs out of inodes, the cluster triggers an EMS event (`callhome.no.inodes`), and a Lenovo AutoSupport® message is triggered.

64-Bit File Identifiers

By default, NFS in ONTAP uses 32-bit file IDs. 32-bit file IDs are limited to 2,147,483,647 maximum unsigned integers. With the 2 billion inode limit in FlexVol, this value fits nicely into the architecture.

However, because FlexGroup volumes can officially support up to 400 billion files in a single container (and theoretically, many more), the implementation of 64-bit file IDs was needed. 64-bit file IDs support up to 9,223,372,036,854,775,807 unsigned integers.

Note: Lenovo strongly recommends enabling the NFS server option `-v3-64bit-identifiers` at the advanced privilege level before you create a FlexGroup volume, especially if your file system exceeds or might exceed the two billion inode threshold.

The 64-bit file identifier option is set to “off/disabled” by default. This was by design, to make certain that legacy applications and operating systems that require 32-bit file identifiers were not unexpectedly affected by ONTAP changes before administrators could properly evaluate their environments. Check with your application and OS vendor for their support for 64-bit file IDs before enabling them. Alternatively, create a test SVM and enable it to see how applications and clients react with 64-bit file IDs. Most modern applications and operating systems can handle 64-bit file IDs without issue.

This option can currently be enabled only with the advanced privilege level on the command line:

```
cluster::> set advanced
cluster::*> nfs server modify -vserver SVM -v3-64bit-identifiers enabled
```

After enabling or disabling this option, you must remount all clients. Otherwise, because the file system IDs change, the clients might receive stale file handle messages when attempting NFS operations. For more information about how enabling or disabling FSID change options can affect SVMs in high-file-count environments.

If a FlexGroup volume does not exceed two billion files, you can leave this value unchanged. However, to prevent any file ID conflicts, the inode maximum on the FlexGroup volume should also be increased to no more than 2,147,483,647.

```
cluster::*> vol show -vserver SVM -volume flexgroup -fields files
```

Note: This option does not affect SMB operations and is unnecessary with volumes that use only SMB.

NFSv3 Versus NFSv4.x – File IDs

NFSv3 and NFSv4.x use different file ID semantics. Now that FlexGroup volumes support NFSv4.x, ONTAP 9.7 provides two different options for enabling/disabling 64-bit file IDs.

When you use both NFSv3 and NFSv4.x in an SVM and you want the 64-bit ID option to apply to both protocols, you must set both options.

If only one option is set and volumes are accessed by both protocols, you might see undesired behavior between protocols. For instance, NFSv3 might be able to create and view more than 2 billion files, whereas NFSv4.x would throw an error.

The options are:

```
-v3-64bit-identifiers [enabled/disabled]
-v4-64bit-identifiers [enabled/disabled]
```

Effect of File ID Collision

If 64-bit file IDs are not enabled, the risk for file ID collisions increases. When a file ID collision occurs, the effect can range from a stale file handle error on the client, to the failure of directory and file listings, to the entire failure of an application. Usually, it is imperative to enable the 64-bit file ID option when you use FlexGroup volumes.

You can check a file’s ID from the client using the `stat` command. When an inode or file ID collision occurs, it might look like the following. The inode is 3509598283 for both files.

```
[root@client]# stat libs/
  File: `libs/'
  Size: 12288           Blocks: 24         IO Block: 65536  directory
Device: 4ch/76d Inode: 3509598283  Links: 3
Access: (0755/drwxr-xr-x)  Uid: (60317/   user1)   Gid: (10115/      group1)
Access: 2017-01-06 16:00:28.207087000 -0700
Modify: 2017-01-06 15:46:50.608126000 -0700
Change: 2017-01-06 15:46:50.608126000 -0700

[root@client example]# stat iterable/
  File: `iterable/'
  Size: 4096            Blocks: 8          IO Block: 65536  directory
Device: 4ch/76d Inode: 3509598283  Links: 2
Access: (0755/drwxr-xr-x)  Uid: (60317/   user1)   Gid: (10115/      group1)
Access: 2017-01-06 16:00:44.079145000 -0700
Modify: 2016-05-05 15:12:11.000000000 -0600
Change: 2017-01-06 15:23:58.527329000 -0700
```

A collision can result in issues such as circular directory structure errors on the Linux client and an inability to remove files.

```
rm: WARNING: Circular directory structure.
This almost certainly means that you have a corrupted file system.
NOTIFY YOUR SYSTEM MANAGER.
The following directory is part of the cycle:
  ‘/directory/iterable’

rm: cannot remove ‘/directory’: Directory not empty
```

Note: This option does not affect SMB operations and is unnecessary with volumes using only SMB.

How FSIDs Operate with SVMs in High-File-Count Environments

The FSID change option for NFSv3 and NFSv4.x provides FlexVol and FlexGroup volumes with their own unique file systems, which means that the number of files allowed in the SVM is dictated by the number of volumes. However, disabling the FSID change options causes the 32-bit or 64-bit file identifiers to apply to the SVM itself, meaning that the two billion file limit with 32-bit would apply to all volumes. Therefore, the entire SVM would be soft-limited to two billion files, rather than the individual FlexVol or FlexGroup volume. If the SVM exceeds 2 billion files with the FSID change disabled, clients would potentially see duplicate file IDs. The effect of this is covered in "Effect of File ID Collision."

Leaving the FSID change option enabled allows volumes to operate as independent file systems with their own dedicated file counts.

Lenovo recommends leaving the FSID change option enabled with FlexGroup volumes to help prevent file ID collisions.

Note: The `-v4-fsid-change` option does not apply to FlexGroup volumes prior to ONTAP 9.7, because NFSv4 is unsupported with FlexGroup volumes in those releases.

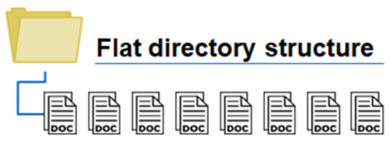
Directory Size Considerations

In ONTAP, there are limitations to the maximum directory size on disk. This limit is known as maxdirsize. The `maxdirsize` value for a volume is capped at 320MB, regardless of platform. This means that the memory allocation for the directory size can reach a maximum of only 320MB before a directory can no longer grow larger.

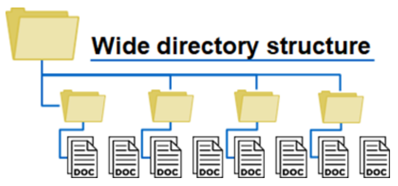
What Directory Structures Can Affect maxdirsize?

The `maxdirsize` value can be a concern when you're using flat directory structures, where a single folder contains millions of files at a single level. Folder structures where files, folders, and subfolders are interspersed have a low effect on `maxdirsize`. There are several directory structure methodologies.

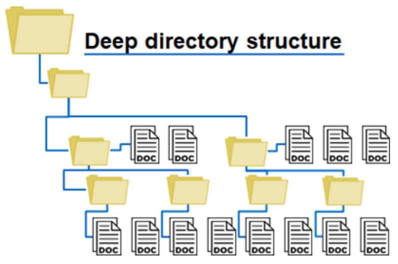
- **Flat directory structure:** a single directory with many files.



- **Wide directory structure.** Many top-level directories with files spread across directories.



- **Deep directory structures.** Fewer top-level directories, but with many subfolders; files spread across directories.



How Flat Directory Structures Can Affect FlexGroup Volumes

Flat directory structures (many files in a single/few directories) have a negative effect on a wide array of file systems, whether they're Lenovo systems or not. Areas of impact can include, but are not limited to:

- Memory pressure
- Network performance/latency (particularly during mass queries of files, `GETATTR` operations, `READDIR` operations, and so on)
- CPU utilization

FlexGroup volumes can also have an extra effect on `maxdirsize`. Unlike a FlexVol volume, a FlexGroup volume uses remote hard links inside ONTAP to help redirect traffic. These remote hard links are what allow a FlexGroup volume to deliver scale-out performance and capacity in a cluster.

However, in flat directories, a higher ratio of remote hard links to local files is seen. These remote hard links count against the total `maxdirsize` value, so a FlexGroup volume might approach the `maxdirsize` limit faster than a FlexVol will.

For example, if a directory has millions of files in it and generates roughly 85% remote hard links for the file system, you can expect `maxdirsize` to be exhausted at nearly twice the amount as a FlexVol volume would.

Querying for Used maxdirsize Values

It is important to monitor and evaluate `maxdirsize` allocation in ONTAP. However, there are no commands for this specific to ONTAP. Instead, `maxdirsize` allocation would need to be queried from the client.

The following command from an NFS client would be able to retrieve the directory size information for a folder inside a FlexGroup volume for the 10 largest directories in a given mount point, while omitting Snapshot copies from the search.

```
# find /mountpoint -name .snapshot -prune -o -type d -ls -links 2 -prune | sort -rn -k 7 | head
```

The following example took less than a second on a dataset in folders with millions of files:

```
[root@centos7 /]# time find /flexgroup/manyfiles/ -name .snapshot -prune -o -type d -ls -links 2
-prune | sort -rn -k 7 | head
787227871 328976 drwxr-xr-x   2 root     root     335544320 May 29 21:23
/flexgroup/manyfiles/folder3/topdir_8/subdir_0
384566806 328976 drwxr-xr-x   2 root     root     335544320 May 29 13:14
/flexgroup/manyfiles/folder3/topdir_9/subdir_0
3605793347 328976 drwxr-xr-x   2 root     root     335544320 May 29 21:23
/flexgroup/manyfiles/folder3/topdir_0/subdir_0
3471151639 328976 drwxr-xr-x   2 root     root     335544320 May 29 13:45
/flexgroup/manyfiles/folder3/topdir_4/subdir_0
2532103978 328976 drwxr-xr-x   2 root     root     335544320 May 29 14:16
/flexgroup/manyfiles/folder3/topdir_2/subdir_0
2397949155 328976 drwxr-xr-x   2 root     root     335544320 May 29 14:15
/flexgroup/manyfiles/folder3/topdir_1/subdir_0
1994984460 328976 drwxr-xr-x   2 root     root     335544320 May 29 13:43
/flexgroup/manyfiles/folder3/topdir_6/subdir_0
```

```
1860674357 328976 drwxr-xr-x   2 root     root     335544320 May 29 13:18
/flexgroup/manyfiles/folder3/topdir_5/subdir_0
1458235096 328976 drwxr-xr-x   2 root     root     335544320 May 29 14:25
/flexgroup/manyfiles/folder3/topdir_3/subdir_0
1325327652 328976 drwxr-xr-x   2 root     root     335544320 May 29 14:25
/flexgroup/manyfiles/folder3/topdir_7/subdir_0

real    0m0.055s
user    0m0.002s
sys     0m0.035s
```

Using XCP to Check maxdirsize

The XCP Migration Tool is mostly considered a RapidData mover, but it also derives value in its robust file scanning abilities. XCP is able to run `find` commands in parallel as well, so the previous examples can be run even faster on the storage system.

The following XCP command example allows you to run `find` only on directories with more than 2,000 entries:

```
# xcp diag find --branch-match True -fmt "'{size} {name}'.format(size=x.digest, name=x)"
localhost:/usr 2>/dev/null | awk '{if ($1 > 2000) print $1 " " $2}'
```

And this XCP command helps you find the directory size values:

```
# xcp -match "type == d" -fmt "'{} {}'.format(used, x)" localhost:/usr | awk '{if ($1 > 100000)
print}' | sort -nr
```

When XCP looks for the directory size values, it scans the file system first. Here's an example:

```
[root@XCP flexgroup]#  xcp -match "type == d" -fmt "'{} {}'.format(used, x)"
x.x.x.a:/flexgroup_16/manyfiles | awk '{if ($1 > 100000) print}' | sort -nr

 660,693 scanned, 54 matched, 123 MiB in (24.6 MiB/s), 614 KiB out (122 KiB/s), 5s
 1.25M scanned, 58 matched, 234 MiB in (22.1 MiB/s), 1.13 MiB out (109 KiB/s), 10s
 …
 31.8M scanned, 66 matched, 5.83 GiB in (4.63 MiB/s), 28.8 MiB out (22.8 KiB/s), 7m52s

Filtered: 31816172 did not match
31.8M scanned, 66 matched, 5.83 GiB in (12.6 MiB/s), 28.8 MiB out (62.4 KiB/s), 7m53s.
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_9/subdir_0
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_8/subdir_0
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_7/subdir_0
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_6/subdir_0
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_5/subdir_0
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_4/subdir_0
336871424 x.x.x.a:/flexgroup_16/manyfiles/folder3/topdir_3/subdir_0
```

Number of Files That Can Fit into a Single Directory with the Default maxdirsize

To determine how many files can fit into a single directory with the default `maxdirsize` setting, use this formula:

- Memory in KB * 53 *25%

Since `maxdirsize` is set to 320MB by default on larger systems, the maximum number of files in a single directory would be 4,341,760 for SMB and NFS on FlexVol volumes.

FlexGroup volumes use remote hardlinks to redirect I/O to member volumes. These hardlinks count against the total directory size, so the maximum number of files allowed with 320MB maxdirsize would depend on the number of hardlinks that were created. The file count per directory might be in the 2-2.6 million range for directories in a FlexGroup volume.

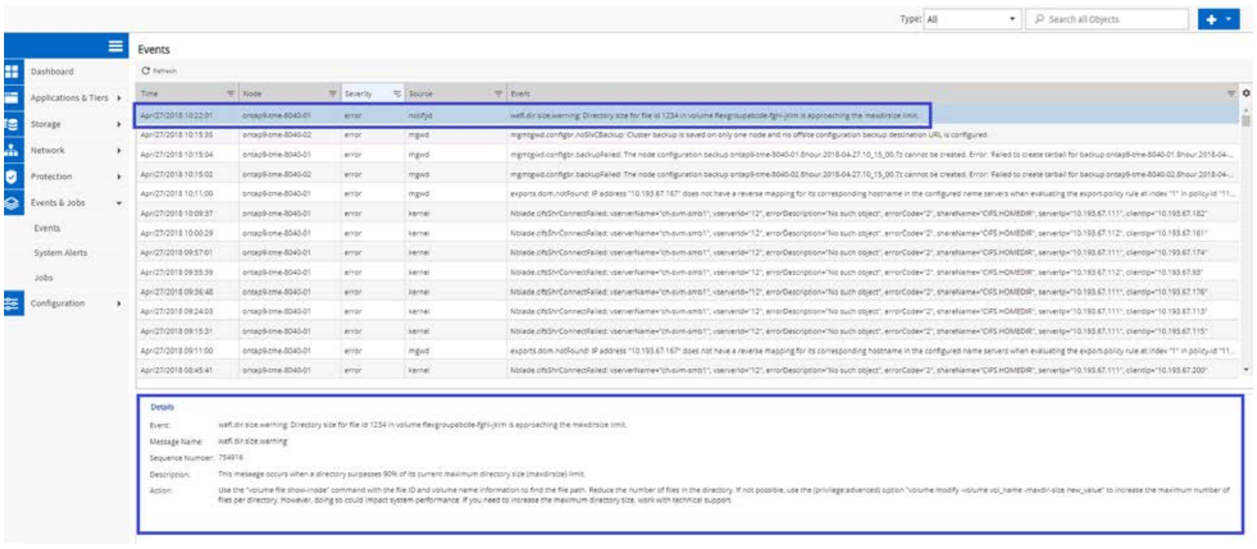
Lenovo strongly recommends that you keep the `maxdirsize` value at the default value.

Event Management System Messages Sent When maxdirsize Is Exceeded

The following event management system (EMS) messages are triggered when `maxdirsize` is either exceeded or close to being exceeded. Warnings are sent at 90% of the `maxdirsize` value and can be viewed with the `event log show` command or with the ThinkSystem Storage Manager for DM Series event section. ThinkSystem Intelligent Monitoring Unified Manager can be used to monitor `maxdirsize`, trigger alarms, and send a notification before the 90% threshold (see Figure 22). These event management system messages also support SNMP traps.

```
wafl.dir.size.max
wafl.dir.size.max.warning
wafl.dir.size.warning
```

Figure 22) ThinkSystem Storage Manager for DM Series event screen with `maxdirsize` warning.



Time	Node	Severity	Source	Event
Apr/27/2018 10:22:01	ontap9-tme-8040-01	error	notifyd	wafl.dir.size.warning: Directory size for file id 1234 in volume flexgroupabcde-fghi-jklm is approaching the maxdirsize limit.
Apr/27/2018 10:15:35	ontap9-tme-8040-02	error	mgwd	mgmtgwd.configbr.noSNCBackup: Cluster backup is saved on only one node and no offsite configuration backup destination URL is configured.
Apr/27/2018 10:15:04	ontap9-tme-8040-01	error	mgwd	mgmtgwd.configbr.backupFailed: The node configuration backup ontap9-tme-8040-01.8hour.2018-04-27.10_15_00.7z cannot be created. Error: 'Failed to create tarball for backup ontap9-tme-8040-01.8hour.2018-04-...
Apr/27/2018 10:15:02	ontap9-tme-8040-02	error	mgwd	mgmtgwd.configbr.backupFailed: The node configuration backup ontap9-tme-8040-02.8hour.2018-04-27.10_15_00.7z cannot be created. Error: 'Failed to create tarball for backup ontap9-tme-8040-02.8hour.2018-04-...
Apr/27/2018 10:11:00	ontap9-tme-8040-01	error	mgwd	exports.dom.notFound: IP address "10.193.67.167" does not have a reverse mapping for its corresponding hostname in the configured name servers when evaluating the export-policy rule at index "1" in policy-id "11...
Apr/27/2018 10:09:37	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.111", clientIp="10.193.67.182"
Apr/27/2018 10:00:29	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.112", clientIp="10.193.67.161"
Apr/27/2018 09:57:01	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.111", clientIp="10.193.67.174"
Apr/27/2018 09:55:39	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.112", clientIp="10.193.67.93"
Apr/27/2018 09:36:48	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.111", clientIp="10.193.67.176"
Apr/27/2018 09:24:03	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.111", clientIp="10.193.67.113"
Apr/27/2018 09:15:31	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.111", clientIp="10.193.67.115"
Apr/27/2018 09:11:00	ontap9-tme-8040-01	error	mgwd	exports.dom.notFound: IP address "10.193.67.167" does not have a reverse mapping for its corresponding hostname in the configured name servers when evaluating the export-policy rule at index "1" in policy-id "11...
Apr/27/2018 08:45:41	ontap9-tme-8040-01	error	kernel	Nblade.cifsShrConnectFailed: vserverName="ch-svm-smb1", vserverId="12", errorDescription="No such object", errorCode="2", shareName="CIFS.HOMEDIR", serverIp="10.193.67.111", clientIp="10.193.67.200"

Details

Event:	wafl.dir.size.warning: Directory size for file id 1234 in volume flexgroupabcde-fghi-jklm is approaching the maxdirsize limit.
Message Name:	wafl.dir.size.warning
Sequence Number:	754916
Description:	This message occurs when a directory surpasses 90% of its current maximum directory size (maxdirsize) limit.
Action:	Use the "volume file show-inode" command with the file ID and volume name information to find the file path. Reduce the number of files in the directory. If not possible, use the (privilege:advanced) option "volume modify -volume vol_name -maxdir-size new_value" to increase the maximum number of files per directory. However, doing so could impact system performance. If you need to increase the maximum directory size, work with technical support.

Effect of Increasing maxdirsize

When a single directory contains many files, the lookups (such as in a `find` operation) can consume large amounts of CPU and memory. Starting in ONTAP 9.2, directory indexing creates an index file for directory sizes exceeding 2MB to help offset the need to perform so many lookups and avoid cache misses. Usually, this helps large directory performance. However, for wildcard searches and `readdir` operations, indexing is not of much use.

Do FlexGroup Volumes Avoid maxdirsize Limitations?

In FlexGroup volumes, each member volume has the same `maxdirsize` setting. Even though a directory could potentially span multiple FlexVol member volumes and nodes, the `maxdirsize` performance effect can still come into play, because directory size is the key component, not individual FlexVol volumes. Directory size is tied to the parent volume and does not divide up across other member volumes. Therefore, the overall size of a directory is still an issue. Thus, FlexGroup volumes do not provide relief for environments facing `maxdirsize` limitations.

Effect of Exceeding maxdirsize

When `maxdirsize` is exceeded in ONTAP, an `out of space` error (`ENOSPC`) is issued to the client and an event management system message is triggered. To remediate this problem, a storage administrator must increase the `maxdirsize` setting or move files out of the directory.

11 NFS on Nontraditional Operating Systems

This section covers NFS use on nontraditional NFS platforms, such as Windows and Apple operating systems.

11.1 NFS on Windows

Red Hat's Cygwin emulates NFS but leverages the SMB protocol rather than NFS, which requires a CIFS license. True Windows NFS is available natively only through Services for Network File System or third-party applications such as Hummingbird/OpenText.

Native Windows NFS in ONTAP

In RFC 1813, the following section covers MS-DOS as a supported client for NFS.

```
   nlm4_share

      struct nlm4_share {
           string      caller_name<LM_MAXSTRLEN>;
           netobj      fh;
           netobj      oh;
           fsh4_mode   mode;
           fsh4_access access;
      };

   This structure is used to support DOS file sharing. The
   caller_name field identifies the host making the request.
   The fh field identifies the file to be operated on. The oh
   field is an opaque object that identifies the host or process
   that is making the request. The mode and access fields
   specify the file-sharing and access modes. The encoding of fh
   is a byte count, followed by the file handle byte array. See
   the description of nlm4_lock for more details.
```

The way that Windows uses NLM is with nonmonitored lock calls. The following nonmonitored lock calls are required for Windows NFS support:

```
NLM_SHARE
NLM_UNSHARE
NLM_NM_LOCK
```

These lock calls were not supported in versions of clustered ONTAP earlier than 8.3.1 or in versions of clustered ONTAP earlier than 8.2.3.

Note: PCNFS, WebNFS, and HCLNFS (legacy Hummingbird NFS client) are not supported with ONTAP software and there are no plans to include support for these clients.

Considerations for Using Windows NFS in ONTAP

Keep the following considerations in mind when using Windows NFS with clustered ONTAP.

- Network Status Monitor (NSM) is not supported in Windows NFS. Therefore, volume moves and storage failovers can cause disruptions that might not be seen on NFS clients that do support NSM.
- If using Windows NFS on an SVM, the following options need to be set to "disabled."

```
enable-ejukebox
v3-connection-drop
```

Note: These options are enabled by default. Disabling them does not harm other NFS clients but might cause some unexpected behavior.

- Always mount Windows NFS using the mount option `mtype=hard`.

- When using Windows NFS, the showmount option should be enabled. Otherwise, renames of files and folders on Windows NFS clients might fail.

```
cluster::> nfs server modify -vserver SVM -showmount enabled
```

- Windows NFS clients are not able to properly see the used space and space available through the `df` commands.

Example of mounting NFS in Windows:

```
C:\>mount -o mtype=hard \\x.x.x.e\unix  Z:
Z: is now successfully connected to \\x.x.x.e\unix

The command completed successfully.
```

Enabling Windows NFS Support

In ONTAP, there is a specific NFS server option that needs to be toggled to “enabled” to allow Windows NFS clients. This option is disabled by default. If reverting from a ONTAP version that supports Windows NFS, this option must be disabled prior to attempting the revert.

```
cluster::> nfs server show -vserver nfs_svm -fields v3-ms-dos-client
vserver v3-ms-dos-client
------- ----------------
nfs_svm disabled
```

Windows NFS Use Cases

Windows NFS can be used to provide access to Lenovo storage devices on Windows operating systems in lieu of a CIFS license. Additional use cases include:

- Applications that run on Windows and require NFS connectivity and/or Linux-style commands and functions (such as GETATTR, SETATTR, and so on)
- When a user wants to leverage the NFS protocol rather than CIFS
- Where a user wants to avoid multiprotocol connectivity

Although Windows can be used to leverage NFS connectivity, it might make more sense to use CIFS and the newer features of the latest SMB version that Windows provides for performance and NDO functionality. Additionally, using Windows NFS with clustered ONTAP requires some considerations, covered later.

11.2 NFS Using Apple OS

NFS mounts are also possible using Apple OS using the Finder or terminal windows. For complete mount options in the Apple OS, use the `man mount_nfs` command in a terminal window. When using Apple clients for NFS, there are some things to keep in mind.

Dynamic Versus Static UIDs

When using a Mac with Active Directory, the default behavior of the Mac is to dynamically create a UID/GID based on the Windows SID of the user. In many cases, this is sufficient, but if control over the UIDs and GIDs is needed (such as integration with an existing LDAP server), then static UIDs can be leveraged. For information about best practices for using Apple OS with Active Directory, see the white paper called Best Practices for Integrating OS X with Active Directory**.**

Apple OS Disables Root by Default

Apple disables the root user (UID 0) by default in its OS. Users are instead required to log in with a user name other than root and use sudo if performing root-level tasks. It is possible to reenable the root user.

Apple UIDs Start at 501

The Apple UID structure starts at UID 501. This UID is not a default UNIX user in clustered ONTAP, nor does it exist in most name service servers. This situation is the same for every Apple OS client in an environment, so it is possible that multiple users exist with the same UID. The options to handle this are as follows:

- Create a user on the cluster or in a name service server with UID 501 to authenticate all Apple users.
- Change the UID on the Apple OS for each user who intends to use NFS on Apple.

Use of Apple NFS with NTFS Security-Style Volumes

Apple NFS handles NTFS security-style volumes differently than Linux NFS clients. Therefore, copies/writes to an NFS mount using Finder applications fail by default when NTFS security style is used. This issue occurs when the Apple client attempts an EXCLUSIVE CREATE operation on the file, which is only allowed by SMB clients in ONTAP.

As a workaround, the NFS server option `–ntfs-unix-security-ops` can be set to ignore to allow NTFS security–style volumes to work properly with NFS mounts on Apple.

NFS Rootonly Operations Do Not Work as Expected with Apple OS

In the section called “The rootonly Options—nfsrootonly and mountrootonly,” rootonly operations are described. By default, `-mount-rootonly` is enabled, and `-nfs-rootonly` is disabled. Apple OS behavior when mounting using NFS defaults is to always use reserved ports for the MOUNT protocol and nonreserved ports for the NFS protocol. The Linux NFS mount option of resvport/noresvport applies in the Apple OS, but noresvport does not control the client’s MOUNT port sent to the cluster. Therefore, Apple NFS clients always use ports in range <1024 for MOUNT.

There presently is not a known method to change this behavior, so Apple technical support would need to be engaged to use nonreserved ports for NFS MOUNT calls. For NFSv4.x mounts, this does not matter, because NFSv4.x does not leverage the MOUNT protocol. NFS client ports for NFS operations (port 2049) can be controlled using the resvport/noresvport mount options, but the NFS server option on the cluster would need to be toggled to honor the client behavior. Doing so would affect all versions of NFS.

Additionally, when attempting to mount with the resvport option specified in the Apple OS, the `sudo` command would need to be used, because root is disabled and the `-users` option is not specified by default.

Note: When using the Finder to mount NFS, mount options cannot be specified.

Appendix A: Examples, Configuration Output, and Other Information

This appendix contains command examples, configuration output, and other information that would have cluttered the main sections of this document.

Examples

This section shows examples of commands and of NFS feature functionality.

Examples of Controlling the Root User

In some cases, storage administrators might want to control how the root user is handled from specific clients. These examples show how the approaches work.

Squashing Root

The following examples show how to squash root to anon in various configuration scenarios.

Example 1: Root is squashed to the anon user for all clients.

This approach uses superuser for all NFS clients using sec=sys; other sec types are denied access.

```
cluster::> vserver export-policy rule show –policyname root_squash -instance
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_squash
                                 Rule Index: 1
                            Access Protocol: nfs     ← only NFS is allowed (NFSv3 and v4)
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   ← all clients
                             RO Access Rule: sys     ← only AUTH_SYS is allowed
                             RW Access Rule: sys     ← only AUTH_SYS is allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                   Superuser Security Types: none    ← superuser (root) squashed to anon user
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_squash

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_squash

[root@nfsclient mnt]# ls -la
total 116
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nobody nobody      0 Apr 24 11:33 root_squash

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3     0     0 106496 Apr 24  2013 .
dr-xr-xr-x. 26     0     0   4096 Apr 24 11:24 ..
drwxrwxrwx. 12     0     0   4096 Apr 24 11:05 .snapshot
drwxr-xr-x.  2     0     1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 65534 65534      0 Apr 24  2013 root_squash

[root@nfsclient /]# mount -o sec=krb5 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol
```

Example 2: Root is squashed to the anon user using superuser for a specific client.

In this example, `sec=sys` and `sec=none` are allowed.

```
cluster::> vserver export-policy rule show –policyname root_squash_client -instance
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_squash_client
                                 Rule Index: 1
                            Access Protocol: nfs     ← only NFS is allowed (NFSv3 and v4)
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x     ← just this client
                             RO Access Rule: sys,none       ← AUTH_SYS and AUTH_NONE are allowed
                             RW Access Rule: sys,none       ← AUTH_SYS and AUTH_NONE are allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                   Superuser Security Types: none    ← superuser (root) squashed to anon user
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true
```

```
cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_squash_client

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_squash_client

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nfsnobody nfsnobody      0 Apr 24  2013 root_squash_client

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3     0     0 106496 Apr 24  2013 .
dr-xr-xr-x. 26     0     0   4096 Apr 24 11:24 ..
drwxrwxrwx. 12     0     0   4096 Apr 24 11:05 .snapshot
drwxr-xr-x.  2     0     1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 65534 65534      0 Apr 24  2013 root_squash_client
```

Example 3: Root is squashed to the anon user using superuser for a specific set of clients.

This approach uses `sec=krb5` (Kerberos) and only NFSv4 and CIFS are allowed.

```
cluster::> vserver export-policy rule show –policyname root_squash_krb5 -instance
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_squash_krb5
                                 Rule Index: 1
                            Access Protocol: nfs4,cifs      ← only NFSv4 and CIFS are allowed
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.0/24  ← just clients with an IP
address of 10.10.100.X
                             RO Access Rule: krb5    ← only AUTH_RPCGSSD is allowed
                             RW Access Rule: krb5    ← only AUTH_RPCGSSD is allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                   Superuser Security Types: none    ← superuser (root) squashed to anon user
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_squash

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol

[root@nfsclient /]# mount -t nfs4 cluster:/nfsvol /mnt
mount.nfs4: Operation not permitted

[root@nfsclient /]# mount -t nfs4 -o sec=krb5 krbsn:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_squash_krb5

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nobody nobody      0 Apr 24 11:50 root_squash_krb5

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
```

```
-rw-r--r--.  1 99 99      0 Apr 24 11:50 root_squash_krb5
```

Note: The UID of 99 in this example occurs in NFSv4 when the user name cannot map into the NFSv4 domain. A look at `/var/log/messages` confirms this:

```
Apr 23 10:54:23 nfsclient nfsidmap[1810]: nss_getpwnam: name 'pcuser' not found in domain
nfsv4domain.netapp.com'
```

In the preceding examples, when the root user requests access to a mount, it maps to the anon UID. In this case, the UID is 65534. This mapping prevents unwanted root access from specified clients to the NFS share. Because “sys” is specified as the rw and ro access rules in the first two examples, only clients using `sec=sys` gain access. The third example shows a possible configuration using Kerberized NFS authentication. Setting the access protocol to NFS allows only NFS access to the share (including NFSv3 and NFSv4). If multiprotocol access is desired, then the access protocol must be set to allow NFS and CIFS. NFS access can be limited to only NFSv3 or NFSv4 here as well.

Root Is Root (no_root_squash)

The following examples show how to enable the root user to come into an NFS share as the root user. This is also known as `no_root_squash`.

Example 1: Root is allowed access as root using superuser for all clients only for `sec=sys`.

In this example, `sec=none` and `sec=sys` are allowed rw and ro access; all other anon access is mapped to 65534.

```
cluster::> vserver export-policy rule show –policyname root_allow_anon_squash -instance
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_allow_anon_squash
                                 Rule Index: 1
                            Access Protocol: nfs     🡐 only NFS is allowed (NFSv3 and v4)
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0  🡐 all clients
                             RO Access Rule: sys,none       🡐 AUTH_SYS and AUTH_NONE allowed
                             RW Access Rule: sys,none       🡐 AUTH_SYS and AUTH_NONE allowed
User ID To Which Anonymous Users Are Mapped: 65534  🡐 mapped to 65534
                   Superuser Security Types: sys     🡐 superuser for AUTH_SYS only
               Honor SetUID Bits in SETATTR: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_allow_anon_squash

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_anon_squash_nfsv3

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root      root      106496 Apr 24  2013 .
dr-xr-xr-x. 26 root      root        4096 Apr 24 11:24 ..
drwxrwxrwx. 12 root      root        4096 Apr 24 11:05 .snapshot
drwxr-xr-x.  2 root      bin         4096 Apr 18 12:54 junction
-rw-r--r--.  1 root      root           0 Apr 24  2013 root_allow_anon_squash_nfsv3

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  0      0 Apr 24 11:56 root_allow_anon_squash_nfsv3
```

Example 2: Root is allowed access as root using superuser for `sec=krb5` only.

In this example, anon access is mapped to 65534; `sec=sys` and `sec=krb5` are allowed, but only using NFSv4.

```
cluster::> vserver export-policy rule show –policyname root_allow_krb5_only -instance
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_allow_krb5_only
                                 Rule Index: 1
                            Access Protocol: nfs4    🡐 only NFSv4 is allowed
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   🡐 all clients
                             RO Access Rule: sys,krb5       🡐 AUTH_SYS and AUTH_RPCGSS allowed
                             RW Access Rule: sys,krb5       🡐 AUTH_SYS and AUTH_RPCGSS allowed
User ID To Which Anonymous Users Are Mapped: 65534  🡐 mapped to 65534
                   Superuser Security Types: krb5    🡐 superuser via AUTH_RPCGSS only
               Honor SetUID Bits in SETATTR: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_allow_krb5_only

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol

[root@nfsclient /]# mount -t nfs4 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_krb5_only_notkrb5

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nobody nobody      0 Apr 24  2013 root_allow_krb5_only_notkrb5

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 99 99      0 Apr 24  2013 root_allow_krb5_only_notkrb5


NOTE: Again, the UID of an unmapped user in NFSv4 is 99. This is controlled via /etc/idmapd.conf
in Linux and /etc/default/nfs in Solaris.

[root@nfsclient /]# mount -t nfs4 -o sec=krb5 cluster:/nfsvol /mnt
[root@nfsclient /]# kinit
Password for root@KRB5DOMAIN.NETAPP.COM:
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_krb5_only_krb5mount

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 root   daemon      0 Apr 24  2013 root_allow_krb5_only_krb5mount

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  1      0 Apr 24  2013 root_allow_krb5_only_krb5mount
```

Example 3: Root and all anonymous users are allowed access as root using `anon=0`.

This example allows only for `sec=sys` and `sec=krb5` over NFSv4.

```
cluster::> vserver export-policy rule show –policyname root_allow_anon0 -instance
```

```
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_allow_anon0
                                 Rule Index: 1
                            Access Protocol: nfs4    ← only NFSv4 is allowed
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   ← all clients
                             RO Access Rule: krb5, sys      ← AUTH_SYS and AUTH_RPCGSS allowed
                             RW Access Rule: krb5, sys      ← AUTH_SYS and AUTH_RPCGSS allowed
User ID To Which Anonymous Users Are Mapped: 0       ← mapped to 0
                   Superuser Security Types: none    ← superuser (root) squashed to anon user
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true


cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_allow_anon0

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol

[root@nfsclient /]# mount -t nfs4 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_anon0

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 root   daemon      0 Apr 24  2013 root_allow_anon0

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  1      0 Apr 24  2013 root_allow_anon0


[root@nfsclient /]# mount -t nfs4 -o sec=krb5 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_anon0_krb5

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 root   daemon      0 Apr 24  2013 root_allow_anon0_krb5

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  1      0 Apr 24  2013 root_allow_anon0_krb5
```

Example of Creating Load Sharing Mirrors for vsroot

This example shows how to create LS mirrors for the vsroot volume. Use of LS mirrors applies to NFSv3 and SMB only.

```
cluster::*> vol show -vserver NFS -vsroot true -fields size
vserver volume size
------- ------ ----
NFS     vsroot 1GB
```

```
cluster::*> vol create -vserver NFS -volume vsroot_mirror1 -aggregate aggr1_node1 -size 1g -type
DP
[Job 26705] Job succeeded: Successful

cluster::*> vol create -vserver NFS -volume vsroot_mirror2 -aggregate aggr1_node2 -size 1g -type
DP
[Job 26707] Job succeeded: Successful

cluster::*> snapmirror create -source-path NFS:vsroot -destination-path NFS:vsroot_mirror1 -
vserver NFS -type LS -schedule daily
[Job 26709] Job succeeded: SnapMirror: done

cluster::*> snapmirror create -source-path NFS:vsroot -destination-path NFS:vsroot_mirror2 -
vserver NFS -type LS -schedule daily
[Job 26711] Job succeeded: SnapMirror: done

cluster::*> snapmirror show -source-path NFS:vsroot -fields schedule,state
source-path                  destination-path                    schedule state
---------------------------- ----------------------------------- -------- ------------
cluster://NFS/vsroot         cluster://NFS/vsroot_mirror1 daily    -       -
cluster://NFS/vsroot         cluster://NFS/vsroot_mirror2 daily    -       -
2 entries were displayed.

cluster::*> snapmirror initialize-ls-set -source-path NFS:vsroot
[Job 26714] Job is queued: snapmirror initialize-ls-set for source " cluster://NFS/vsroot".

cluster::*> snapmirror show -source-path NFS:vsroot -fields schedule,state
source-path                  destination-path                    schedule state
---------------------------- ----------------------------------- -------- ------------
cluster://NFS/vsroot         cluster://NFS/vsroot_mirror1         daily    Snapmirrored
cluster://NFS/vsroot         cluster://NFS/vsroot_mirror2         daily    Snapmirrored
2 entries were displayed.
```

Export Policy Rule Inheritance Example

In the following example, the SVM root volume has limited superuser access only to the client `x.x.x.x`. When the root user attempts to access a mount from a client other than `x.x.x.x`, it squashes to the anon user, which is 65534:

```
cluster::> vol show -vserver nfs_svm -volume rootvol -fields policy
  (volume show)
vserver volume  policy
------- ------- -------
nfs_svm rootvol default

cluster::> export-policy rule show -vserver nfs_svm -policyname default -instance
  (vserver export-policy rule show)

                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 2
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: none
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true
2 entries were displayed.
```

As per the example in the section “Limiting Access to the SVM Root Volume,” root would not be able to list the contents of the SVM root based on the volume permissions (711) and the existing export policy rules on any hosts other than `x.x.x.x`.

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# mount | grep mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /mnt
# ls
ls: cannot open directory .: Permission denied
```

If the data volumes in the SVM also are set to this export policy, they use the same rules, and only the client set to have root access is able to log in as root.

If root access is desired to the data volumes, then a new export policy can be created and root access can be specified for all hosts or a subset of hosts through subnet, netgroup, or multiple rules with individual client IP addresses or host names.

The same concept applies to the other export policy rule attributes, such as RW.

For example, if the default export policy rule is changed to disallow write access to all clients except `x.x.x.x` and to allow superuser, then even root is disallowed write access to volumes with that export policy applied.

```
cluster::> export-policy rule modify -vserver nfs_svm -policyname default -ruleindex 2 -rwrule
never -superuser any

cluster::> export-policy rule show -vserver nfs_svm -policyname default -instance
  (vserver export-policy rule show)

                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 2
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: any
                             RW Access Rule: never
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true
2 entries were displayed.

# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
```

```
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# mount | grep mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /mnt
# touch rootfile
touch: cannot touch `rootfile': Read-only file system
```

When a new policy and rule are created and applied to the data volume, the same user is allowed to write to the data volume mounted below the SVM root volume. This is the case despite the export policy rule at the SVM root volume disallowing write access.

For example:

```
cluster::> export-policy create -vserver nfs_svm -policyname volume
cluster::> export-policy rule create -vserver nfs_svm -policyname volume -clientmatch 0.0.0.0/0 -
rorule any -rwrule any -allow-suid true -allow-dev true -ntfs-unix-security-ops fail -chown-mode
restricted -superuser any -protocol any -ruleindex 1 -anon 65534

cluster::> export-policy rule show -vserver nfs_svm -policyname volume -instance
  (vserver export-policy rule show)

                                    Vserver: nfs_svm
                                Policy Name: volume
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

::> volume modify -vserver flexvol -volume unix -policy volume
```

From the client:

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# cd /mnt/unix
[root@linux-client unix]# ls
file
[root@linux-client unix]# touch rootfile
[root@linux-client unix]# ls -la | grep rootfile
-rw-r--r--. 1 root root     0 Apr  1  2014 rootfile
# cd ..
# ls
nfs4  ntfs  unix
# touch rootdir
touch: cannot touch `rootdir': Read-only file system
```

However, the read-only attribute for the export policy rules needs to allow read access from the parent to allow mounts to occur. Setting `rorule` to `never` or not setting an export policy rule in the parent volume’s export policy (empty policy) disallows mounts to volumes underneath that parent.

In the following example, the vsroot volume has an export policy that has `rorule` and `rwrule` set to `never`, while the data volume has an export policy with a rule that is wide open.

```
cluster::> export-policy rule show -vserver nfs -policyname wideopen -instance
  (vserver export-policy rule show)

                                    Vserver: nfs
                                Policy Name: wideopen
```

```
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: any
                             RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 0
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> export-policy rule show -vserver nfs -policyname deny -instance
  (vserver export-policy rule show)

                                    Vserver: nfs
                                Policy Name: deny
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: never
                             RW Access Rule: never
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: sys
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> volume show -vserver nfs -volume rootvol -fields policy,unix-permissions
vserver volume  policy unix-permissions
------- ------- ------ ----------------
nfs     rootvol deny   ---rwx--x—x

cluster::> volume show -vserver nfs -volume unix -fields policy,unix-permissions
vserver volume policy   unix-permissions
------- ------ -------- ----------------
nfs     unix   wideopen ---rwxrwxrwx
```

When a mount of the volume UNIX is attempted, access is denied.

```
# mount -o nfsvers=3 x.x.x.e:/unix /cdot
mount.nfs: access denied by server while mounting x.x.x.e:/unix
```

When the `deny` policy is changed to allow read-only access, mounting is allowed.

```
cluster::> export-policy rule modify -vserver nfs -policyname deny -rorule any -ruleindex 1

# mount -o nfsvers=3 x.x.x.e:/unix /cdot
# mount | grep unix
x.x.x.e:/unix on /cdot type nfs (rw,nfsvers=3,addr=x.x.x.e)
```

As a result, storage administrators can have complete and granular control over what users see and access file systems using export policies, rules, and volume permissions.

Export Policy Rule Index Example

In the following example, a client with the IP address of `x.x.x.x` (host name of `centos64`) has been denied access to read a volume while all other clients are allowed access. However, the client rule is below the `all access` rule, so mount and read are allowed.

For example:

```
cluster::> export-policy rule show -vserver NAS -policyname allow_all
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
NAS          allow_all       1       any      0.0.0.0/0             any
NAS          allow_all       99      any      x.x.x.x        never
2 entries were displayed.

cluster::> vol show -vserver NAS -volume unix -fields policy
```

```
vserver volume policy
------- ------ ---------
NAS     unix   allow_all

[root@centos64 ~]# mount -o nfsvers=3 x.x.x.a:/vol/nfs /7mode
[root@centos64 ~]# mount x.x.x.a:/unix /mnt
[root@centos64 ~]# cd /mnt
[root@centos64 mnt]# ls -la
total 12
drwxrwxrwx.  3 root root    4096 Dec 10 14:49 .
dr-xr-xr-x. 46 root root    4096 Dec 10 14:57 ..
drwxrwx---.  2 root root 4096 Dec 10 15:00 file
```

If those rules are flipped, the client is denied access despite the rule allowing access to everyone being in the policy. Rule index numbers can be modified with the `export-policy rule setindex` command. In the following example, rule #1 has been changed to rule #99. Rule #99 gets moved to #98 by default.

```
cluster::> export-policy rule setindex -vserver NAS -policyname allow_all -ruleindex 1 -
newruleindex 99

cluster::> export-policy rule show -vserver NAS -policyname allow_all
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
NAS          allow_all       98      any      x.x.x.x        never
NAS          allow_all       99      any      0.0.0.0/0             any
2 entries were displayed.

cluster::> export-policy cache flush -vserver NAS -cache all

Warning: You are about to flush the "all (but showmount)" cache for Vserver "NAS" on node
"node2", which will result in increased traffic to the name servers. Do you want to proceed with
         flushing the cache? {y|n}: y


[root@centos64 /]# mount x.x.x.a:/unix /mnt
mount.nfs: access denied by server while mounting x.x.x.a:/unix
```

NFSv4.x ACL Explicit DENY Example

For example, the user `ldapuser` belongs to the group Domain Users.

```
sh-4.1$ id
uid=55(ldapuser) gid=513(Domain Users) groups=513(Domain Users),503(unixadmins)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Permissions on the volume `mixed` are 775. The owner is root and the group is Domain Users.

```
[root@nfsclient /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rwaDxtTnNcy
D:g:GROUP@:C
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

[root@nfsclient /]# ls -la | grep mixed
drwxrwxr-x.   3 root     Domain Users   4096 Apr 30 09:52 mixed
```

Because `ldapuser` is a member of Domain Users, it should have write access to the volume, and it does.

```
sh-4.1$ cd /mixed
sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root Domain Users 4096 Apr 30 09:52 .
dr-xr-xr-x. 28 root root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root root         4096 Apr 30 08:00 .snapshot
```

```
sh-4.1$ touch newfile
sh-4.1$ nfs4_getfacl /mixed
sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 09:56 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile
```

However, if the ACLs are reordered and the explicit DENY for EVERYONE is placed ahead of group, then `ldapuser` is denied access to write to the same volume it just had access to write to:

```
[root@nfsclient /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC
A:g:GROUP@:rwaDxtTnNcy

[root@nfsclient /]# su ldapuser
sh-4.1$ cd /mixed
sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 09:56 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile

sh-4.1$ touch newfile2
touch: cannot touch `newfile2': Permission denied
```

If the explicit DENY rule is removed, the desired access is restored:

```
[root@nfsclient /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A::EVERYONE@:rxtncy
A:g:GROUP@:rwaDxtTnNcy

[root@nfsclient /]# su ldapuser

sh-4.1$ cd /mixed

sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 09:56 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile

sh-4.1$ touch newfile2

sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 10:06 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 10:06 newfile2
```

NFSv4.x ACL Preservation Example

This is a newly created UNIX-style volume.

```
cluster::> volume show -vserver vs0 -volume unix -fields security-style,
unix-permissions,user,group
vserver volume user group security-style unix-permissions
------- ------ ---- ----- -------------- ----------------
vs0     unix   0    1     unix           ---rwxr-xr-x
```

```
cluster::> vserver security file-directory show -vserver vs0 -path /unix

                Vserver: vs0
              File Path: /unix
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
         Unix Mode Bits: 755
 Unix Mode Bits in Text: rwxr-xr-x
                   ACLs: -
```

In the preceding example, the volume (`/unix`) has 755 permissions. That means that the owner has ALL access, the owning group has READ/EXECUTE access, and everyone else has READ/EXECUTE access.

Even though there are no NFSv4 ACLs in the fsecurity output, there are default values set that can be viewed from the client:

```
[root@nfsclient /]# mount -t nfs4 krbsn:/unix /unix
[root@nfsclient /]# ls -la | grep unix
drwxr-xr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy
```

The NFSv4 ACLs earlier show the same: the owner has ALL access, the owning group has READ/EXECUTE access, and everyone else has READ/EXECUTE access. The default mode bits are tied to the NFSv4 ACLs.

When mode bits are changed, the NFSv4 ACLs are also changed:

```
[root@nfsclient /]# chmod 775 /unix
[root@nfsclient /]# ls -la | grep unix
drwxrwxr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rwaDxtTnNcy
A::EVERYONE@:rxtncy
```

NFS Audit Event Example

The following example is from an audit event using NFSv3:

```
- <Event>
- <System>
  <Provider Name="Netapp-Security-Auditing" />
  <EventID>4663</EventID>
  <EventName>Get Object Attributes</EventName>
  <Version>1</Version>
  <Source>NFSv3</Source>
  <Level>0</Level>
  <Opcode>0</Opcode>
  <Keywords>0x8020000000000000</Keywords>
  <Result>Audit Success</Result>
  <TimeCreated SystemTime="2013-08-08T20:36:05.011243000Z" />
  <Correlation />
  <Channel>Security</Channel>
  <Computer>e284de25-3edc-11e2-92d0-123478563412/525c9a2c-dce2-11e2-b94f-123478563412</Computer>
  <Security />
  </System>
- <EventData>
```

```
  <Data Name="SubjectIP" IPVersion="4">x.x.x.x</Data>
  <Data Name="SubjectUnix" Uid="10000" Gid="503" Local="false" />
  <Data Name="ObjectServer">Security</Data>
  <Data Name="ObjectType">Directory</Data>
  <Data Name="HandleID">00000000000453;00;00000040;3a2cada4</Data>
  <Data Name="ObjectName">/</Data>
  <Data Name="InformationRequested">File Type; File Size; Last Accessed Time; Last Metadata
Modfied Time; Last Modified Time; Unix Mode; Unix Owner; Unix Group;</Data>
  </EventData>
  </Event>
```

NFSv4.x Referral Example

For example:

The data volume lives on node1:

```
cluster::> volume show -vserver vs0 -volume nfsvol -fields node
vserver volume node
------- ------ -------------
vs0     nfsvol node1
```

The data LIF lives on node2:

```
cluster::> net int show -vserver vs0 -lif data2 -fields curr-node,home-node
  (network interface show)
vserver lif   home-node     curr-node     address
------- ----- ------------- ------------- -----------
vs0     data2 node2         node2         x.x.x.a
```

There is also a data LIF on node1:

```
cluster::> net int show -vserver vs0 -curr-node node1 -role data
  (network interface show)
            Logical    Status     Network            Current       Current Is
Vserver     Interface  Admin/Oper Address/Mask       Node          Port    Home
----------- ---------- ---------- ------------------ ------------- ------- ----
vs0
            data1         up/up    x.x.x.a/24     node1        e0a     true
```

The client makes a mount request to the data LIF on node2, at the IP address x.x.x.a:

```
[root@nfsclient /]# mount -t nfs4 x.x.x.a:/nfsvol /mnt
```

The mount location looks to be at the IP address specified by the client:

```
[root@nfsclient /]# mount | grep /mnt
x.x.x.a:/nfsvol on /mnt type nfs4 (rw,addr=x.x.x.a,clientaddr=x.x.x.z)
```

The data volume lives on node1:

But the cluster shows that the connection was actually established to node1, where the data volume lives. No connection was made to node2:

```
cluster::> network connections active show -node node1 -service nfs*
              Vserver   Interface         Remote
      CID Ctx Name      Name:Local Port   Host:Port            Protocol/Service
--------- --- --------- ----------------- -------------------- ----------------
Node: node1
286571835   6 vs0          data:2049         x.x.x.z:763    TCP/nfs

cluster::> network connections active show -node node2 -service nfs*
There are no entries matching your query.
```

Because clients might become “confused” about which IP address they are actually connected to as per the `mount` command, Lenovo recommends using host names in mount operations.

When a user ACE is added to the ACL, the entry is reflected in the ACL on the appliance. In addition, the entire ACL is now populated. Note that the ACL is in SID format.

```
[root@nfsclient /]# nfs4_setfacl -a A::ldapuser@nfsv4domain.lenovo.com:ratTnNcCy /unix
[root@nfsclient /]# nfs4_getfacl /unix
A::ldapuser@nfsv4domain.lenovo.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rwaDxtTnNcy
A::EVERYONE@:rxtncy

cluster::> vserver security file-directory show -vserver vs0 -path /unix

                Vserver: vs0
              File Path: /unix
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
         Unix Mode Bits: 775
 Unix Mode Bits in Text: rwxrwxr-x
                   ACLs: NFSV4 Security Descriptor
                         Control:0x8014
                         DACL - ACEs
                           ALLOW-S-1-8-55-0x16019d
                           ALLOW-S-1-520-0-0x1601ff
                           ALLOW-S-1-520-1-0x1201ff-IG
                           ALLOW-S-1-520-2-0x1200a9
```

To see the translated ACLs, use fsecurity from the node shell on the node that owns the volume:

```
cluster::> node run -node node2 fsecurity show /vol/unix

[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0
    gid: 1
    mode: 0775 (rwxrwxr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55 - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Allow - GROUP@ - 0x001201ff
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
    SACL:
      No entries.
```

When a change is made to the mode bit when NFSv4 ACLs are present, the NFSv4 ACL that was just set is wiped by default.

```
[root@nfsclient /]# chmod 755 /unix
[root@nfsclient /]# ls -la | grep unix
drwxr-xr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy

cluster::> node run -node node2 fsecurity show /vol/unix

[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0
    gid: 1
    mode: 0755 (rwxr-xr-x)

  No security descriptor available.
```

To control this behavior in clustered ONTAP, use the following `diag-level` option.

```
cluster::> set diag
cluster::*> nfs server modify -vserver vs0 -v4-acl-preserve [enabled|disabled]
```

After the option is enabled, the ACL stays intact when mode bits are set.

```
[root@nfsclient /]# nfs4_setfacl -a A::ldapuser@nfsv4domain.lenovo.com:ratTnNcCy /unix
[root@nfsclient /]# ls -la | grep unix
drwxr-xr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::ldapuser@nfsv4domain.lenovo.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy

cluster::> vserver security file-directory show -vserver vs0 -path /unix

                Vserver: vs0
              File Path: /unix
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
         Unix Mode Bits: 755
 Unix Mode Bits in Text: rwxr-xr-x
                   ACLs: NFSV4 Security Descriptor
                         Control:0x8014
                         DACL - ACEs
                           ALLOW-S-1-8-55-0x16019d
                           ALLOW-S-1-520-0-0x1601ff
                           ALLOW-S-1-520-1-0x1200a9-IG
                           ALLOW-S-1-520-2-0x1200a9

cluster::> node run -node node2 fsecurity show /vol/unix

[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0
    gid: 1
    mode: 0755 (rwxr-xr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55 - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Allow - GROUP@ - 0x001200a9 (Read and Execute)
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
    SACL:
      No entries.
```

Note: The ACL is still intact after mode bits are set.

```
[root@nfsclient /]# chmod 777 /unix
[root@nfsclient /]# ls -la | grep unix
drwxrwxrwx.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::ldapuser@win2k8.ngslabs.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rwaDxtTnNcy
A::EVERYONE@:rwaDxtTnNcy

cluster::> vserver security file-directory show -vserver vs0 -path /unix

                Vserver: vs0
              File Path: /unix
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
         Unix Mode Bits: 777
 Unix Mode Bits in Text: rwxrwxrwx
                   ACLs: NFSV4 Security Descriptor
                         Control:0x8014
                         DACL - ACEs
                           ALLOW-S-1-8-55-0x16019d
                           ALLOW-S-1-520-0-0x1601ff
                           ALLOW-S-1-520-1-0x1201ff-IG
                           ALLOW-S-1-520-2-0x1201ff

cm6080-rtp2::*> node run -node node2 fsecurity show /vol/unix

[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0
    gid: 1
    mode: 0777 (rwxrwxrwx)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55 - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Allow - GROUP@ - 0x001201ff
      Allow - EVERYONE@ - 0x001201ff
    SACL:
      No entries.
```

NFS Connected-Clients Sample Output

The following shows sample output of the NFS `connected-clients` command:

```
cluster::*> nfs connected-clients show -vserver DEMO

     Node: node1
  Vserver: DEMO
  Data-Ip: 10.x.x.a
Client-Ip       Volume-Name      Protocol Idle-Time     Local-Reqs Remote-Reqs
--------------- ---------------- -------- ------------- ---------- -----------
10.x.x.b        scripts          nfs3     1h 3m 46s     391        0
10.x.x.c        XCP_catalog      nfs3     17s           0          372
10.x.x.d        scripts          nfs3     17s           372        0
10.x.x.c        home             nfs4     12h 50m 28s   256        0
10.x.x.b        home             nfs4.1   9h 51m 48s    47         0
```

Performance Examples for Different TCP Max Transfer Window Sizes

The following examples include the following performance tests:

- High file count/small file (<1K)/high metadata workloads (multithreaded Python script; NFSv3 only):
  - This test simulates operations that can be seen in an EDA or scratch space workload.
- High file count/512K file size/high metadata workloads (multithreaded dd; NFSv3 and NFSv4.1):
  - 100% write, this test simulates operations that can be seen in an EDA or scratch space workload.
- Large file/sequential throughput workload (50/50 read/write; multithreaded dd; NFSv3 and NFSv4.1):
  - This test simulates the type of operations that can be seen in database or virtualization workloads.

Note: The large file dd test used a 4k block size for its operations. The 512K file dd test used 8K block size for its operations.

The scripts can be found here: https://github.com/whyistheinternetbroken/NetAppFlexGroup/.

The goal for these tests is not to show the maximum possible performance. To do that, we'd be using larger AFA systems, more nodes in the cluster, more clients, and so on.

Instead, there are two goals for these tests:

1. Show the effect of different wsize and rsize values on different workload types.
2. Show NFSv3 and NFSv4.1 performance comparisons with sequential I/O workloads.

Here's some information about the test environment:

- NFSv3 and NFSv4.1
- Clients:
  - Two RHEL 7.6 servers
  - 64GB RAM each
  - 24 CPU cores
- AFA Flash model – 2 nodes (ONTAP 9.7P3)
- FlexGroup volume (16 member volumes)

- 10GB network

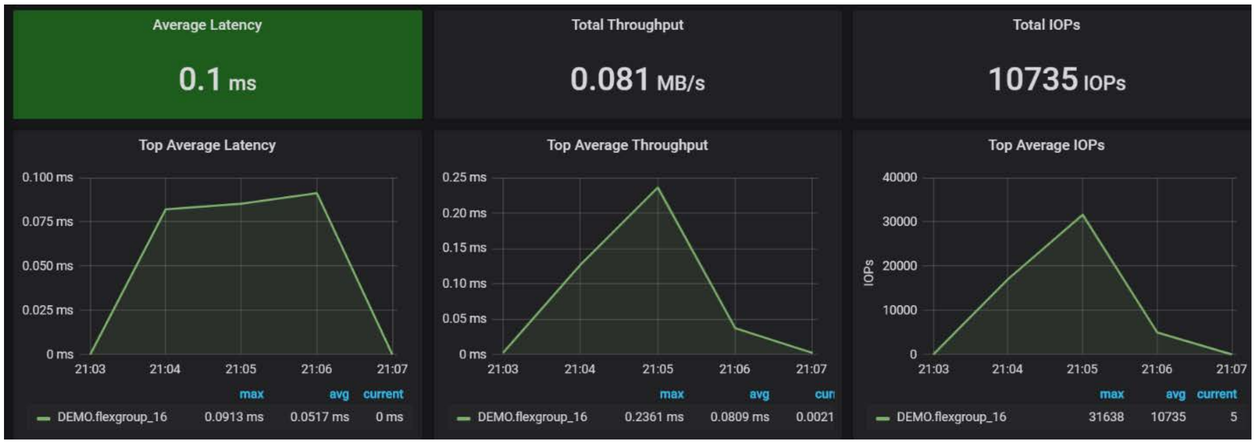
Note: This test is not intended to show the maximum performance of NFS or the systems being used, but to show comparisons between NFS versions and block sizes using tests on the same network and hardware.

Test #1: NFSv3 High File Count test – Many Folders and Files

This test creates ~1.1 million small files (less than 1K in size) and folders and uses multiple different TCP transfer size values with the mounts.

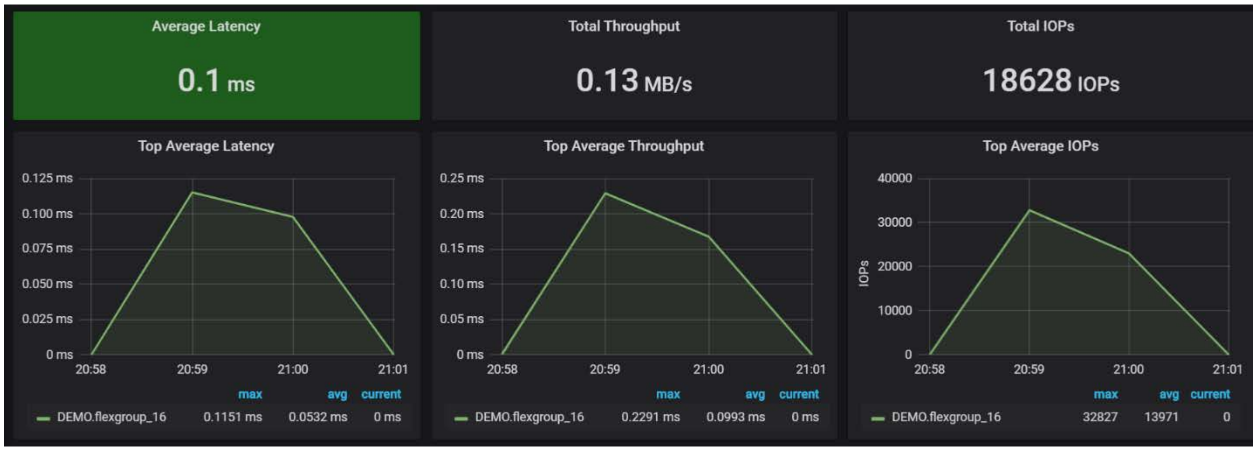
- 1MB wsize/rsize – Single Client, NFSv3
- **Completion time:** ~101.6 seconds
- **Average CPU %:** ~19%

Figure 23) High file count performance stats, 1MB wsize/rsize.



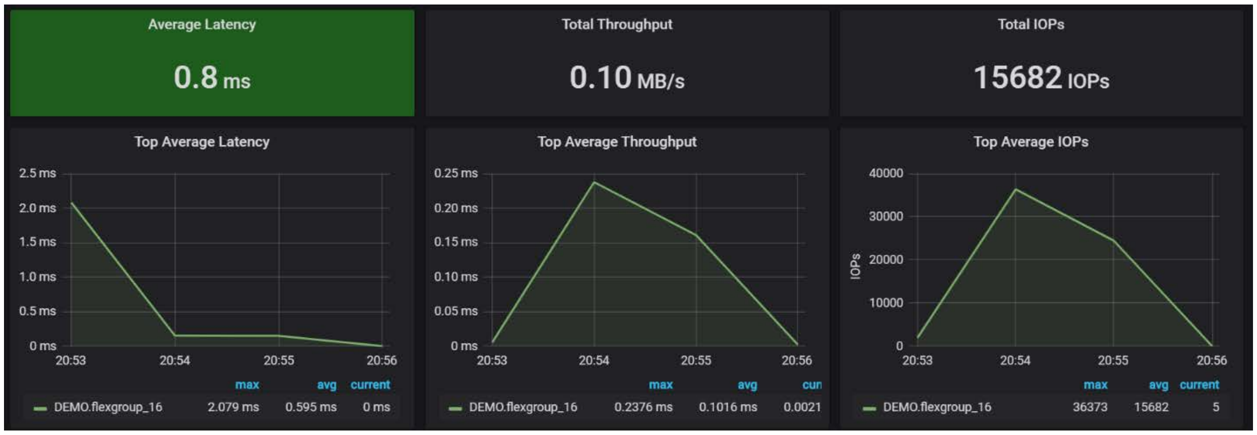
- 256K wsize/rsize – Single Client, NFSv3
- **Completion time:** ~101 seconds
- **Average CPU %:** ~23%

Figure 24) High file count performance stats, 256K wsize/rsize.



- 64K wsize/rsize – Single Client, NFSv3
- **Completion time:** ~102 seconds
- **Average CPU %:** ~24%

Figure 25) High file count performance stats, 64K wsize/rsize.

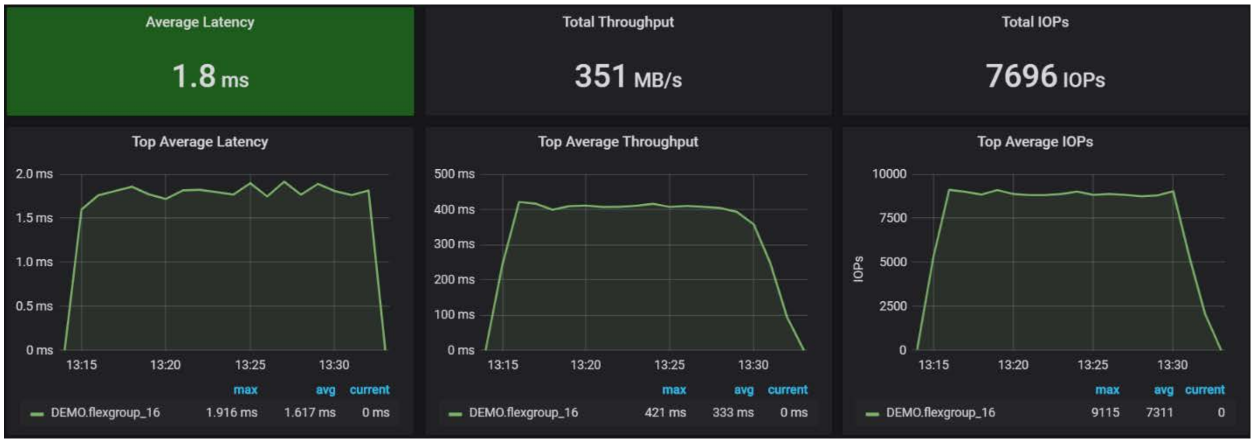


Test #2: High File Count Test – Many 512K Files, Two Folders Folder (NFSv3 and NFSv4.1)

These tests use multithreaded dd to create 800,000 small files (512K in size) across 8 directories (100,000 files per folder), with two clients, and uses multiple different TCP transfer size values with the mounts across NFSv3 and NFSv4.1.

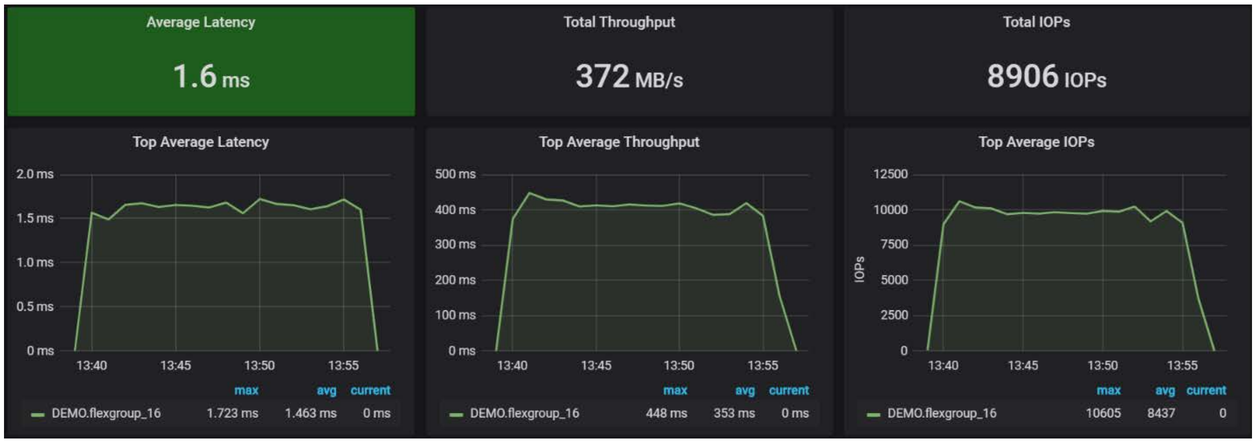
- 1MB wsize/rsize – 800,000 files, NFSv3
- **Completion time:** ~17 minutes, 29 seconds
- **Average CPU %:** ~32%

Figure 26) High file count performance NFSv3 – 800,000 files, 1MB wsize/rsize.



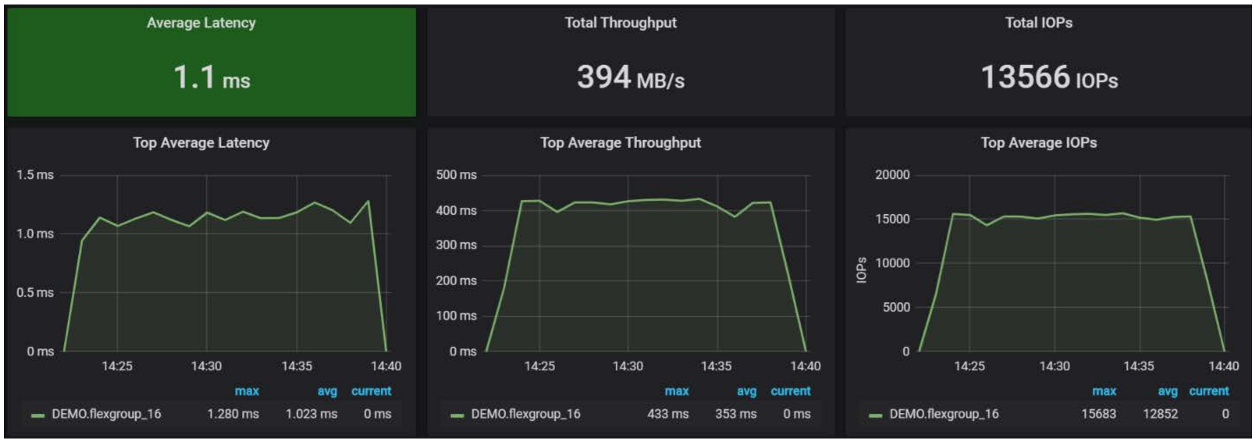
- 256K wsize/rsize – 800,000 files, NFSv3
- **Completion time:** ~16 minutes, 34 seconds
- **Average CPU %:** ~34.5%

Figure 27) High file count performance NFSv3 – 800,000 files/single folder, 256K wsize/rsize.



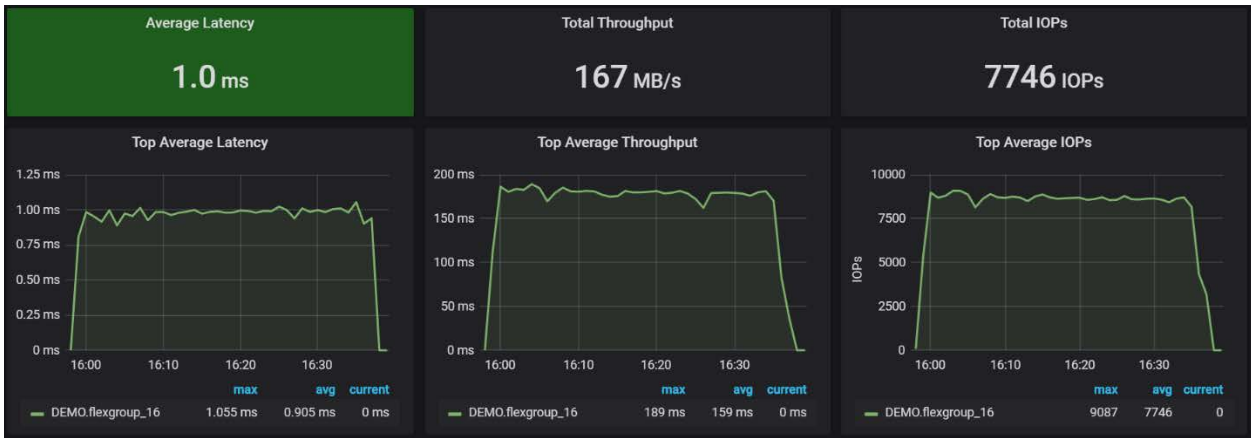
- 64K wsize/rsize – 800,000 files, NFSv3
- **Completion time:** ~16 minutes, 11 seconds
- **Average CPU %:** ~39%

Figure 28) High file count performance NFSv3 – 800,000 files/single folder, 64K wsize/rsize.



- 1MB wsize/rsize – 800,000 files, NFSv4.1
- **Completion time:** ~38 minutes, 20 seconds
- **Average CPU %:** ~26%

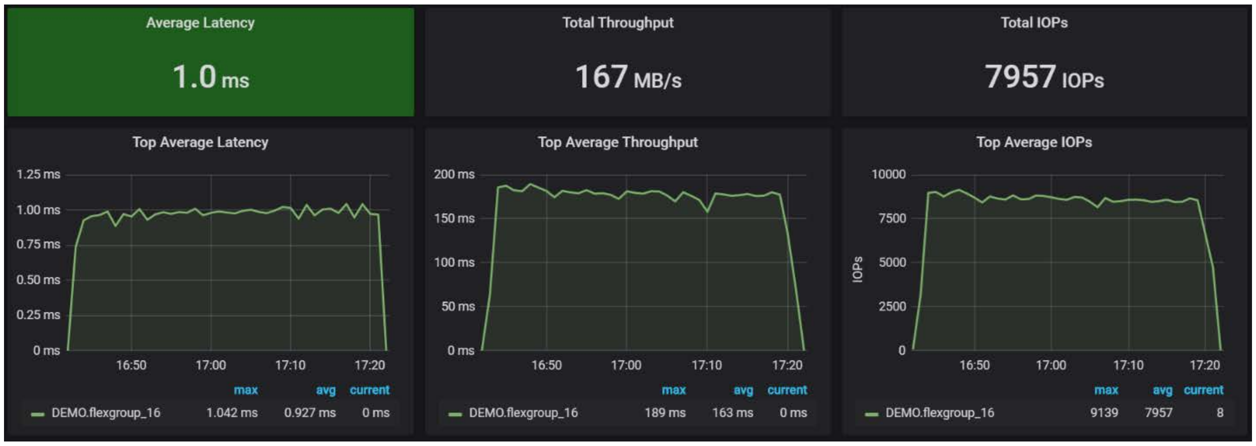
Figure 29) High file count performance NFSv4.1 – 800,000 files/single folder, 1MB wsize/rsize.



	max	avg	current
DEMO.flexgroup_16	1.055 ms	0.905 ms	0 ms
DEMO.flexgroup_16	189 ms	159 ms	0 ms
DEMO.flexgroup_16	9087	7746	0

- 256K wsize/rsize – 800,000 files, NFSv4.1
- **Completion time:** ~38 minutes, 15 seconds
- **Average CPU %:** ~27.5%

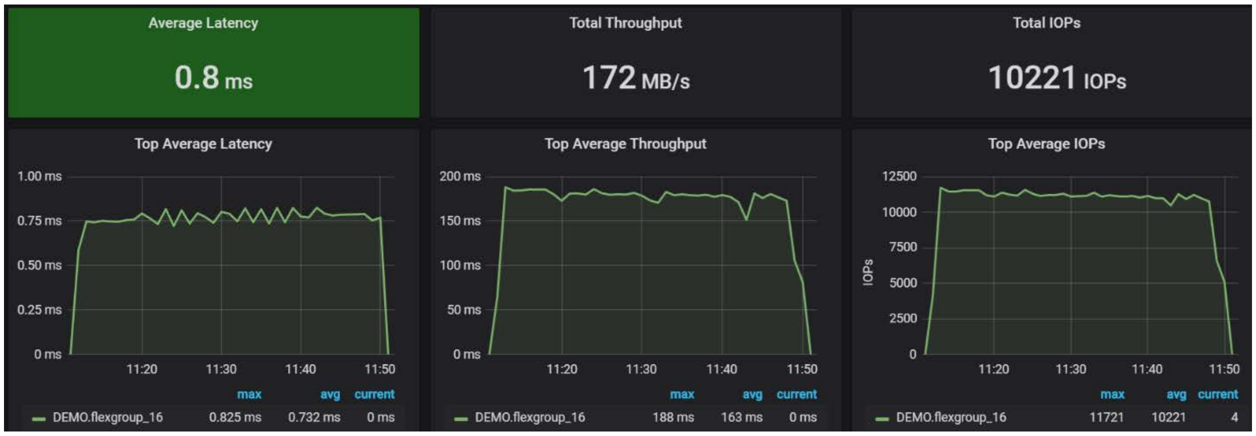
Figure 30) High file count performance NFSv4.1 – 800,000 files/single folder, 256K wsize/rsize.



	max	avg	current
DEMO.flexgroup_16	1.042 ms	0.927 ms	0 ms
DEMO.flexgroup_16	189 ms	163 ms	0 ms
DEMO.flexgroup_16	9139	7957	8

- 64K wsize/rsize – 800,000 files, NFSv4.1
- **Completion time:** ~38 minutes, 13 seconds
- **Average CPU %:** ~31%

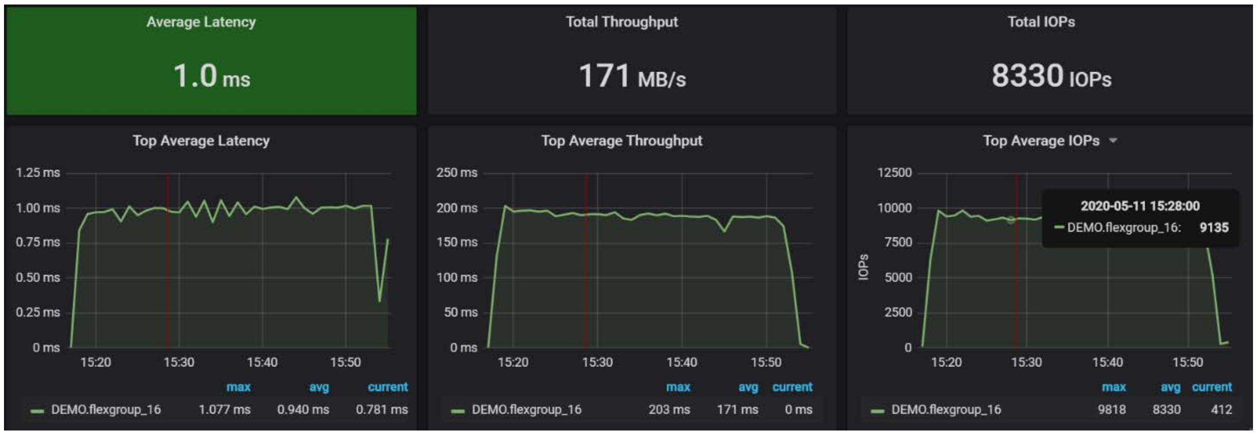
Figure 31) High file count performance NFSv4.1 – 800,000 files/single folder, 64K wsize/rsize.



	max	avg	current
DEMO.flexgroup_16	0.825 ms	0.732 ms	0 ms
DEMO.flexgroup_16	188 ms	163 ms	0 ms
DEMO.flexgroup_16	11721	10221	4

- 1MB wsize/rsize – 800,000 files, NFSv4.1 (pNFS)
- **Completion time:** ~35 minutes, 44 seconds
- **Average CPU %:** ~27%

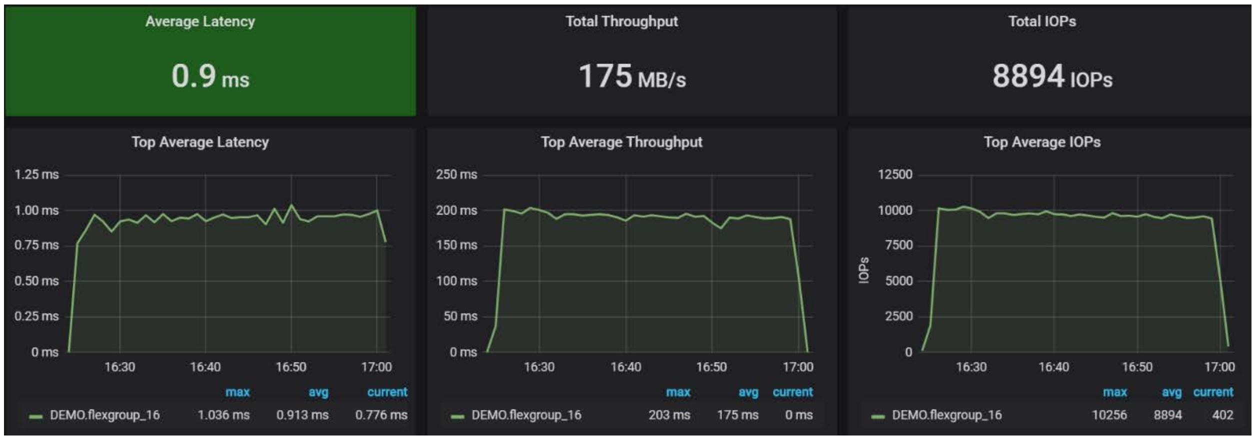
Figure 32) High file count performance NFSv4.1 (pNFS) – 800,000 files/single folder, 1MB wsize/rsize.



	max	avg	current
DEMO.flexgroup_16	1.077 ms	0.940 ms	0.781 ms
DEMO.flexgroup_16	203 ms	171 ms	0 ms
DEMO.flexgroup_16	9818	8330	412

- 256K wsize/rsize – 800,000 files, NFSv4.1 (pNFS)
- **Completion time:** ~35 minutes, 9 seconds
- **Average CPU %:** ~28.5%

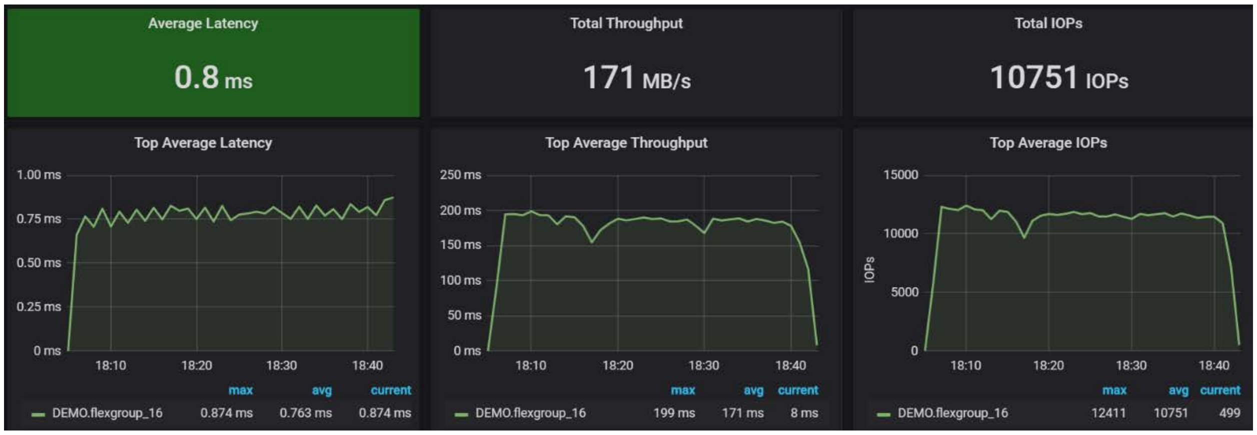
Figure 33) High file count performance NFSv4.1 (pNFS) – 800,000 files/single folder, 256K wsize/rsize.



	max	avg	current
DEMO.flexgroup_16	1.036 ms	0.913 ms	0.776 ms
DEMO.flexgroup_16	203 ms	175 ms	0 ms
DEMO.flexgroup_16	10256	8894	402

- 64K wsize/rsize – 800,000 files, NFSv4.1 (pNFS)
- **Completion time:** ~36 minutes, 41 seconds
- **Average CPU %:** ~33%

Figure 34) High file count performance NFSv4.1 (pNFS) – 800,000 files/single folder, 64K wsize/rsize.

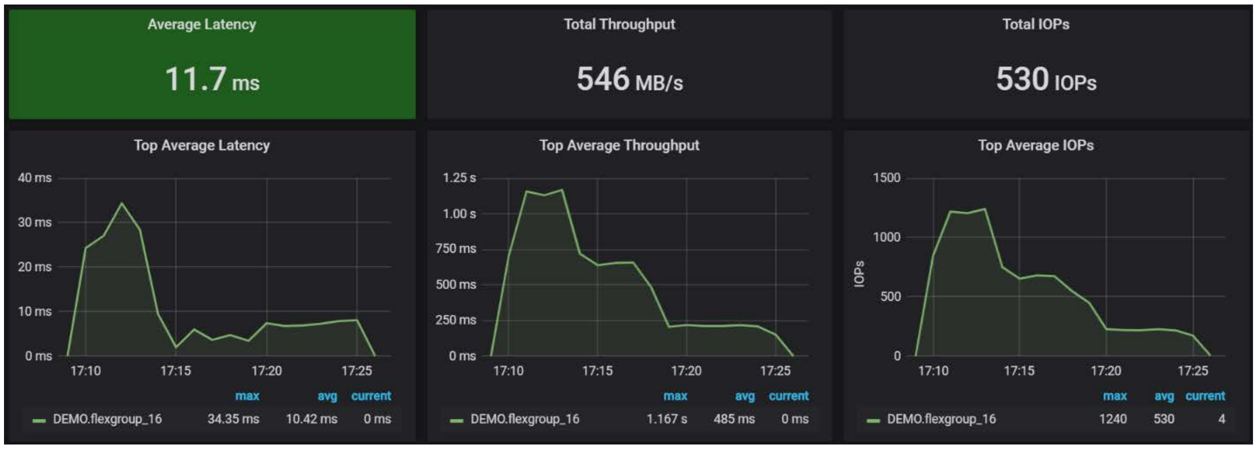


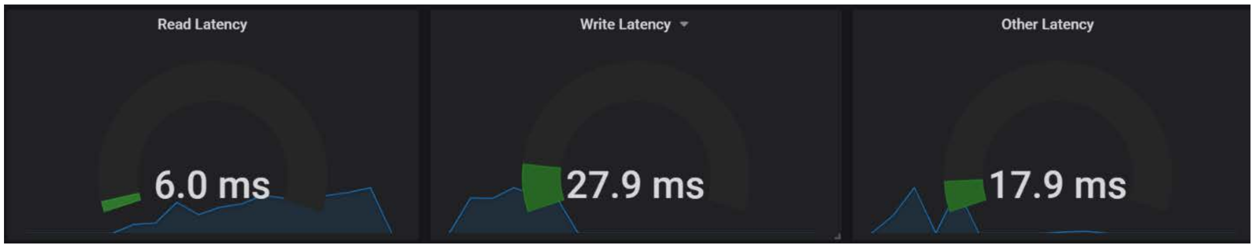
Test #3: Multithreaded dd Operations (NFSv3) – Multiple Clients

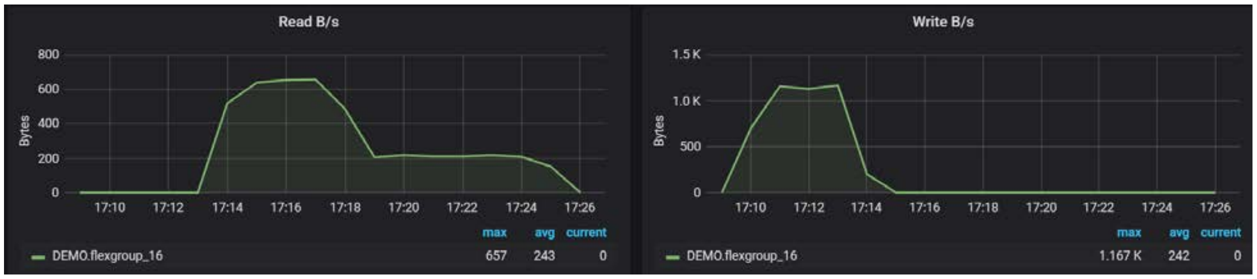
This test runs a multithreaded dd script that creates 8 folders with two 16GB files per folder. Two clients run the tests at the same time, splitting the workload to two different network ports. The tests use NFSv3 and NFSv4.1 (regular, delegations and pNFS) to measure time to completion and various other performance stats like throughput and NFS operations. The workload is a sequential 50/50 read/write workload.

- 1MB wsize/rsize – NFSv3
- **Completion time:** ~15 minutes, 23 seconds
- **Average maximum read throughput:** 654MBps
- **Average maximum write throughput:** 1.16GBps
- **Average CPU %:** ~19.5%

Figure 35) Parallel dd performance stats NFSv3, 1MB wsize/rsize.





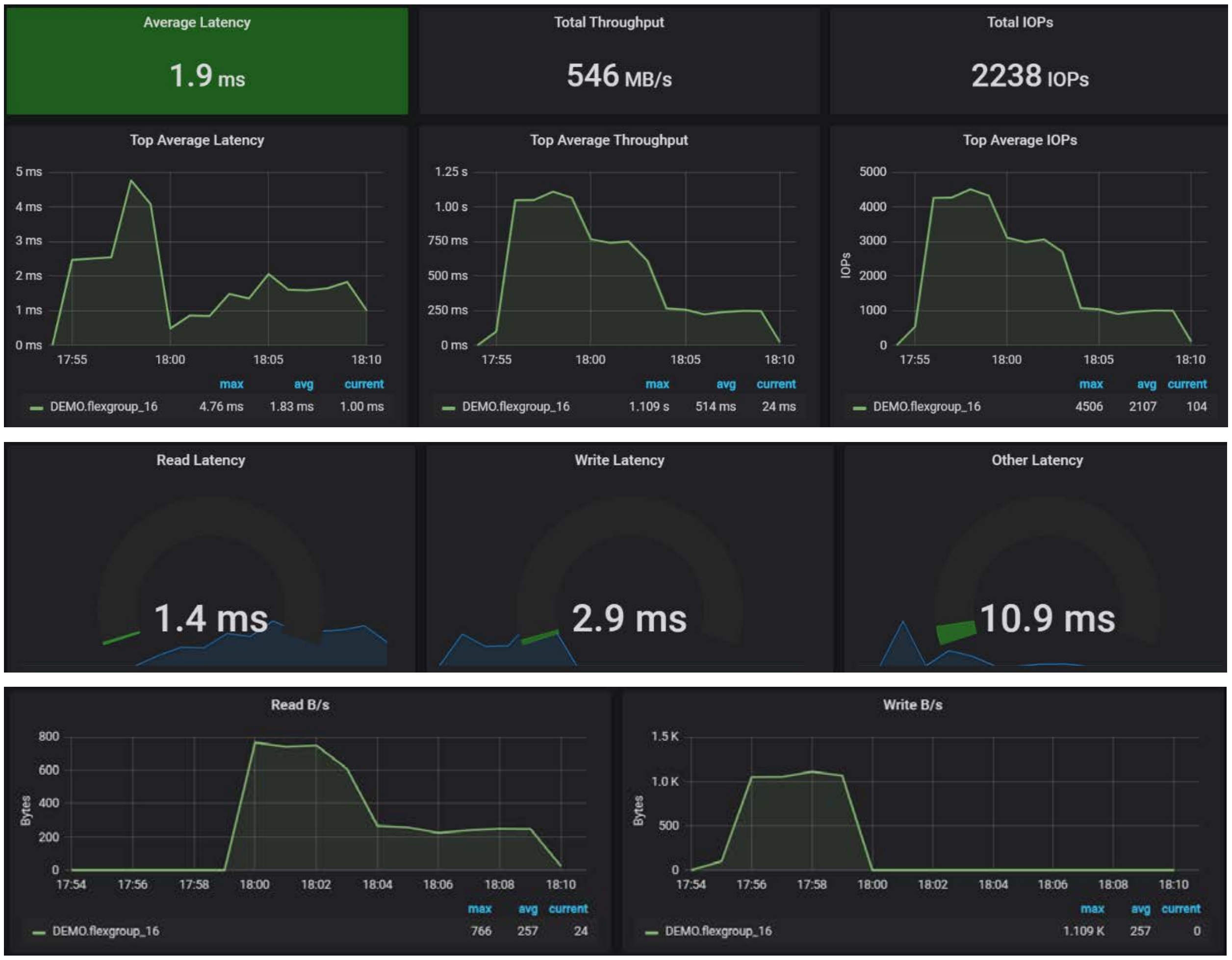


NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv3 -counter *size*

    Counter                                                      Value
    -------------------------------- --------------------------------
    nfsv3_read_size_histo                                            -
                               <=4KB                                 1
                              <=16KB                                16
                              <=64KB                                16
                             <=256KB                                16
                               <=1MB                            262144
    nfsv3_write_size_histo                                           -
                              <=512B                                52
                               <=1KB                                23
                               <=2KB                                54
                               <=4KB                               101
                               <=8KB                                29
                              <=16KB                                 1
                              <=64KB                                 6
                             <=128KB                                 6
                             <=256KB                                 4
                             <=512KB                                14
                               <=1MB                            262146
```

- 256K wsize/rsize – NFSv3
- **Completion time:** ~14 minutes, 17 seconds
- **Average maximum read throughput:** 766MBps
- **Average maximum write throughput:** 1.11GBps
- **Average CPU %:** ~19.5%

Figure 36) Parallel dd performance stats NFSv3, 256K wsize/rsize.



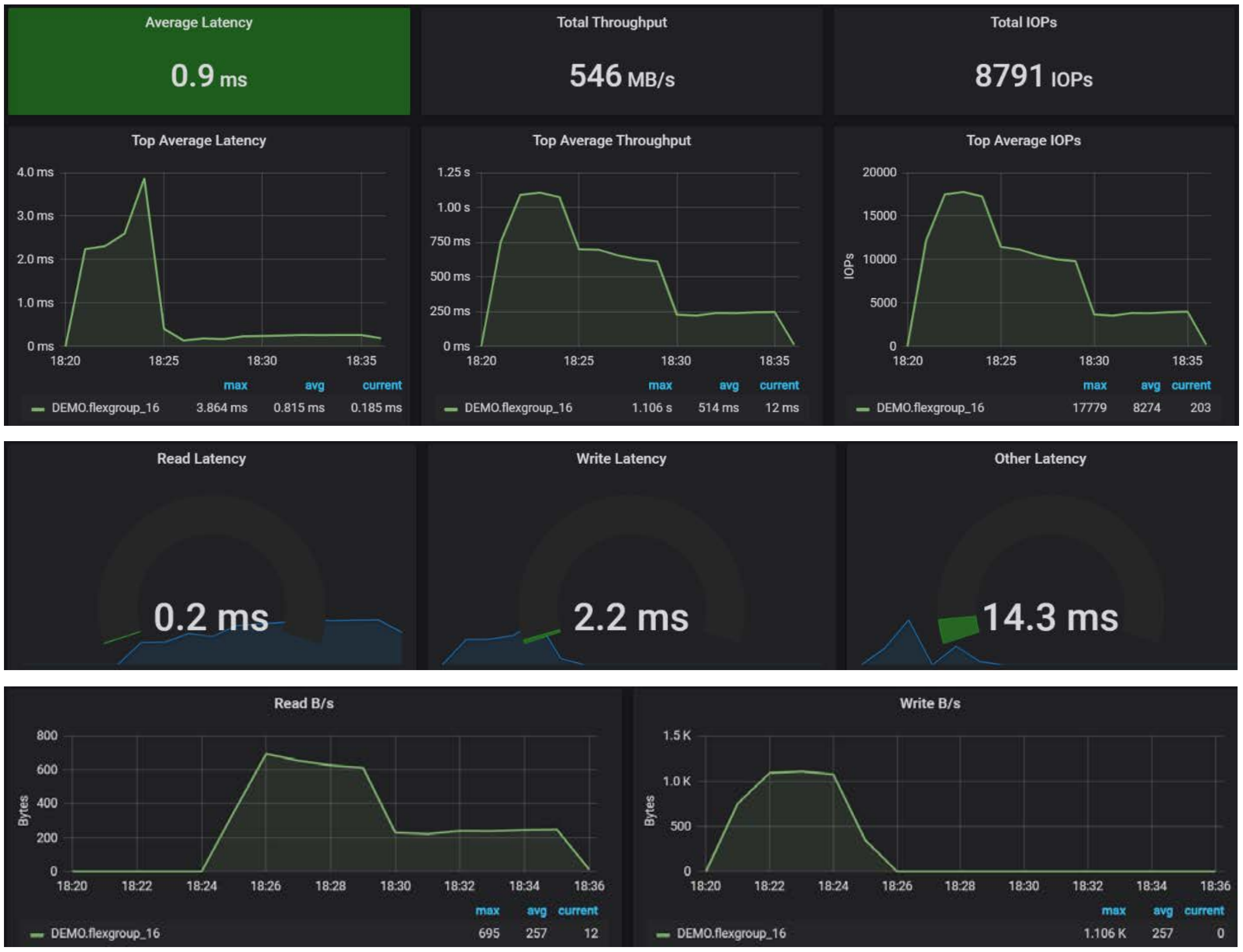
NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv3 -counter *size*

    Counter                                                          Value
    -------------------------------- --------------------------------
    nfsv3_read_size_histo                                                -
                               <=4KB                                     2
                              <=16KB                                    16
                              <=64KB                                    16
                             <=256KB                               1048576
    nfsv3_write_size_histo                                               -
                              <=512B                                    52
                               <=1KB                                    19
                               <=2KB                                    50
                               <=4KB                                    92
                               <=8KB                                    30
                              <=16KB                                     2
                              <=32KB                                     3
                              <=64KB                                     6
                             <=128KB                                    10
                             <=256KB                               1048579
```

- 64K wsize/rsize – NFSv3
- **Completion time:** ~14 minutes, 48 seconds
- **Average maximum read throughput:** 695MBps

- **Average maximum write throughput:** 1.11GBps
- **Average CPU %:** ~34.5%

Figure 37) Parallel dd performance stats NFSv3, 64K wsize/rsize.


Average Latency
0.9 ms
Total Throughput
546 MB/s
Total IOPs
8791 IOPs
Top Average Latency
DEMO.flexgroup_16 max 3.864 ms avg 0.815 ms current 0.185 ms
Top Average Throughput
DEMO.flexgroup_16 max 1.106 s avg 514 ms current 12 ms
Top Average IOPs
DEMO.flexgroup_16 max 17779 avg 8274 current 203
Read Latency
0.2 ms
Write Latency
2.2 ms
Other Latency
14.3 ms
Read B/s
DEMO.flexgroup_16 max 695 avg 257 current 12
Write B/s
DEMO.flexgroup_16 max 1.106 K avg 257 current 0

NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv3 -counter *size*

    Counter                                                          Value
    -------------------------------- --------------------------------
    nfsv3_read_size_histo                                                -
                               <=4KB                                     2
                              <=16KB                                    16
                              <=64KB                               4194304
    nfsv3_write_size_histo                                               -
                              <=512B                                    51
                               <=1KB                                    17
                               <=2KB                                    44
                               <=4KB                                    86
                               <=8KB                                    24
                              <=16KB                                     4
                              <=32KB                                    14
                              <=64KB                               4194303
```

Table 23) NFSv3 wsize/rsize performance summary.

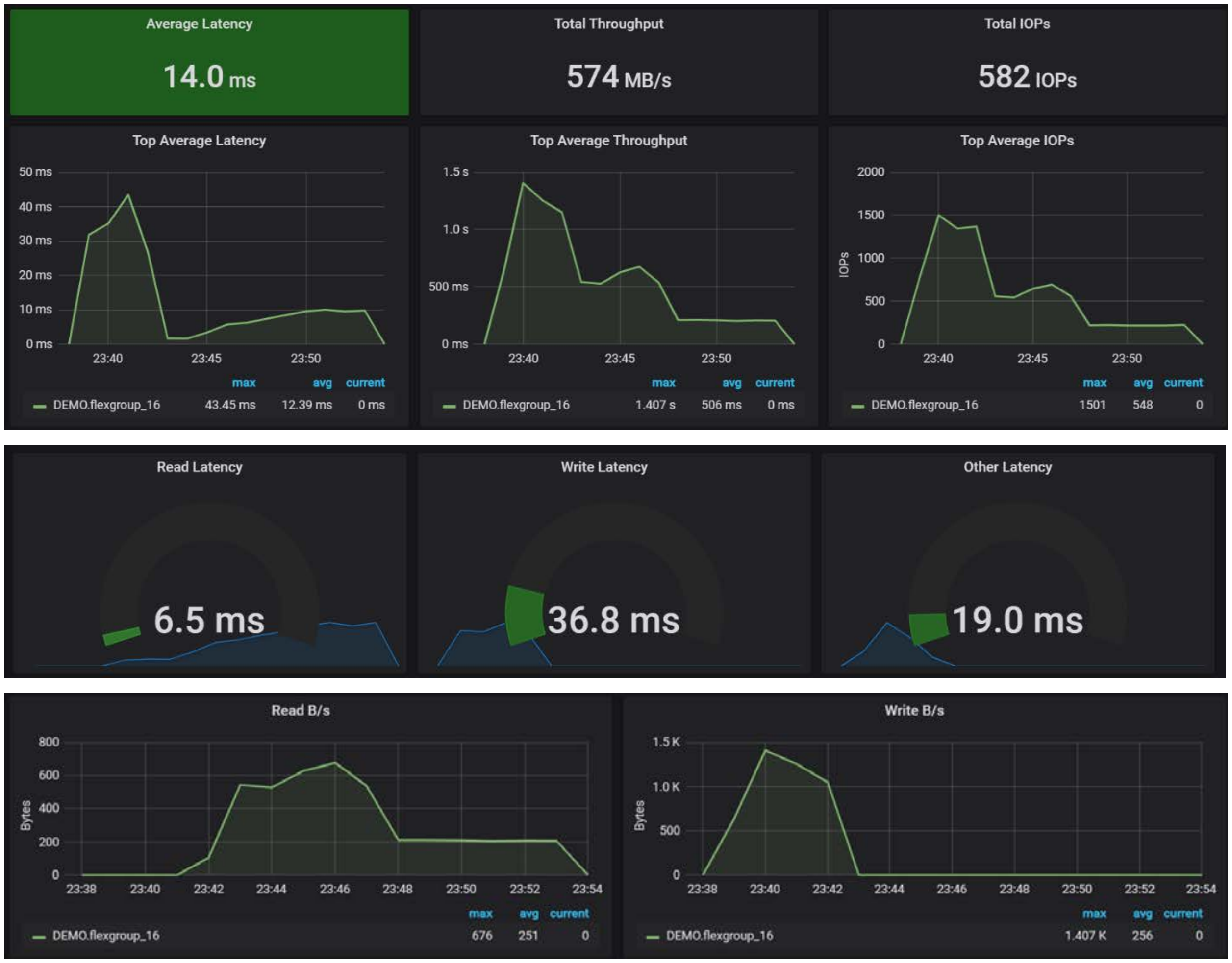
Test (rsize/wsize)	Average CPU %	Average Latency (ms)	Average IOPS	Completion Time
NFSv3 (1MB)	24%	11.7	530	15m23s
NFSv3 (256K)	27.5%	1.9	2108	14m17s
NFSv3 (64K)	34.5%	.9	8791	14m48s

Test #4: Multithreaded dd Operations (NFSv4.1) – Multiple Clients

This test runs a multithreaded dd script that creates 8 folders with two 16GB files per folder. Two clients run the tests at the same time, splitting the workload to two different network ports. The tests use NFSv4.1 to measure time to completion and various other performance stats like throughput and NFS operations. The workload is a sequential 50/50 read/write workload.

- 1MB wsize/rsize – NFSv4.1
- **Completion time:** ~15 minutes, 6 seconds
- **Average maximum read throughput:** 627MBps
- **Average maximum write throughput:** 1.4GBps
- **Average CPU %:** ~25%

Figure 38) Parallel dd performance stats NFSv4.1, 1MB wsize/rsize.

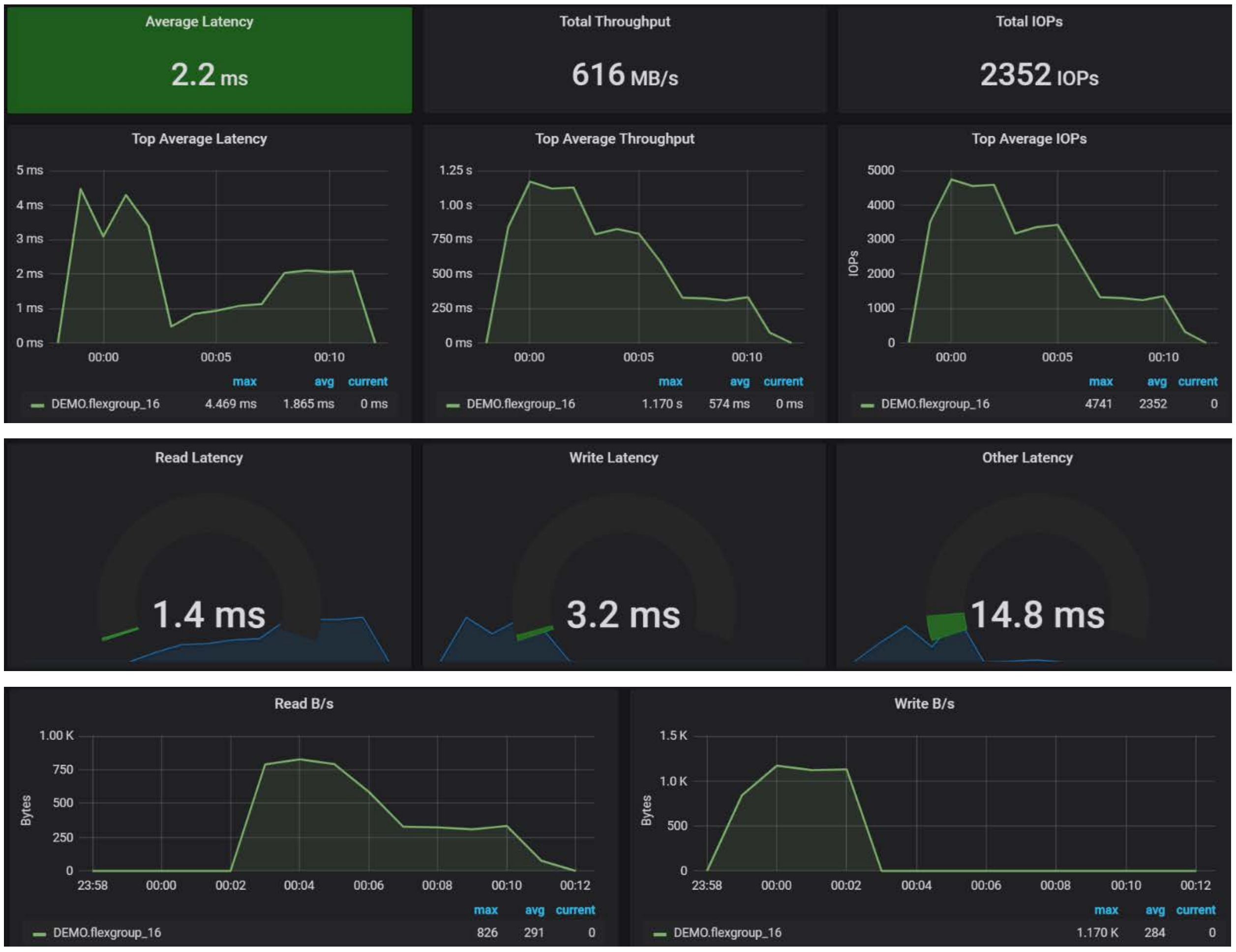


NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                            Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                                  -
                               <=16KB                                     16
                               <=64KB                                     16
                              <=256KB                                     16
                                <=1MB                                 262144
    nfs41_write_size_histo                                                 -
                               <=512B                                     48
                               <=32KB                                      1
                               <=64KB                                      2
                              <=128KB                                      6
                              <=256KB                                      9
                              <=512KB                                     13
                                <=1MB                                 262143
```

- 256K wsize/rsize – NFSv4.1
- **Completion time:** ~12 minutes, 10 seconds
- **Average maximum read throughput:** 712MBps
- **Average maximum write throughput:** 1.16GBps
- **Average CPU %:** ~19.5%

Figure 39) Parallel dd performance stats NFSv4.1, 256K wsize/rsize.

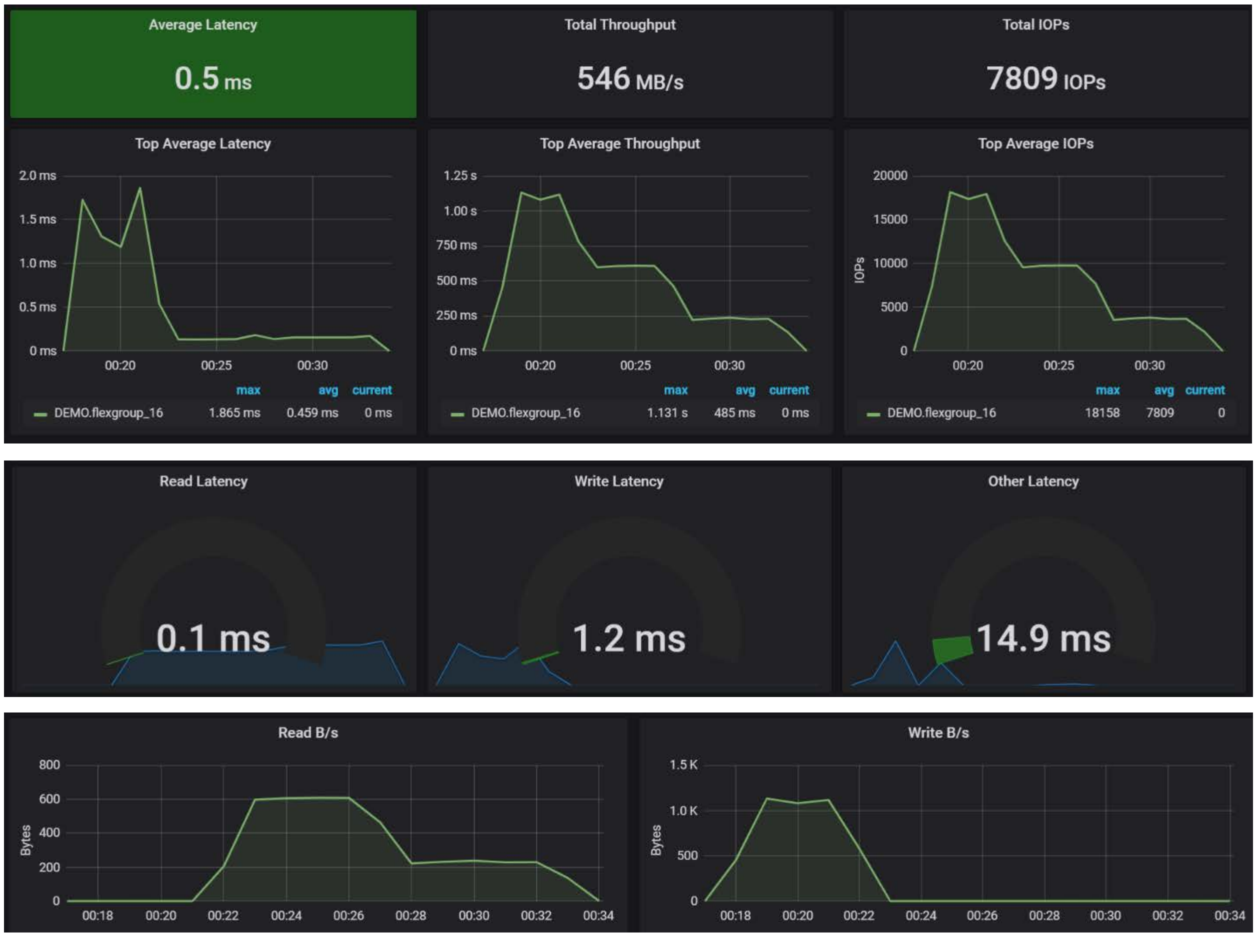


NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                          Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                                -
                              <=16KB                                    16
                              <=64KB                                    16
                             <=256KB                               1048576
    nfs41_write_size_histo                                               -
                              <=512B                                    48
                               <=8KB                                     1
                              <=16KB                                     3
                              <=32KB                                     3
                              <=64KB                                    11
                             <=128KB                                    13
                             <=256KB                               1048573
```

- 64K wsize/rsize – NFSv4.1
- **Completion time:** ~15 minutes, 8 seconds
- **Average maximum read throughput:** 606MBps
- **Average maximum write throughput:** 1.31GBps
- **Average CPU %:** ~32.5%

Figure 40) Parallel dd performance stats NFSv4.1, 64K wsize/rsize.



NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                    Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                          -
                              <=16KB                              16
                              <=64KB                         4194304
    nfs41_write_size_histo                                         -
                              <=512B                              48
                               <=4KB                               5
                               <=8KB                               3
                              <=16KB                               5
                              <=32KB                              15
                              <=64KB                         4194303
```

Table 24) NFSv4.1 wsize/rsize performance summary.

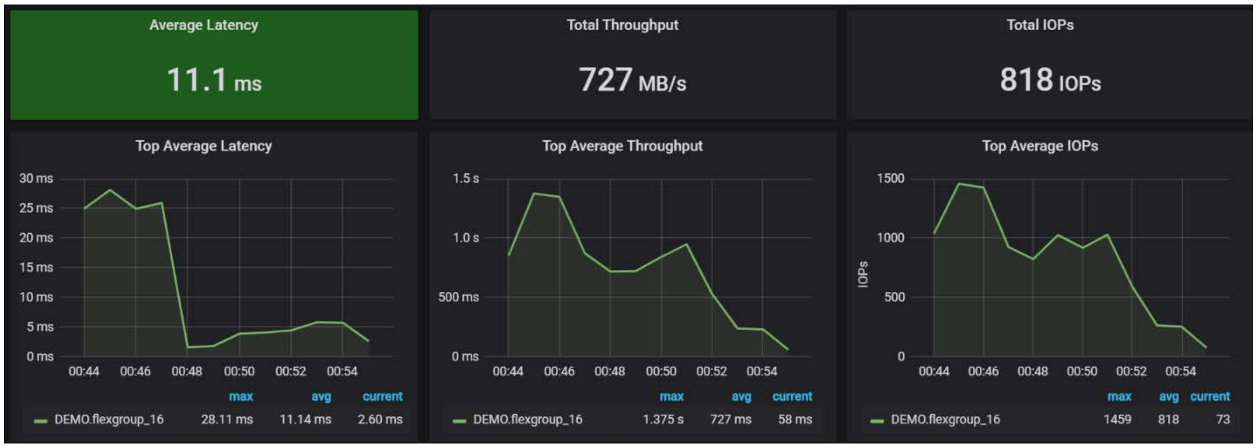
Test (rsize/wsize)	Average CPU %	Average Latency (ms)	Average IOPS	Completion Time
NFSv4.1 (1MB)	25%	14	582	15m8s
NFSv4.1 (256K)	19.5%	2.2	2352	12m10s
NFSv4.1 (64K)	32.5%	.5	7809	15m6s

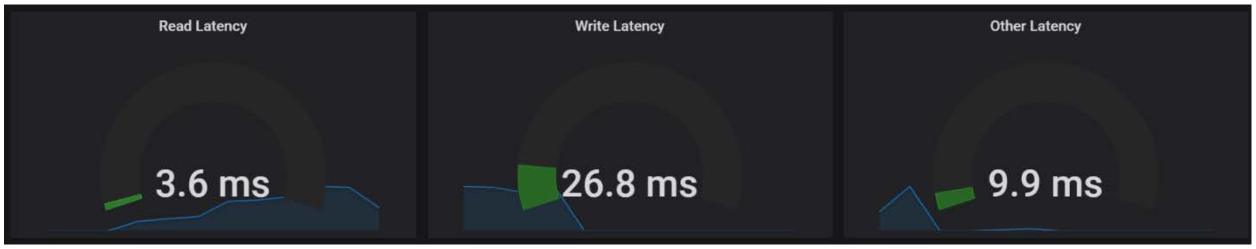
Test #5: Multithreaded dd Operations (NFSv4.1 pNFS) – Multiple Clients

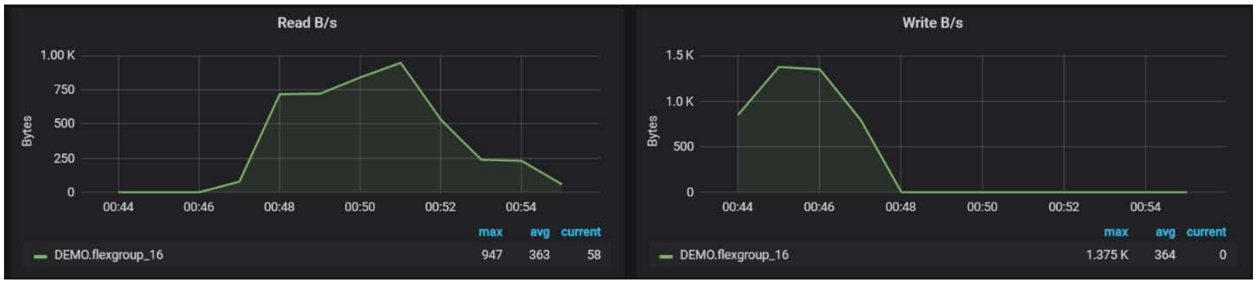
This test runs a multithreaded dd script that creates 8 folders with two 16GB files per folder. Two clients run the tests at the same time, splitting the workload to two different network ports. The tests use NFSv4.1 with pNFS (for information on pNFS, see the section “Parallel Network File System (pNFS)”) to measure time to completion and various other performance stats like throughput and NFS operations. The workload is a sequential 50/50 read/write workload.

- 1MB wsize/rsize – NFSv4.1 (pNFS)
- **Completion time:** ~10 minutes, 54 seconds
- **Average maximum read throughput:** 840MB/s
- **Average maximum write throughput:** 1.37GB/s
- **Average CPU %:** ~28%

Figure 41) Parallel dd performance stats NFSv4.1 (pNFS), 1MB wsize/rsize.





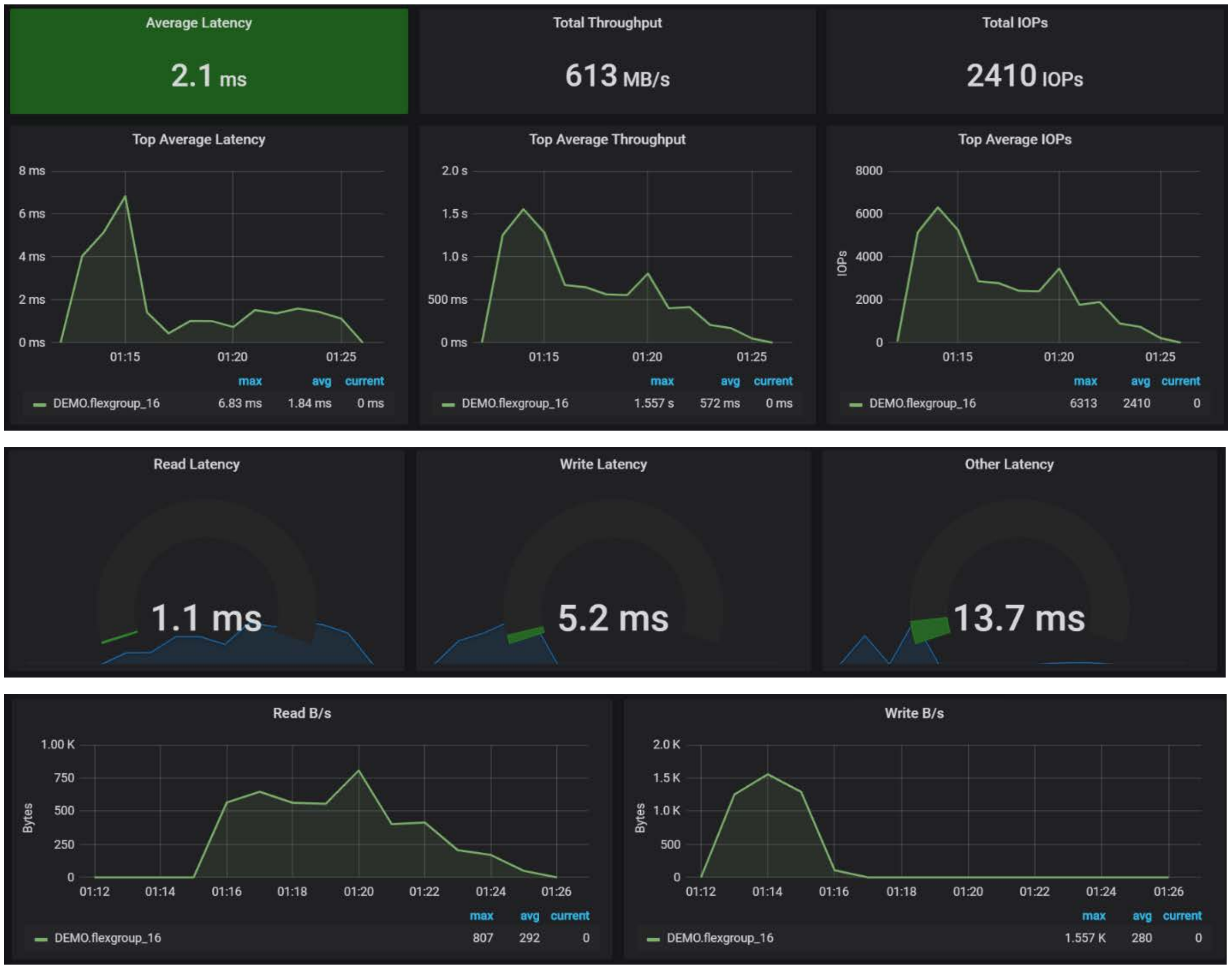


NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                     Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                           -
                              <=16KB                               16
                              <=64KB                               16
                             <=256KB                               16
                             <=512KB                            17520
                               <=1MB                           262144
    nfs41_write_size_histo                                          -
                              <=512B                               48
                               <=4KB                                1
                              <=64KB                               22
                             <=128KB                               64
                             <=256KB                               99
                             <=512KB                              160
                               <=1MB                           262144
```

- 256K wsize/rsize – NFSv4.1 (pNFS)
- **Completion time:** ~12 minutes, 16 seconds
- **Average maximum read throughput:** 807MBps
- **Average maximum write throughput:** 1.56GBps
- **Average CPU %:** ~28%

Figure 42) Parallel dd performance stats NFSv4.1 (pNFS), 256K wsize/rsize.



	max	avg	current
Top Average Latency: DEMO.flexgroup_16	6.83 ms	1.84 ms	0 ms
Top Average Throughput: DEMO.flexgroup_16	1.557 s	572 ms	0 ms
Top Average IOPs: DEMO.flexgroup_16	6313	2410	0
Read B/s: DEMO.flexgroup_16	807	292	0
Write B/s: DEMO.flexgroup_16	1.557 K	280	0

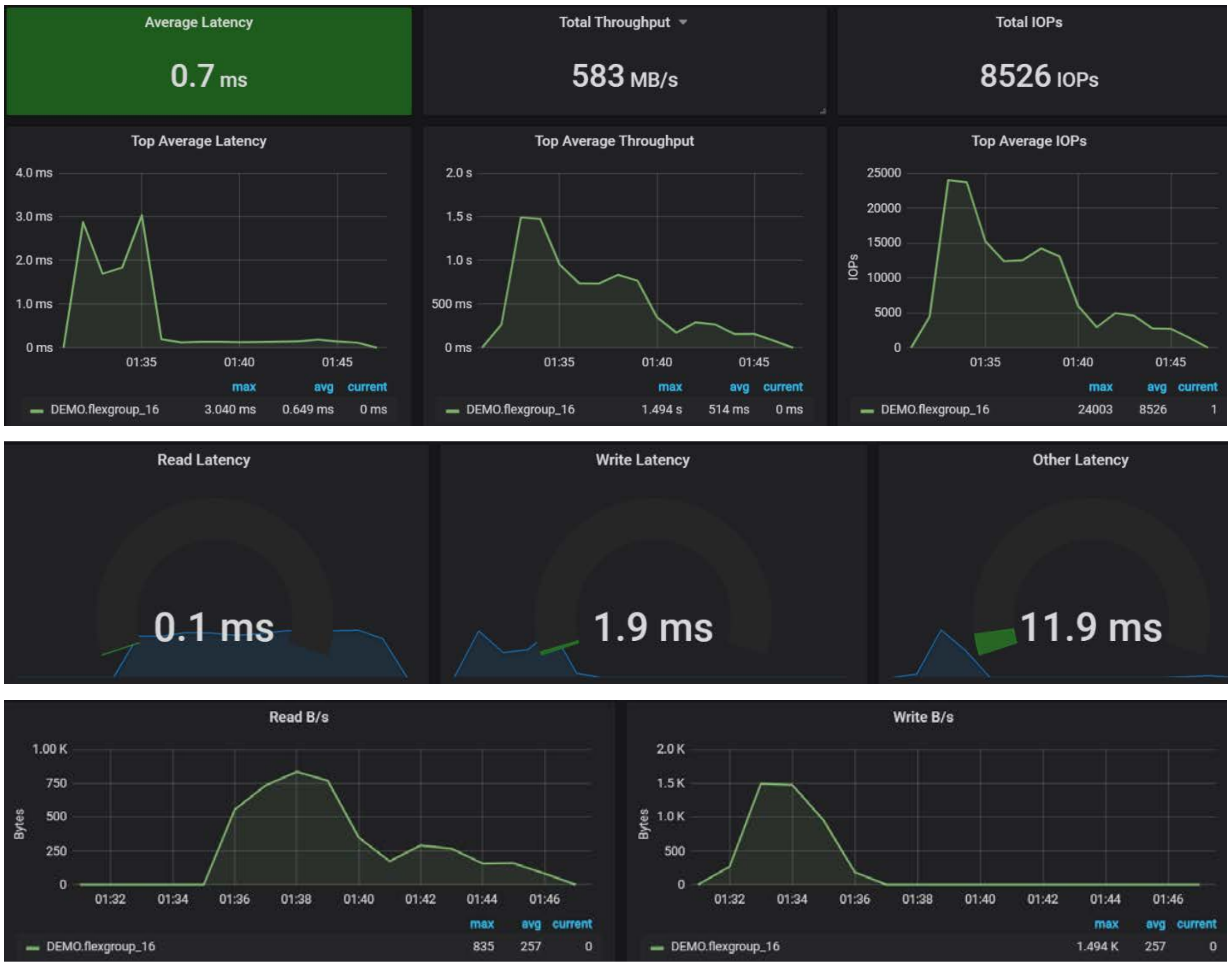
NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                     Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                           -
                               <=16KB                              16
                               <=64KB                              16
                              <=128KB                           69920
                              <=256KB                         1048576
    nfs41_write_size_histo                                          -
                               <=512B                              48
                                <=4KB                              17
                                <=8KB                              20
                               <=16KB                              49
                               <=32KB                              21
                               <=64KB                              56
                              <=128KB                             104
                              <=256KB                         1048576
```

- 64K wsize/rsize – NFSv4.1 (pNFS)
- **Completion time:** ~13 minutes, 57 seconds
- **Average maximum read throughput:** 835MBps
- **Average maximum write throughput:** 1.49GBps

- **Average CPU %:** ~26%

Figure 43) Parallel dd performance stats NFSv4.1 (pNFS), 64K wsize/rsize.



NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                   Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                          -
                              <=16KB                          262144
                              <=64KB                         4194304
    nfs41_write_size_histo                                         -
                              <=512B                              48
                               <=4KB                              32
                               <=8KB                              30
                              <=16KB                              52
                              <=32KB                             112
                              <=64KB                         4194285
```

Table 25) NFSv4.1 pNFS wsize/rsize performance summary.

Test (rsize/wsize)	Average CPU %	Average Latency (ms)	Average IOPS	Completion Time
NFSv4.1 pNFS (1MB)	28%	11.1	818	10m54s

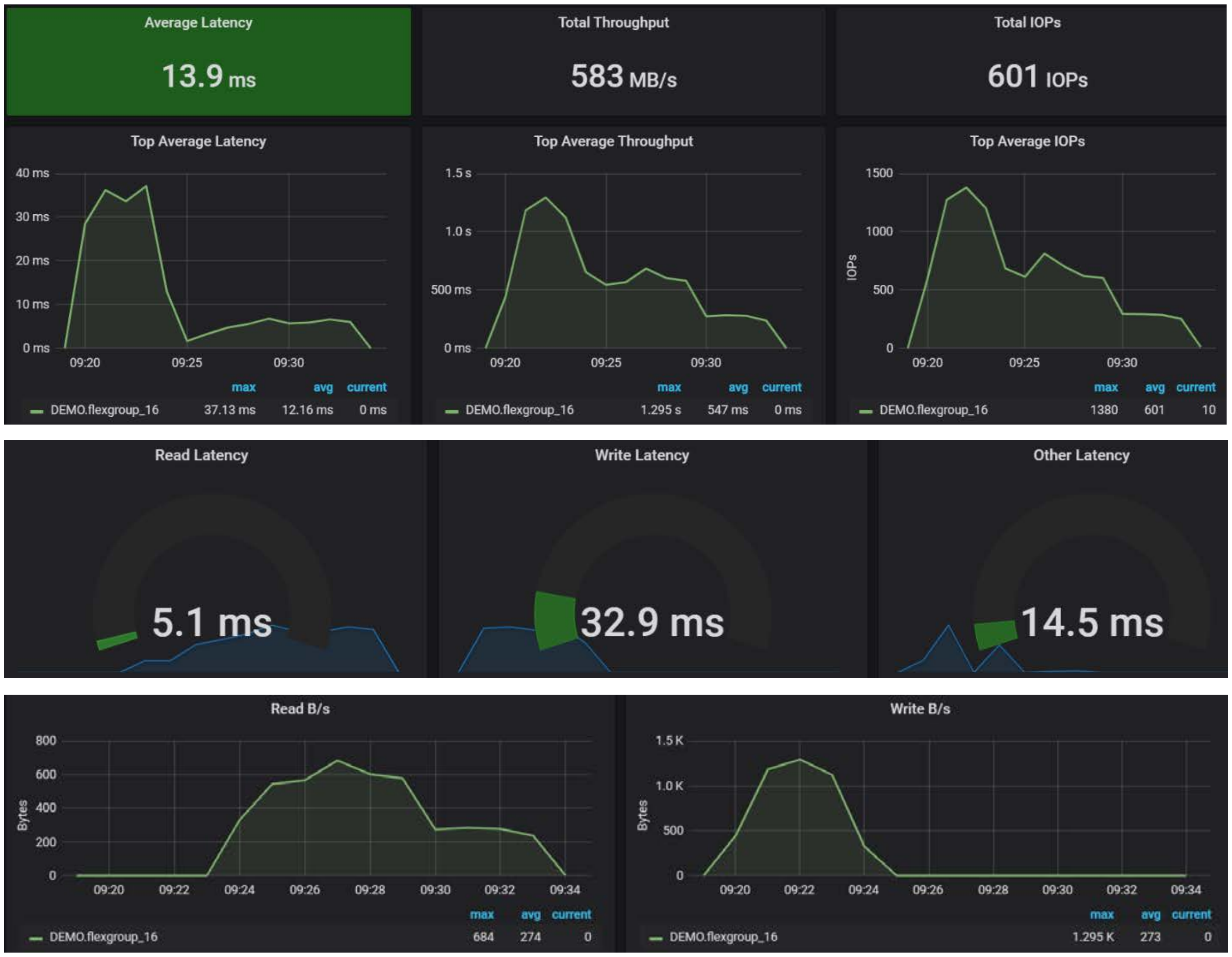
Test (rsize/wsize)	Average CPU %	Average Latency (ms)	Average IOPS	Completion Time
NFSv4.1 pNFS (256K)	28%	2.1	2410	12m16s
NFSv4.1 pNFS (64K)	26%	.2	8526	13m57s

Test #6: Multithreaded dd Operations (NFSv4.1 delegations) – Multiple Clients

This test runs a multithreaded dd script that creates 8 folders with two 16GB files per folder. Two clients run the tests at the same time, splitting the workload to two different network ports. The tests use NFSv4.1 with read and write delegations enabled (for information on delegations, see the section “NFSv4 Delegations”) to measure time to completion and various other performance stats like throughput and NFS operations. The workload is a sequential 50/50 read/write workload.

- 1MB wsize/rsize – NFSv4.1 (Read/Write delegations)
- **Completion time:** ~13 minutes, 16 seconds
- **Average maximum read throughput:** 684MBps
- **Average Maximum write throughput:** 1.29GBps
- **Average CPU %:** ~24%

Figure 44) Parallel dd performance stats NFSv4.1 (delegations); 1MB wsize/rsize.

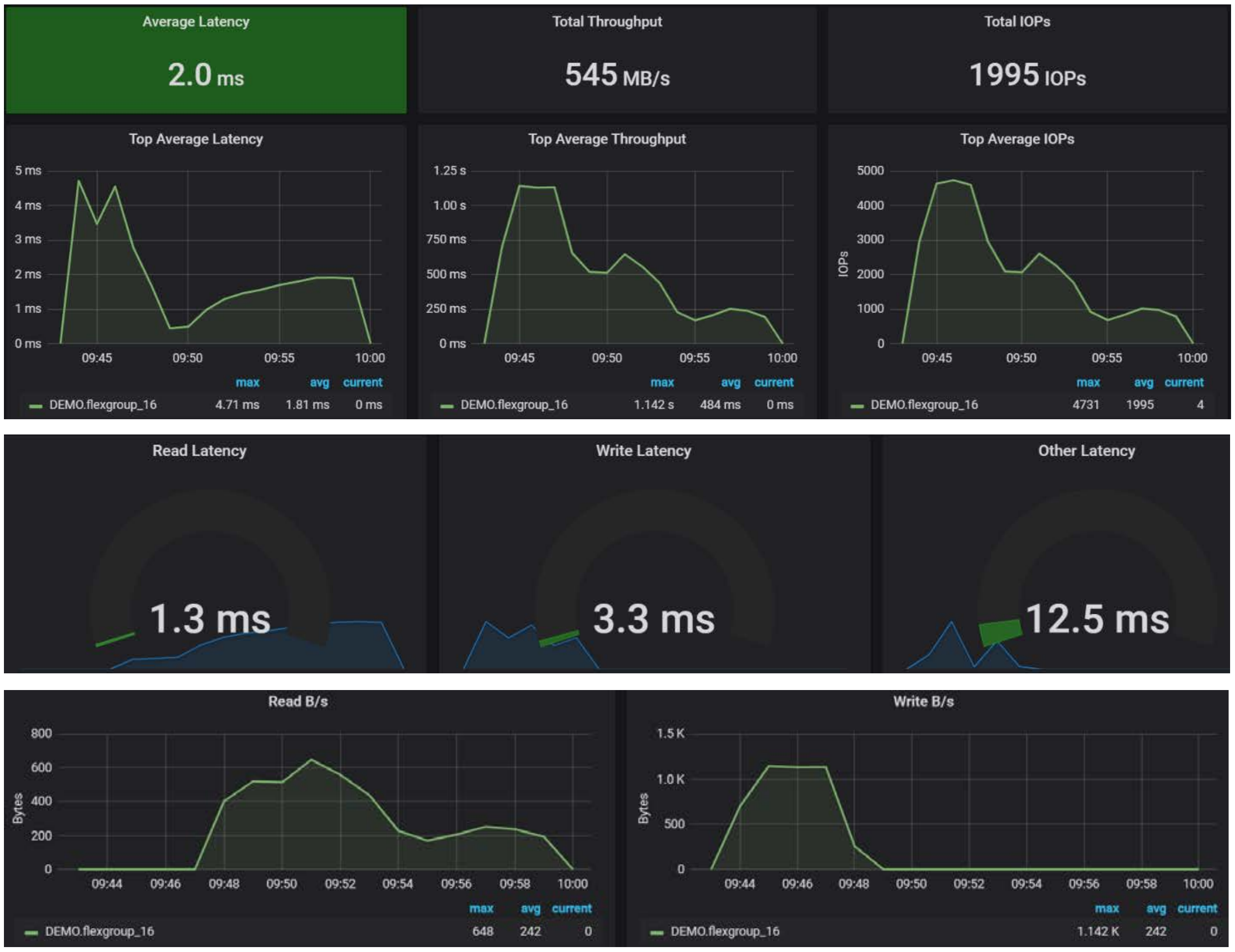


NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                          Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                                -
                              <=16KB                                    16
                              <=64KB                                    16
                             <=256KB                                    16
                               <=1MB                                262144
    nfs41_write_size_histo                                               -
                              <=512B                                    48
                              <=32KB                                     1
                              <=64KB                                     2
                             <=128KB                                     4
                             <=256KB                                     9
                             <=512KB                                    14
                               <=1MB                                262143
```

- 256K wsize/rsize – NFSv4.1 (Read/Write delegations)
- **Completion time:** ~15 minutes, 25 seconds
- **Average maximum read throughput:** 648MBps
- **Average maximum write throughput:** 1.14GBps
- **Average CPU %:** ~24%

Figure 45) Parallel dd performance stats NFSv4.1 (delegations), 256K wsize/rsize.



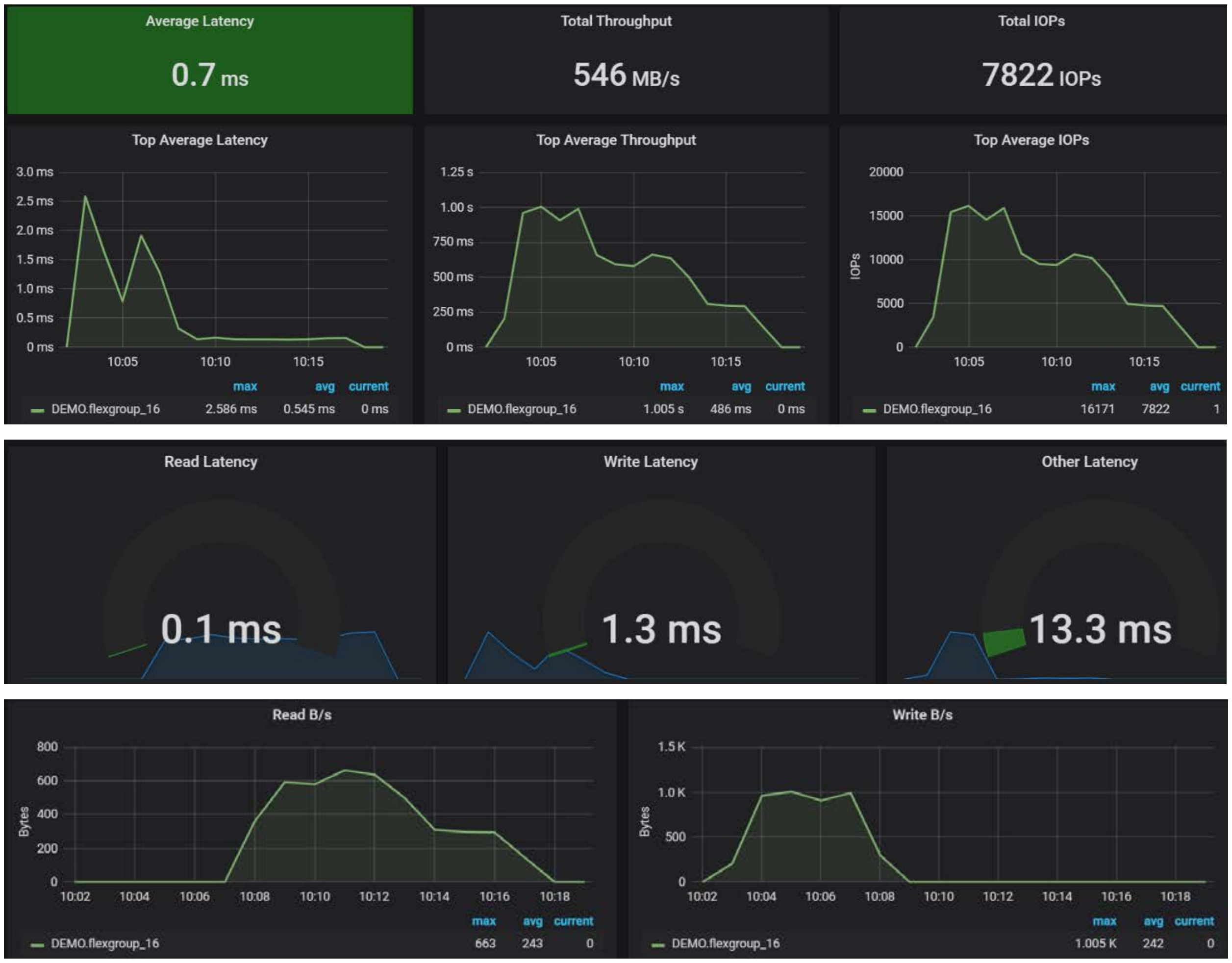
Average Latency
2.0 ms
Total Throughput
545 MB/s
Total IOPs
1995 IOPs
Top Average Latency
DEMO.flexgroup_16 max 4.71 ms avg 1.81 ms current 0 ms
Top Average Throughput
DEMO.flexgroup_16 max 1.142 s avg 484 ms current 0 ms
Top Average IOPs
DEMO.flexgroup_16 max 4731 avg 1995 current 4
Read Latency
1.3 ms
Write Latency
3.3 ms
Other Latency
12.5 ms
Read B/s
DEMO.flexgroup_16 max 648 avg 242 current 0
Write B/s
DEMO.flexgroup_16 max 1.142 K avg 242 current 0

NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                     Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                           -
                               <=16KB                              16
                               <=64KB                              16
                              <=256KB                         1048576
    nfs41_write_size_histo                                          -
                               <=512B                              48
                                <=8KB                               1
                               <=16KB                               2
                               <=32KB                               6
                               <=64KB                              11
                              <=128KB                              12
                              <=256KB                         1048575
```

- 64K wsize/rsize – NFSv4.1 (Read/Write delegations)
- **Completion time:** ~13 minutes, 48 seconds
- **Average maximum read throughput:** 663MBps
- **Average maximum write throughput:** 1GBps
- **Average CPU %:** ~34%

Figure 46) Parallel dd performance stats NFSv4.1 (delegations), 64K wsize/rsize.



NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                    Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                          -
                              <=16KB                              16
                              <=64KB                         4194304
    nfs41_write_size_histo                                         -
                              <=512B                              48
                               <=4KB                               1
                               <=8KB                               1
                              <=16KB                              10
                              <=32KB                              12
                              <=64KB                         4194304
```

Table 26) NFSv4.1 delegations wsize/rsize performance summary.

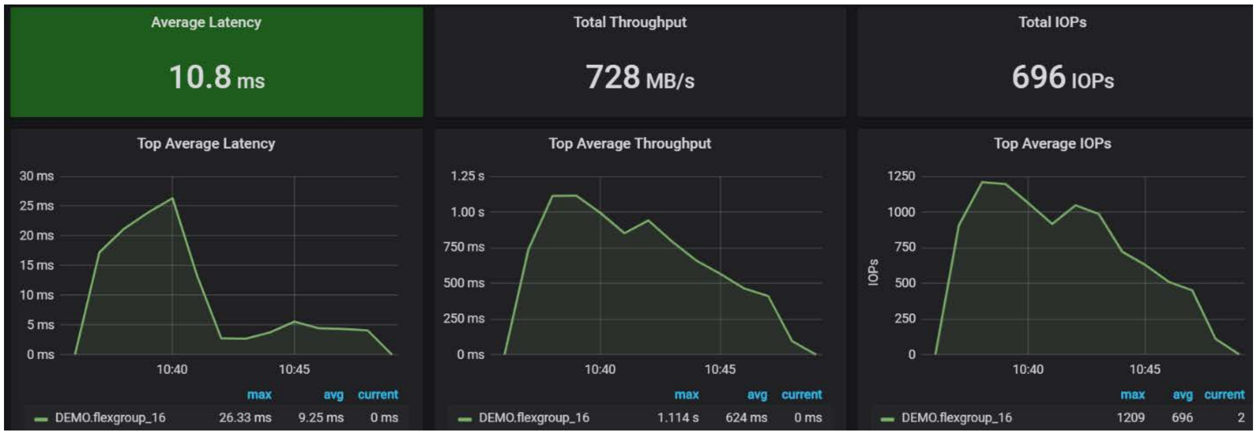
Test (rsize/wsize)	Average CPU %	Average Latency (ms)	Average IOPS	Completion Time
NFSv4.1 delegations (1MB)	24%	13.9	601	13m48s
NFSv4.1 delegations (256K)	24%	2	1995	15m25s
NFSv4.1 delegations (64K)	34%	.7	7822	13m6s

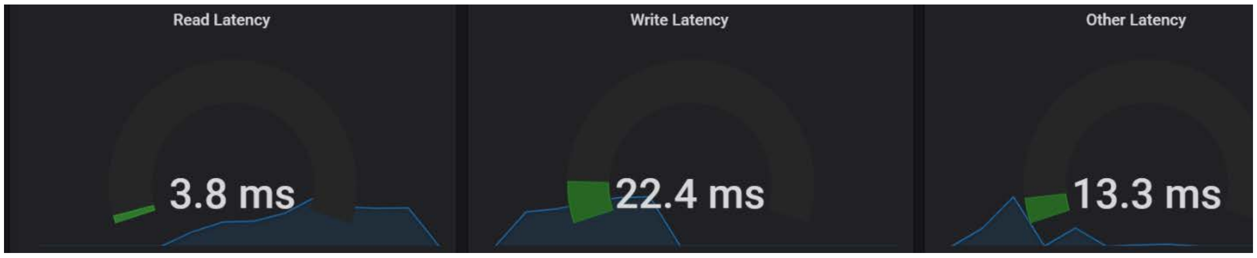
Test #7: Multithreaded dd Operations (NFSv4.1 pNFS + delegations) – Multiple Clients

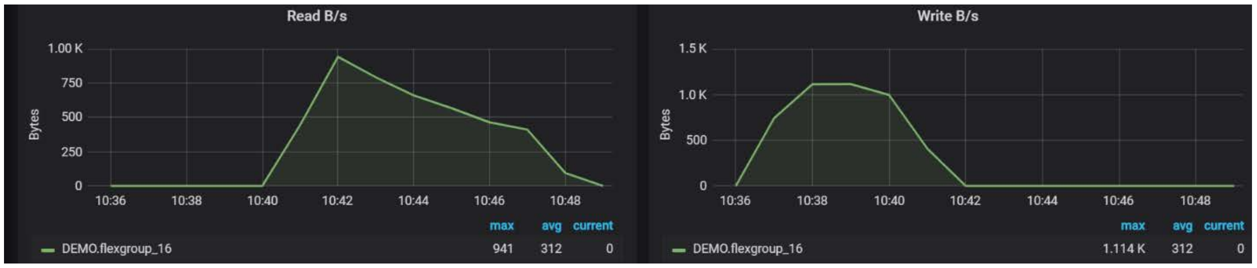
This test runs a multithreaded dd script that creates 8 folders with two 16GB files per folder. Two clients run the tests at the same time, splitting the workload to two different network ports. The tests use NFSv4.1 with pNFS and read/write delegations enabled to measure time to completion and various other performance stats like throughput and NFS operations. The workload is a sequential 50/50 read/write workload.

- 1MB wsize/rsize – NFSv4.1 (pNFS + delegations)
- **Completion time:** ~11 minutes, 7 seconds
- **Average maximum read throughput:** 941MBps
- **Average maximum write throughput:** 1.11GBps
- **Average CPU %:** ~27%

Figure 47) Parallel dd performance stats NFSv4.1 (pNFS + delegations), 1MB wsize/rsize.





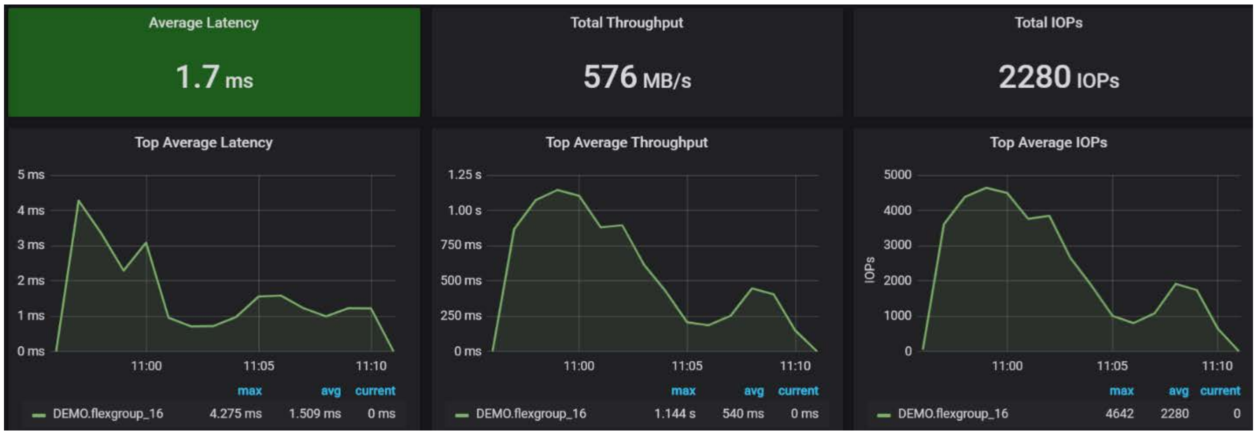


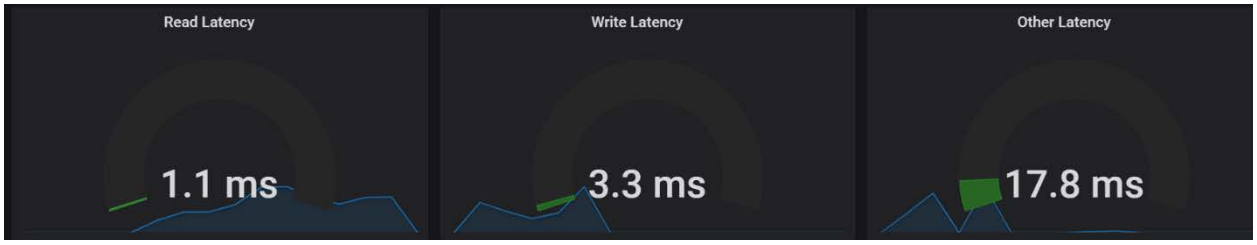
NFS read and write size breakdown:

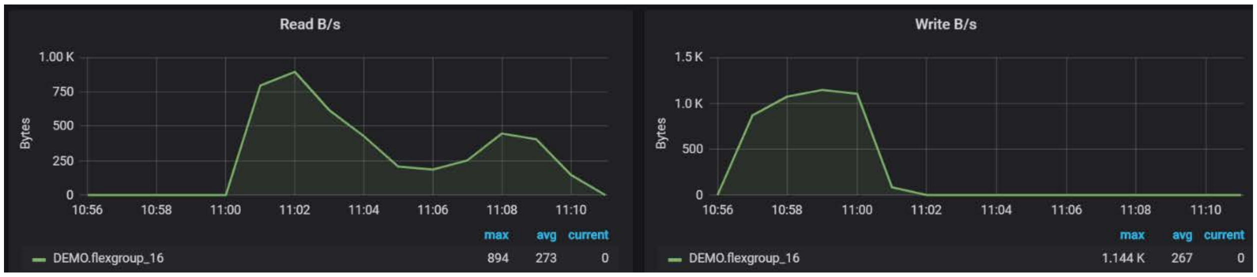
```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                     Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                           -
                              <=16KB                               16
                              <=64KB                               16
                             <=256KB                               16
                             <=512KB                            17520
                               <=1MB                           262144
    nfs41_write_size_histo                                          -
                              <=512B                               48
                              <=16KB                                1
                              <=32KB                               21
                             <=128KB                               45
                             <=256KB                               83
                             <=512KB                              247
                               <=1MB                           262144
```

- 256K wsize/rsize – NFSv4.1 (pNFS + delegations)
- **Completion time:** ~13 minutes, 26 seconds
- **Average maximum read throughput:** 795MBps
- **Average maximum write throughput:** 1.14GBps
- **Average CPU %:** ~28%

Figure 48) Parallel dd performance stats NFSv4.1 (pNFS + delegations), 256K wsize/rsize.





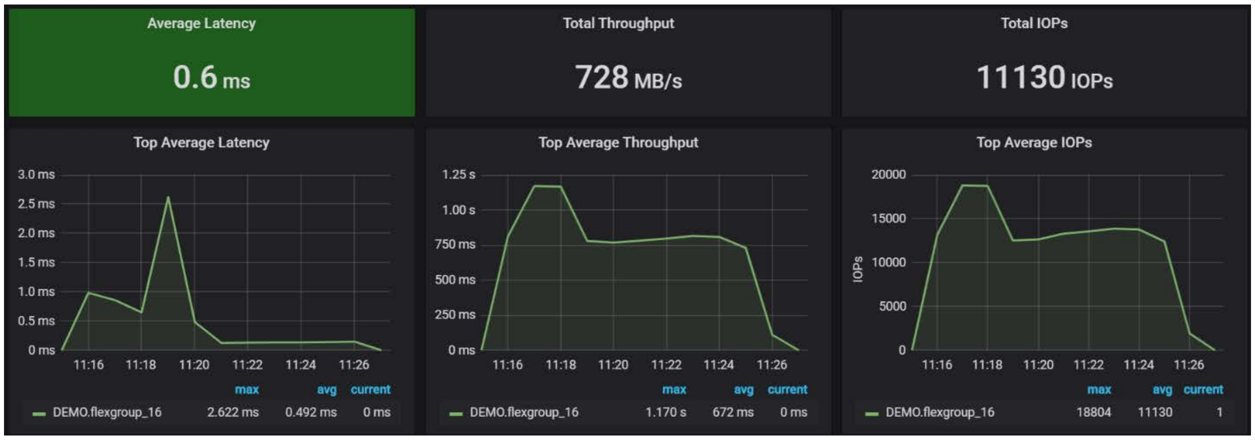


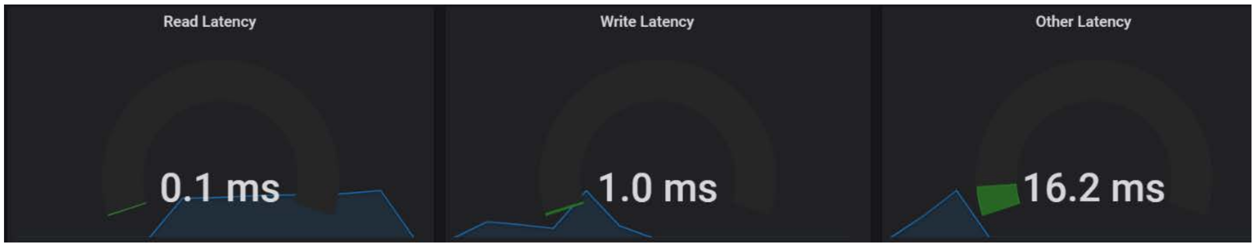
NFS read and write size breakdown:

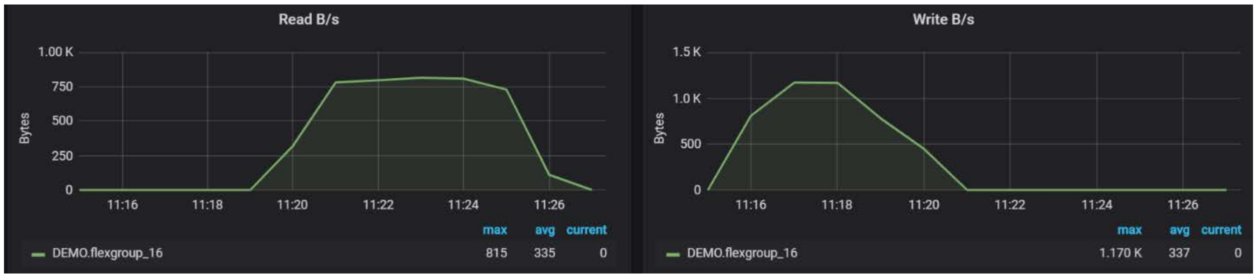
```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                          Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                                -
                              <=16KB                                    16
                              <=64KB                                    16
                             <=128KB                                 69920
                             <=256KB                               1048576
    nfs41_write_size_histo                                               -
                              <=512B                                    48
                               <=8KB                                    24
                              <=16KB                                    32
                              <=32KB                                    91
                              <=64KB                                    84
                             <=128KB                                   177
                             <=256KB                               1048537
```

- 64K wsize/rsize – NFSv4.1 (pNFS + delegations)
- **Completion time:** ~10 minutes, 6 seconds
- **Average maximum read throughput:** 815MBps
- **Average maximum write throughput:** 1.17GBps
- **Average CPU %:** ~27%

Figure 49) Parallel dd performance stats NFSv4.1 (pNFS + delegations), 64K wsize/rsize.







NFS read and write size breakdown:

```
cluster::*> statistics show -object nfsv4_1 -counter *size*

    Counter                                                        Value
    -------------------------------- --------------------------------
    nfs41_read_size_histo                                              -
                              <=16KB                              262144
                              <=64KB                             4194304
    nfs41_write_size_histo                                             -
                              <=512B                                  48
                               <=4KB                                  18
                               <=8KB                                  87
                              <=16KB                                  80
                              <=32KB                                 218
                              <=64KB                             4194251
```

Table 27) NFSv4.1 pNFS/delegations wsize/rsize performance summary.

Test (rsize/wsize)	Average CPU %	Average Latency (ms)	Average IOPS	Completion Time
NFSv4.1 pNFS/delegations (1MB)	27%	10.8	696	11m7s
NFSv4.1 pNFS/delegations (256K)	28%	1.7	2280	13m26s
NFSv4.1 pNFS/delegations (64K)	36.5%	.6	11130	10m6s

Observations

This section covers the observations from performance testing and discusses trends in the results, as well as notable statistical differences.

High File Count Tests – 1 Million Tiny Files and Folders (Python Script)

This test uses python to leverage multiple CPU threads per client to create 1 million files and folders. All files are less than 1K in size and there is a file system structure of n number of top-level directories -> n number of subdirectories per top-level directory > 10 files per subdirectory.

This test uses NFSv3 only. NFSv4.1 performance was not suitable for this type of workload. As always, it's important to test your own workloads with protocol versions to check for suitable performance.

The wsize/rsize mount options didn't make a ton of difference for overall completion time or for latency for very small files. But the effect can be seen on number of operations and CPU %. In general, for workloads that write a large number of small files, test different wsize/rsize options to find the best overall performance.

High File Count Tests – 800,000 files (NFSv3 and NFSv4.1)

This test uses multi-threaded dd to create 8 folders and 100,000 files per folder (800,000 total files). The files grow to 512K. dd used an 8K block size.

The wsize/rsize settings impacted a few areas for this small file/high file count 100% write workload. NFSv3 outperformed NFSv4.1 by a large margin for this specific workload, taking roughly ½ the amount of time to complete.

The completion times appear to be tied to throughput, where NFSv3 was able to do 2x the amount of MB/s on average as compared to NFSv4.1. Much of this is likely tied to the statelessness of NFSv3, whereas NFSv4.1 uses compound operations and share locks, which requires more processing for each packet.

pNFS seemed to add some performance benefits to the workload when compared with basic NFSv4.1, with lower completion times.

Total average IOPS tracked with the wsize/rsize for 64K and 256K but were nearly identical for 1MB and 256K. This is because the file sizes were 512K, so the 1MB wsize/rsize didn't add benefits for total IOPS, since no files were able to push 1MB operations.

Table 28) Summary of Performance Statistics – High File Count: 800,000 files.

Test (wsize/rsize setting)	Completion Time	CPU %	Average Throughput (MBps)	Average Total IOPS
NFSv3 (1MB)	17m29s	32%	351	7696
NFSv3 (256K)	16m34s	34.5%	372	8906
NFSv3 (64K)	16m11s	39%	394	13566
NFSv4.1 (1MB)	38m20s	26%	167	7746
NFSv4.1 (256K)	38m15s	27.5%	167	7957
NFSv4.1 (64K)	38m13s	31%	172	10221
NFSv4.1 pNFS (1MB)	35m44s	27%	171	8330
NFSv4.1 pNFS (256K)	35m9s	28.5%	175	8894
NFSv4.1 pNFS (64K)	36m41s	33%	171	10751

Parallel dd/Sequential I/O Tests

The wsize/rsize settings affected a few areas for this large file sequential read/write workload. Completion times for each job were within a range of a few minutes, with NFSv4.1 outperforming NFSv3 for this specific workload.

- **IOPS.** Larger wsize/rsize mount options reduced the total number of IOPS in general. This is because a larger rsize/wsize transfers more NFS traffic per packet, while smaller rsize/wsize transfers smaller chunks in more packets. These IOPS have a direct effect on latency and CPU%.

- **CPU %.** Larger wsize/rsize mount options reduced the total number of IOPS, which means the storage CPU had fewer operations to process. CPU utilization was also affected by the features being used. For example, NFSv4.1 with pNFS lowered overall CPU use because it redirects reads and writes to local paths, which prevents extra processing over the cluster network. NFSv4.1 delegations showed higher CPU % because they use locking mechanisms to reserve the files, which adds to the CPU processing.
- **Latency.** Latency is a measure of how long it takes for an operation to process after it arrives to the storage system. Latency can be affected by a variety of things, such as network, CPU, disk utilization, etc. In the case of rsize/wsize, larger settings showed an increase in overall latency – especially for writes. This is because more data is being sent per operation, so the storage must process more information. However, since there are fewer IOPS to process, higher latency did not impact the completion times too much. Latency was reduced when NFSv4.1 with pNFS was used, as it avoided the cluster network for read and write operations by redirecting to local paths.

  These test results are intended to be a referential data point. The workload type and application needs for latency, IOPS and CPU utilization should always be factored in when making decisions about what is and is not acceptable.
- **pNFS**. Use of NFSv4.1 with pNFS reduced overall load on the cluster network, which resulted in lower CPU % and lower latency for operations. For example, in the NFSv4.1 tests, the cluster network showed ~30% “cluster busy” in `statistics show-periodic` output during the tests (around 800-900MB transferred over the cluster network). pNFS tests showed 0% “cluster busy” with no data being transferred over the cluster network. This positively affected the completion times of the jobs, which finished 2-5 minutes faster than the NFSv4.1 jobs without pNFS.

Summary of Sequential I/O Results

Modifying rsize/wsize values for NFS mounts can offer some performance benefits, but much of those benefits are dependent on the workload needs. As with any workload, testing different scenarios and configurations before deploying in production makes the most sense.

For more information on NFS mount option recommendations, see “NFS Client Best Practices.”

Table 29) Summary of completion times – parallel dd, sequential large files.

Test (wsize/rsize setting)	Completion Time
NFSv3 (1MB)	15m23s
NFSv3 (256K)	14m17s
NFSv3 (64K)	14m48s
NFSv4.1 (1MB)	15m6s
NFSv4.1 (256K)	12m10s
NFSv4.1 (64K)	15m8s
NFSv4.1 (1MB; pNFS)	10m54s
NFSv4.1 (256K; pNFS)	12m16s
NFSv4.1 (64K; pNFS)	13m57s
NFSv4.1 (1MB; delegations)	13m6s
NFSv4.1 (256K; delegations)	15m25s

Test (wsize/rsize setting)	Completion Time
NFSv4.1 (64K; delegations)	13m48s
NFSv4.1 (1MB; pNFS + delegations)	11m7s
NFSv4.1 (256K; pNFS + delegations)	13m26s
NFSv4.1 (64K; pNFS + delegations)	10m6s

Table 30) Summary of performance statistics – parallel dd.

Test (wsize/rsize setting)	Average Read Latency (ms)	Average Read Throughput (MB/s)	Average Write Latency (ms)	Average Write Throughput (MBps)	Average Ops
NFSv3 (1MB)	6	654	27.9	1160	530
NFSv3 (256K)	1.4	766	2.9	1109	2108
NFSv3 (64K)	.2	695	2.2	1110	8791
NFSv4.1 (1MB)	6.5	627	36.8	1400	582
NFSv4.1 (256K)	1.4	712	3.2	1160	2352
NFSv4.1 (64K)	.1	606	1.2	1310	7809
NFSv4.1 (1MB; pNFS)	3.6	840	26.8	1370	818
NFSv4.1 (256K; pNFS)	1.1	807	5.2	1560	2410
NFSv4.1 (64K; pNFS)	.1	835	1.9	1490	8526
NFSv4.1 (1MB; delegations)	5.1	684	32.9	1290	601
NFSv4.1 (256K; delegations)	1.3	648	3.3	1140	1995
NFSv4.1 (64K; delegations)	.1	663	1.3	1000	7822
NFSv4.1 (1MB; pNFS + delegations)	3.8	941	22.4	1110	696
NFSv4.1 (256K; pNFS + delegations)	1.1	795	3.3	1140	2280
NFSv4.1 (64K; pNFS + delegations)	.1	815	1	1170	11130

Default NFS Ports in ONTAP

Table 31) Default NFS ports in ONTAP.

NFS Service	ONTAP Port	Applicable NFS Version	Option to Change the Port
mountd	635	NFSv3	`-mountd-port`
portmapper	111	NFSv3	N/A – Cannot be changed

NFS Service	ONTAP Port	Applicable NFS Version	Option to Change the Port
NLM	4045	NFSv3	`-nlm-port`
NSM	4046	NFSv3	`-nsm-port`
NFS	2049	NFSv3 and NFSv4.x	N/A: cannot be changed
rquota	4049	NFSv3	`-rquotad-port`

Where to Find Additional Information

To learn more about the information that is described in this document, review the following documents and/or websites:

RFCs

- RFC 2203: RPCSEC_GSS Protocol Specification
  http://www.ietf.org/rfc/rfc2203.txt
- RFC 3530: Network File System (NFS) Version 4 Protocol
  http://www.ietf.org/rfc/rfc3530.txt
- RFC 5661: Network File System (NFS) Version 4 Minor Version 1 Protocol
  http://www.ietf.org/rfc/rfc5661.txt
- RFC 5331: RPC: Remote Procedure Call Protocol Specification Version 2
  http://tools.ietf.org/html/rfc5531

LSIC

For end-to-end storage configuration support, refer to the Lenovo Storage Interoperation Center (LSIC):

https://datacentersupport.lenovo.com/lsic

Validate that the exact product and feature versions described in this document are supported for your specific environment. Specific results depend on each customer's installation in accordance with published specifications.

Contacting Support

You can contact Support to obtain help for your issue.

You can receive hardware service through a Lenovo Authorized Service Provider. To locate a service provider authorized by Lenovo to provide warranty service, go to https://datacentersupport.lenovo.com/serviceprovider and use filter searching for different countries. For Lenovo support telephone numbers, see https://datacentersupport.lenovo.com/supportphonelist for your region support details.

Notices

Trademarks

Version History

Version	Date	Document Version History
Version 1.0	March 2021	Initial release