Parallel NFS

ONTAP 9

NetApp
February 02, 2026

Table of Contents

Parallel NFS

Introduction

Learn about parallel NFS (pNFS) in ONTAP

Parallel NFS was introduced as an RFC standard in January 2010 under RFC-5661 to allow clients to directly access file data on NFSv4.1 servers by separating the metadata and data paths. That direct access offers performance benefits by way of data localization, CPU efficiency, and parallelization of operations. A later RFC was authored in 2018 covering pNFS layout types (RFC-8434), which defines standards for file, block and object layouts. ONTAP leverages the file layout type for pNFS operations.

Beginning in July 2024, content from technical reports previously published as PDFs has been integrated with ONTAP product documentation. The ONTAP NFS storage management documentation now includes content from *TR-4063: Parallel Network File System (pNFS) in NetApp ONTAP*.

For years, NFSv3 was the standard version of the NFS protocol that was used for nearly all use cases. However, there were limitations with the protocol, such as lack of statefulness, rudimentary permission model, and basic locking capabilities. NFSv4.0 (RFC 7530) introduced a series of improvements over NFSv3 and was further improved with the subsequent NFSv4.1 (RFC 5661) and NFSv4.2 (RFC 7862) versions, which added features such as parallel NFS (pNFS).

Benefits of NFSv4.x

NFSv4.x provides the following benefits over NFSv3:

- 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 of characters
- Compound operations
- Works only with TCP
- Stateful protocol (not stateless like NFSv3)
- Full Kerberos integration for efficient authentication mechanisms
- NFS referrals
- Support of access control that is compatible with UNIX and Windows
- String-based user and group identifiers
- pNFS (NFSv4.1)
- Extended attributes (NFSv4.2)
- Security labels (NFSv4.2)
- Sparse file ops (FALLOCATE) (NFSv4.2)

For more information about general NFSv4.x, including best practices and details about features, see NetApp Technical Report 4067: NFS Best Practice and Implementation Guide.

Related information

- NFS configuration overview
- NFS management overview
- FlexGroup volumes management
- NFS trunking overview
- https://www.netapp.com/pdf.html?item=/media/19370-tr-4523.pdf
- NetApp Technical Report 4616: NFS Kerberos in ONTAP with Microsoft Active Directory

Learn about pNFS architecture in ONTAP

The pNFS architecture is comprised of three main components: an NFS client that supports pNFS, a metadata server that provides a dedicated path for metadata operations, and a data server that provides localized paths to files.

Client access to pNFS needs network connectivity to data and metadata paths available on the NFS server. If the NFS server contains network interfaces that are not reachable by the clients, then the server might advertise data paths to the client that are inaccessible, which can cause outages.

Metadata server

The metadata server in pNFS is established when a client initiates a mount using NFSv4.1 or later when pNFS is enabled on the NFS server. When this is done, all metadata traffic is sent over this connection and remains on this connection for the duration of the mount, even if the interface is migrated to another node.

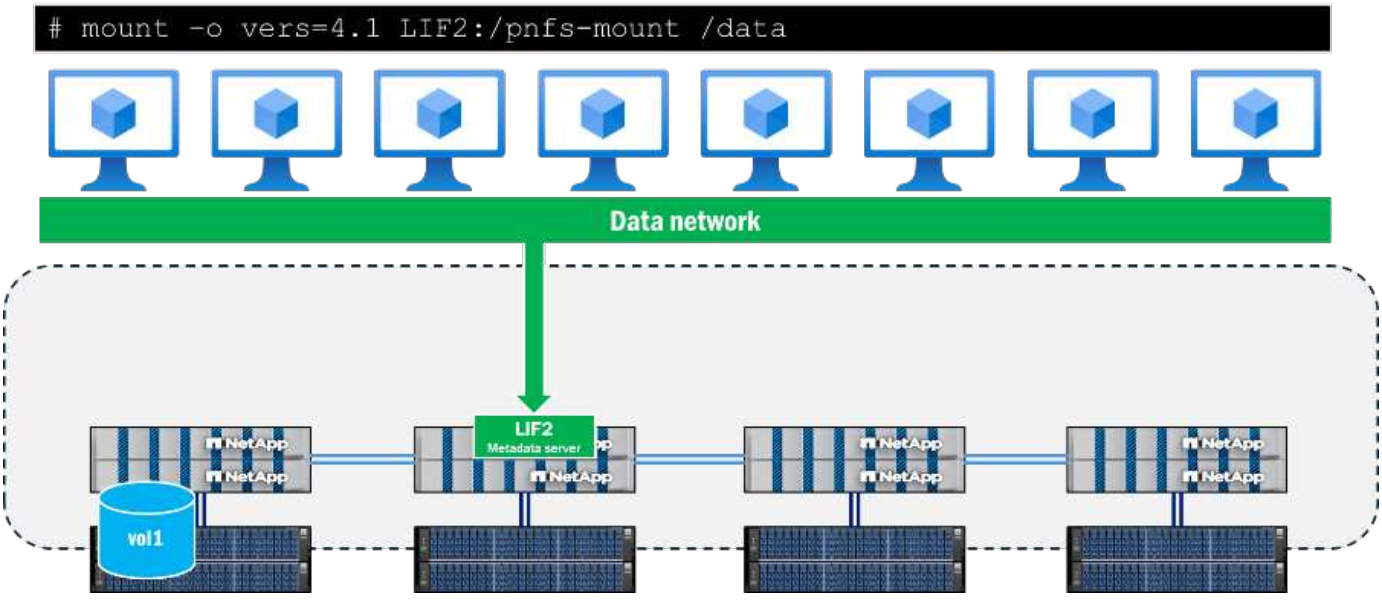


Figure 1. Establish the metadata server in pNFS in ONTAP

pNFS support is determined during the mount call, specifically in the EXCHANGE_ID calls. This can be seen in a packet capture below the NFS operations as a flag. When the pNFS flags `EXCHGID4_FLAG_USE_PNFS_DS` and `EXCHGID4_FLAG_USE_PNFS_MDS` are set to 1, then the interface is eligible for both data and metadata operations in pNFS.

```
Operations (count: 1)
  Opcode: EXCHANGE_ID (42)
      Status: NFS4_OK (0)
      clientid: 0x004050a97100001c
      seqid: 0x00000001
    flags: 0x00060100, EXCHGID4_FLAG_USE_PNFS_DS, EXCHGID4_FLAG_USE_PNFS_MDS, EXCHGID4_FLAG_BIND_PRINC
        0... .... .... .... .... .... .... .... = EXCHGID4_FLAG_CONFIRMED_R: Not set
        .0.. .... .... .... .... .... .... .... = EXCHGID4_FLAG_UPD_CONFIRMED_REC_A: Not set
        .... .... .... .1.. .... .... .... .... = EXCHGID4_FLAG_USE_PNFS_DS: Set
        .... .... .... ..1. .... .... .... .... = EXCHGID4_FLAG_USE_PNFS_MDS: Set
        .... .... .... ...0 .... .... .... .... = EXCHGID4_FLAG_USE_NON_PNFS: Not set
        .... .... .... .... .... ...1 .... .... = EXCHGID4_FLAG_BIND_PRINC_STATEID: Set
        .... .... .... .... .... .... .... ..0. = EXCHGID4_FLAG_SUPP_MOVED_MIGR: Not set
        .... .... .... .... .... .... .... ...0 = EXCHGID4_FLAG_SUPP_MOVED_REFER: Not set
```

Figure 2. Packet capture for pNFS mount

Metadata in NFS generally consists of file and folder attributes, such as file handles, permissions, access and modification times, and ownership information. Metadata can also include create and delete calls, link and unlink calls, and renames.

In pNFS, there is also a subset of metadata calls specific to the pNFS feature and are covered in further detail in RFC 5661. These calls are used to help determine pNFS-eligible devices, mappings of devices to datasets, and other required information. The following table shows a list of these pNFS-specific metadata operations.

Operation	Description
LAYOUTGET	Obtains the data server map from the metadata server.
LAYOUTCOMMIT	Servers commit the layout and update the metadata maps.
LAYOUTRETURN	Returns the layout or the new layout if the data is modified.
GETDEVICEINFO	Client gets updated information on a data server in the storage cluster.
GETDEVICELIST	Client requests the list of all data servers participating in the storage cluster.
CB_LAYOUTRECALL	Server recalls the data layout from a client if conflicts are detected.
CB_RECALL_ANY	Returns any layouts to the metadata server.
CB_NOTIFY_DEVICEID	Notifies of any device ID changes.

Data path information

After the metadata server is established and data operations begin, ONTAP begins to track the device IDs eligible for pNFS read and write operations, as well as the device mappings, which associate the volumes in the cluster with the local network interfaces. This process occurs when a read or write operation is performed in the mount. Metadata calls, such as `GETATTR`. will not trigger these device mappings. As such, running an `ls` command inside of the mount point will not update the mappings.

Devices and mappings can be seen using the ONTAP CLI in advanced privilege, as shown below.

```
::*> pnfs devices show -vserver DEMO
  (vserver nfs pnfs devices show)
Vserver Name     Mapping ID      Volume MSID     Mapping Status
Generation
---------------  --------------- --------------- ---------------
-------------
DEMO             16              2157024470      available       1

::*> pnfs devices mappings show -vserver SVM
  (vserver nfs pnfs devices mappings show)
Vserver Name    Mapping ID      Dsid            LIF IP
--------------  --------------- --------------- --------------------
DEMO            16              2488            10.193.67.211
```

In these commands, the volume names are not present. Instead, the numeric IDs associated with those volumes are used: the master set ID (MSID) and the data set ID (DSID). To find the volumes associated with the mappings, you can use `volume show -dsid [dsid_numeric]` or `volume show -msid [msid_numeric]` in advanced privilege of the ONTAP CLI.

When a client attempts to read or write to a file located on a node that is remote to the metadata server connection, pNFS will negotiate the appropriate access paths to ensure data locality for those operations and the client will redirect to the advertised pNFS device rather than attempting to traverse the cluster network to access the file. This helps reduce CPU overhead and network latency.

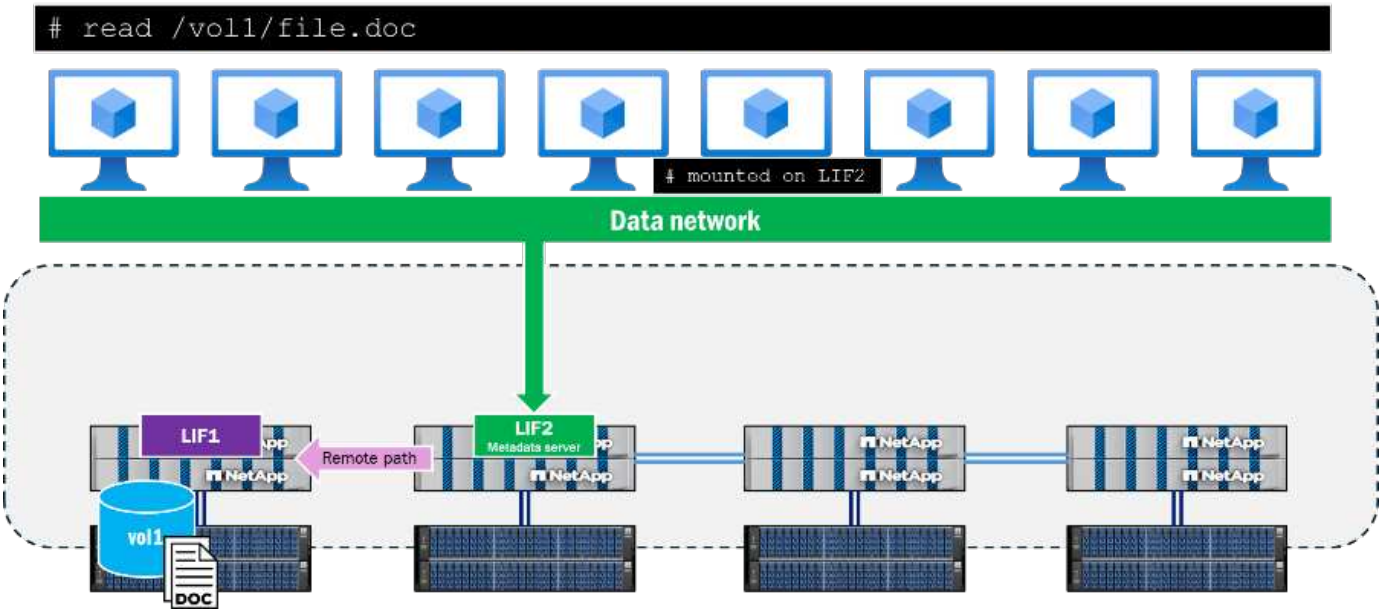


Figure 3. Remote read path using NFSv4.1 without pNFS

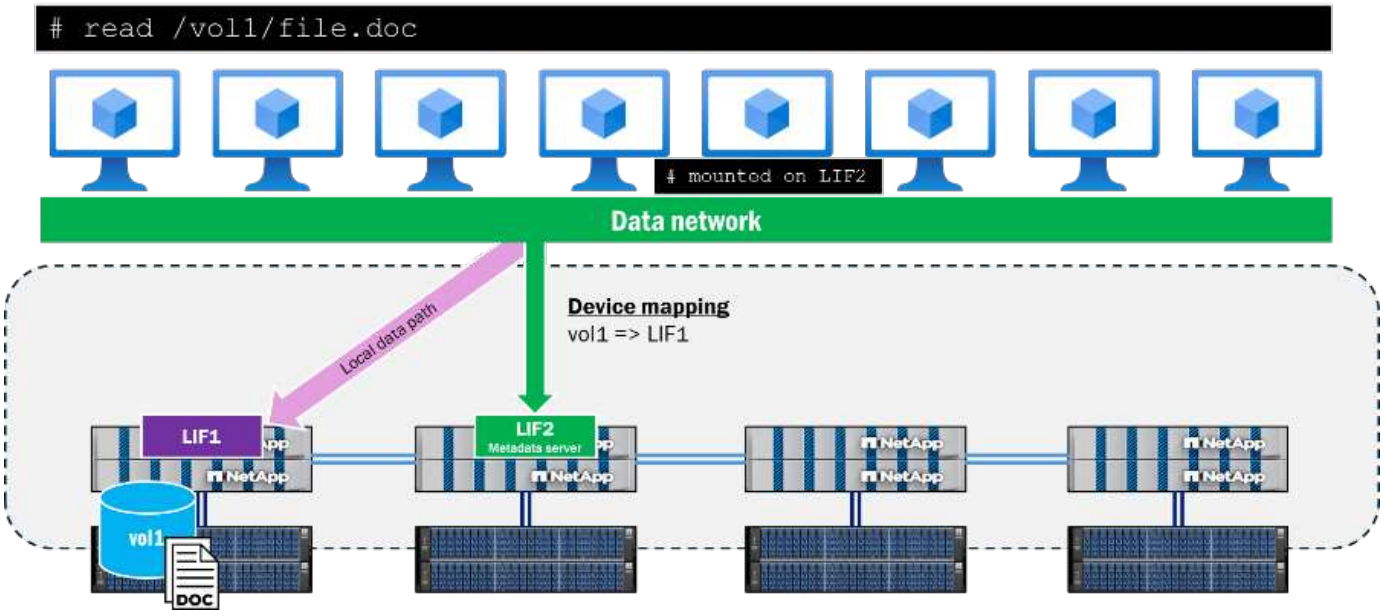


Figure 4. Localized read path using pNFS

pNFS control path

In addition to the metadata and data portions of pNFS, there is also a pNFS control path. The control path is used by the NFS server to synchronize file system information. In an ONTAP cluster, the backend cluster network replicates periodically to ensure all pNFS devices and device mappings are in sync.

pNFS device population workflow

The following describes how a pNFS device populates in ONTAP after a client makes a request to read or write a file in a volume.

1. Client requests read or write; an OPEN is performed and the file handle is retrieved.
2. Once the OPEN is performed, the client sends the file handle to the storage in a LAYOUTGET call over the metadata server connection.
3. LAYOUTGET returns information about the layout of the file, such as the state ID, the stripe size, file segment, and device ID, to the client.
4. The client then takes the device ID and sends a GETDEVINFO call to the server to retrieve the associated IP address with the device.
5. The storage sends a reply with the list of associated IP addresses for local access to the device.
6. The client continues the NFS conversation over the local IP address sent back from the storage.

Interaction of pNFS with FlexGroup volumes

FlexGroup volumes in ONTAP present storage as FlexVol volume constituents that span multiple nodes in a cluster, which allows a workload to leverage multiple hardware resources while maintaining a single mountpoint. Because multiple nodes with multiple network interfaces interact with the workload, it's a natural result to see remote traffic traverse the backend cluster network in ONTAP.

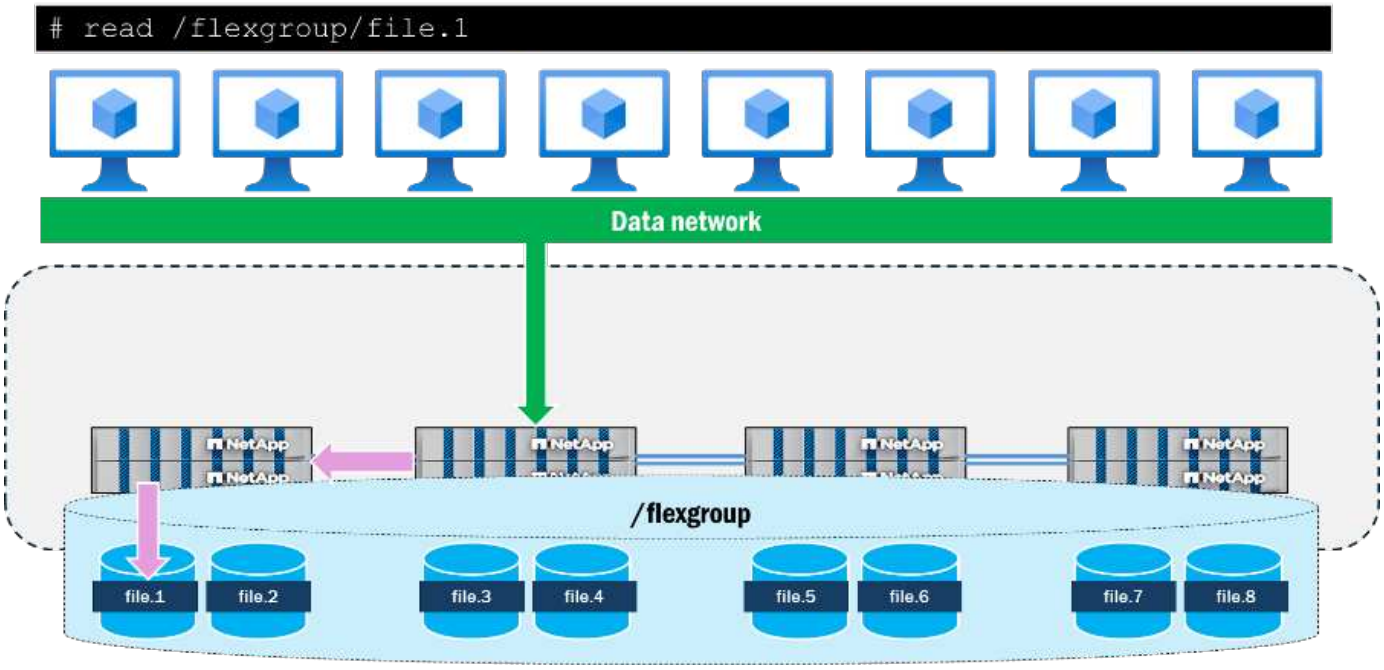


Figure 5. Single file access in a FlexGroup volume without pNFS

When utilizing pNFS, ONTAP keeps track of the file and volume layouts of the FlexGroup volume and maps them to the local data interfaces in the cluster. For example, if a constituent volume that contains a file being accessed resides on node 1, then ONTAP will notify the client to redirect the data traffic to the data interface on node 1.

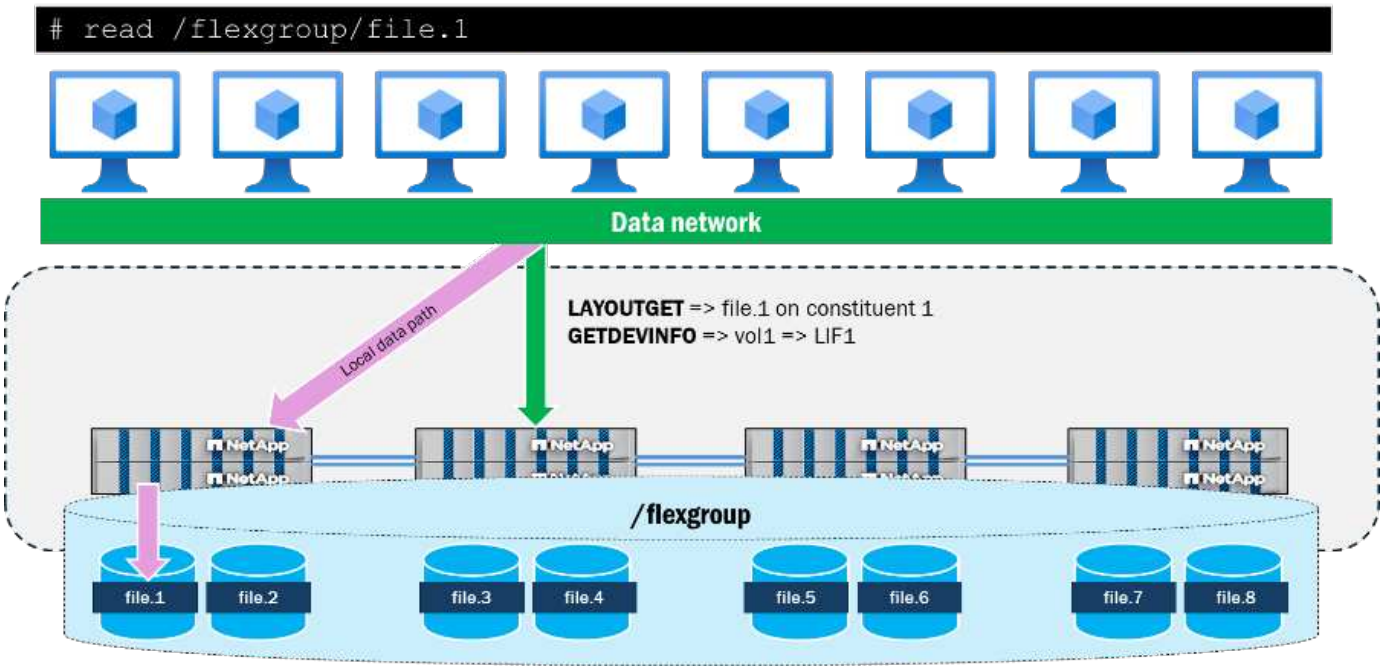


Figure 6. Single file access in a FlexGroup volume with pNFS

pNFS also provides for the presentation of parallel network paths to files from a single client that NFSv4.1 without pNFS does not provide. For example, if a client wants to access four files at the same time from the same mount using NFSv4.1 without pNFS, the same network path would be utilized for all files and the ONTAP cluster would instead send remote requests to those files. The mount path can become a bottleneck for the operations, as they all follow a single path and arrive at a single node and is also servicing metadata operations along with the data operations.

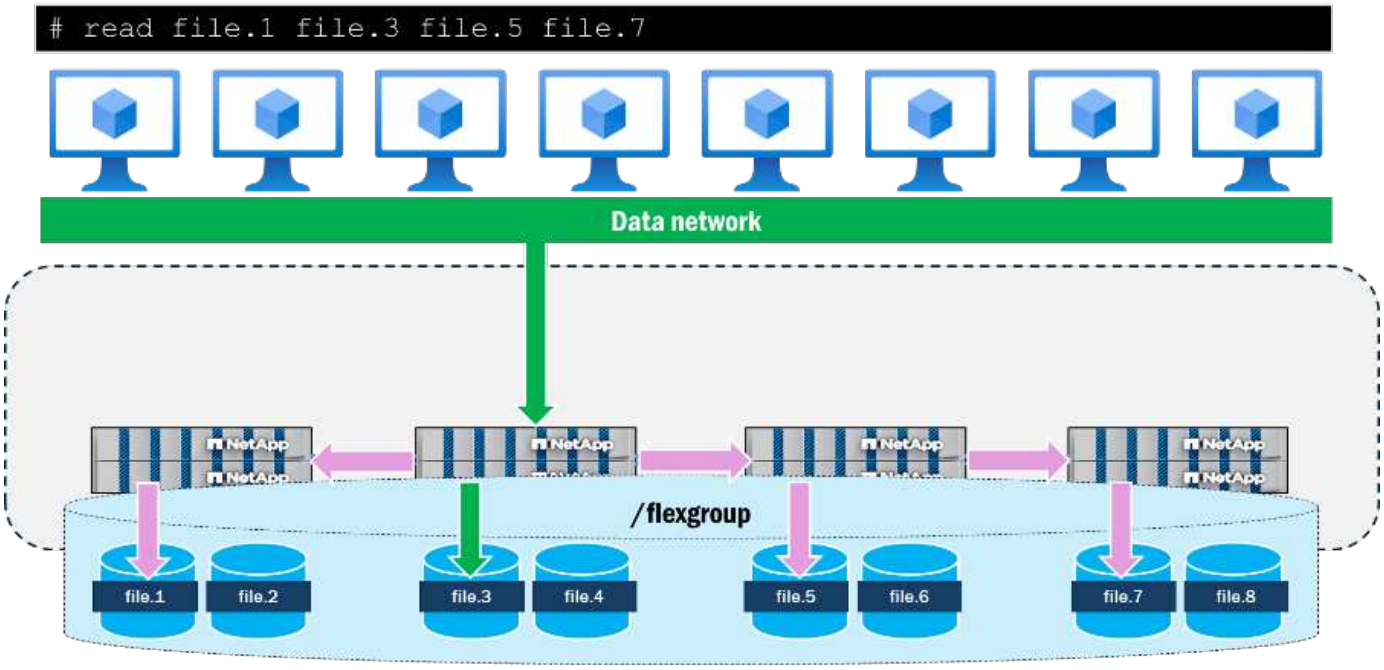


Figure 7. Multiple simultaneous file access in a FlexGroup volume without pNFS

When pNFS is used to access the same four files simultaneously from a single client, the client and server negotiate local paths to each node with the files and uses multiple TCP connections for the data operations, while the mount path acts as the location for all metadata operations. This provides latency benefits by using local paths to the files but also can add throughput benefits by way of multiple network interfaces being used, provided the clients can send enough data to saturate the network.

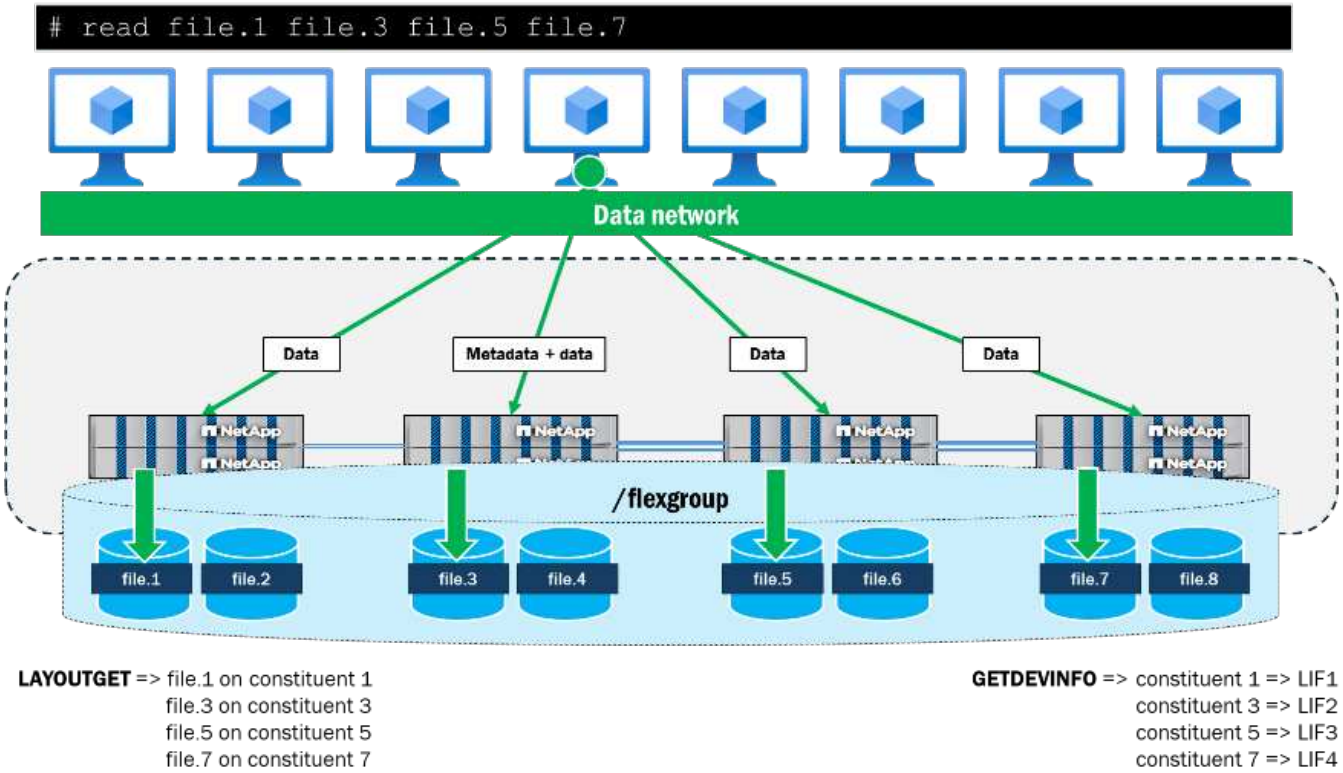


Figure 8. Multiple simultaneous file access in a FlexGroup volume with pNFS

The following shows results from a simple test run on a single RHEL 9.5 client where four 10GB files (all residing on different constituent volumes across two ONTAP cluster nodes) are read in parallel using dd. For

each file, the overall throughput and completion time was improved when using pNFS. When using NFSv4.1 without pNFS, the performance delta between files that were local to the mount point and remote was greater than with pNFS.

Test	Throughput per file (MB/s)	Completion time per file
NFSv4.1: no pNFS	• File.1–228 (local) • File.2–227 (local) • File.3–192 (remote) • File.4–192 (remote)	• File.1–46 (local) • File.2–46.1 (local) • File.3–54.5 (remote) • File.4–54.5 (remote)
NFSv4.1: with pNFS	• File.1–248 (local) • File.2–246 (local) • File.3–244 (local via pNFS) • File.4–244 (local via pNFS)	• File.1–42.3 (local) • File.2–42.6 (local) • File.3–43 (local via pNFS) • File.4–43 (local via pNFS)

Related information

- FlexGroup volumes management
- NetApp Technical Report 4571: FlexGroup Best Practices

pNFS use cases in ONTAP

pNFS can be used with various ONTAP features to improve performance and provide additional flexibility for NFS workloads.

pNFS with nconnect

NFS introduced a new mount option with some more recent clients and servers that provides a way to deliver multiple TCP connections while mounting a single IP address. This provides a mechanism to better parallelize operations, work around NFS server and client limitations, and potentially provide greater overall performance to certain workloads. nconnect is supported in ONTAP 9.8 and later, provided the client supports nconnect.

When using nconnect with pNFS, connections will parallelize using the nconnect option over each pNFS device advertised by the NFS server. For instance, if nconnect is set to four and there are four eligible interfaces for pNFS, then the total number of connections created will be up to 16 per mount point (4 nconnect x 4 IP addresses).

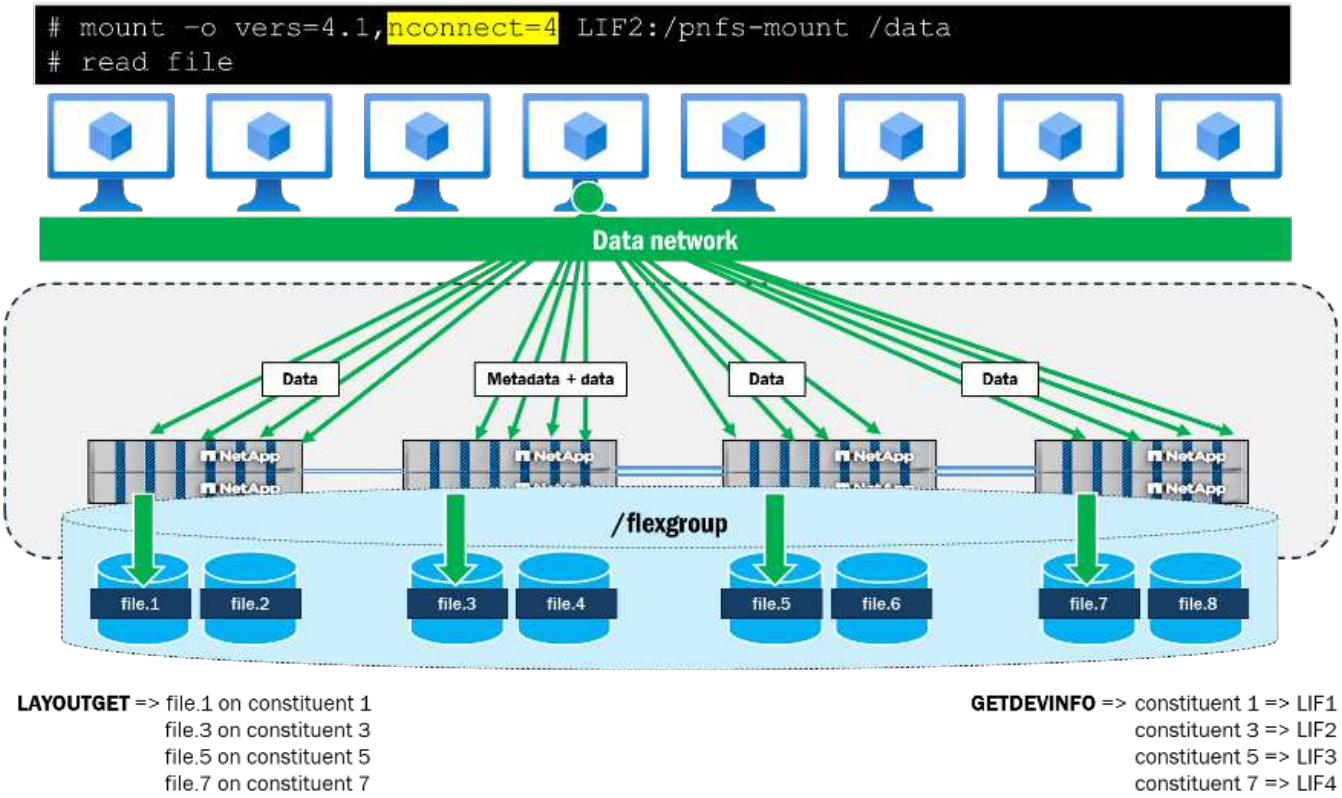


Figure 9. pNFS with nconnect set to 4

Learn more about ONTAP support for NFSv4.1

pNFS with NFSv4.1 session trunking

NFSv4.1 session trunking (RFC 5661, section 2.10.5) is the use of multiple TCP connections between a client and server in order to increase the speed of data transfer. Support for NFSv4.1 session trunking was added to ONTAP 9.14.1 and must be used with clients that also support session trunking.

In ONTAP, session trunking can be used across multiple nodes in a cluster to provide extra throughput and redundancy across connections.

Session trunking can be established in multiple ways:

- **Discover automatically via mount options:** Session trunking in most modern NFS clients can be established via mount options (check your OS vendor's documentation) that signal to the NFS server to send information back to the client about session trunks. This information appears via an NFS packet as an `fs_location4` call.

  The mount option in use depends on the client's OS version. For instance, Ubuntu Linux flavors generally use `max_connect=n` to signal a session trunk is to be used. In RHEL Linux distros, the `trunkdiscovery` mount option is used.

  Ubuntu example

  ```
  mount -o vers=4.1,max_connect=8 10.10.10.10:/pNFS /mnt/pNFS
  ```

RHEL example

```
mount -o vers=4.1,trunkdiscovery 10.10.10.10:/pNFS /mnt/pNFS
```

If you attempt to use `max_connect` on RHEL distros, it will be treated as nconnect instead and session trunking will not work as expected.

- **Establish manually:** You can establish session trunking manually by mounting each individual IP address to the same export path and mount point. For example, if you have two IP addresses on the same node (10.10.10.10 and 10.10.10.11) for an export path of `/pNFS`, you run the mount command twice:

```
mount -o vers=4.1 10.10.10.10:/pNFS /mnt/pNFS
mount -o vers=4.1 10.10.10.11:/pNFS /mnt/pNFS
```

Repeat this process across all interfaces you want to participate in the trunk.

Each node gets its own session trunk. Trunks do not traverse nodes.

When using pNFS, use only session trunking *or* nconnect. Using both will result in undesirable behavior, such as only the metadata server connection getting the benefits of nconnect with the data servers using a single connection.

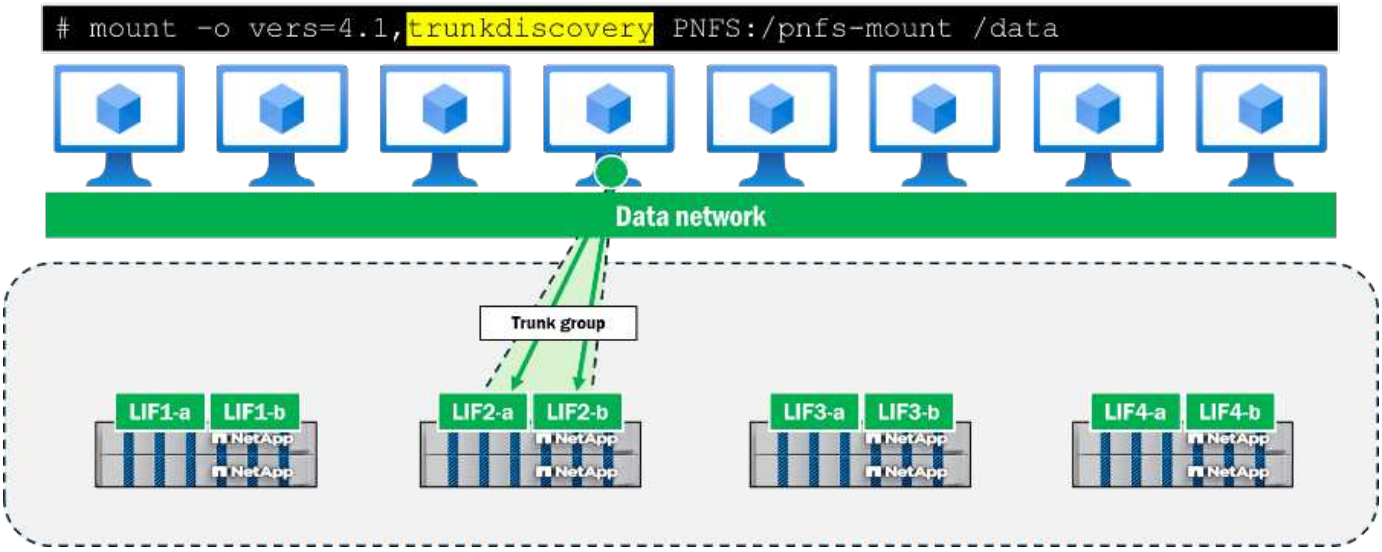
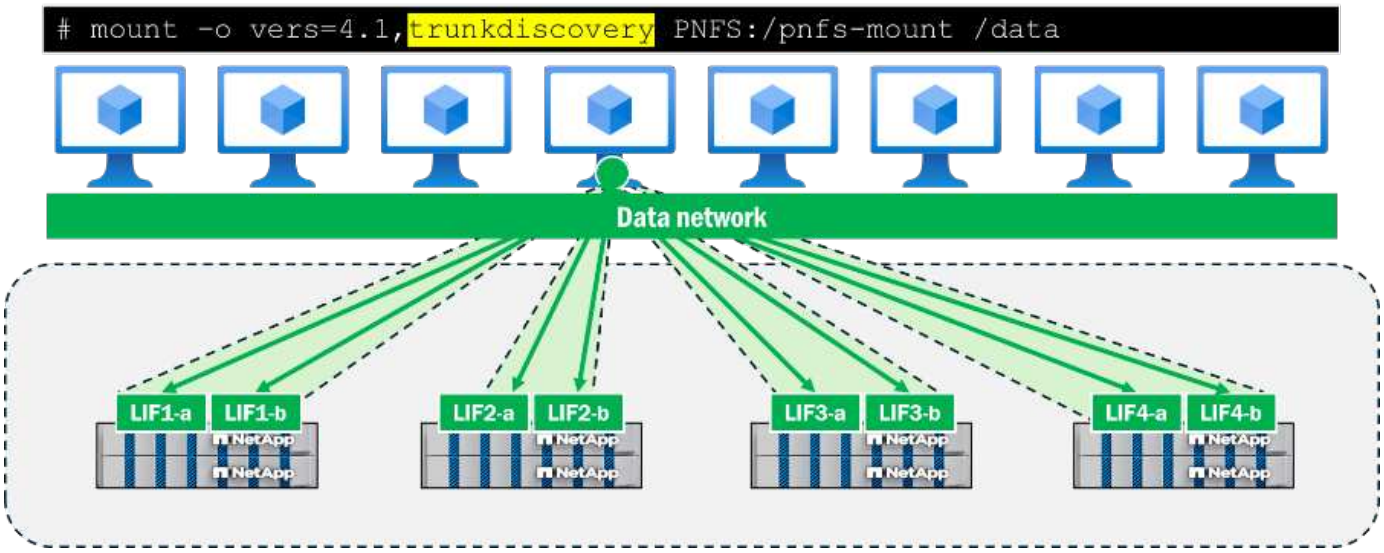


Figure 10. NFSv4.1 session trunking in ONTAP

pNFS can provide a local path to each participating node in a cluster, and when used with session trunking, pNFS can leverage a session trunk per node to maximize throughput for the entire cluster.



When `trunkdiscovery` is used, an added GETATTR call (FS_Locations) is leveraged for the listed session trunk interfaces on the NFS server node where the mount interface is located. Once those are returned, subsequent mounts are made to the returned addresses. This can be seen in a packet capture during mount.

```
198 1.219372  [redacted]  [redacted]  NFS  246 V4 Call (Reply In 199) GETATTR FH: 0x787f5cf1
199 1.219579  [redacted]  [redacted]  NFS  238 V4 Reply (Call In 198) GETATTR

 Opcode: SEQUENCE (53)
     Status: NFS4_OK (0)
     sessionid: 7100001e004090a90000000000000409
     seqid: 0x00000009
     slot id: 0
     high slot id: 63
     target high slot id: 63
   > status flags: 0x00000000
 Opcode: PUTFH (22)
     Status: NFS4_OK (0)
 Opcode: GETATTR (9)
     Status: NFS4_OK (0)
   Attr mask: 0x01000100 (FSID, FS_Locations)
     reqd_attr: FSID (8)
       > fattr4_fsid
     reco_attr: FS_Locations (24)
       fattr4_fs_locations
           pathname components: 0
         fs_location4
             num: 1
           fs_location4
             servers
                 num: 1
               server: [redacted]
                   length: 14
                   contents: [redacted]
                   fill bytes: opaque data
             pathname components: 0
```

Figure 11. NFS session trunk discovery during mount: packet capture

Learn more about NFS trunking

pNFS versus NFSv4.1 referrals

NFSv4.1 referrals provide a mode of initial mount path redirection that directs a client to the location of the

volumes upon a mount request. NFSv4.1 referrals work within a single SVM. This feature attempts to localize the NFS mount to a network interface residing on the same node as the data volume. If that interface or volume moves to another node while mounted to a client, then the data path is no longer localized until a new mount is established.

pNFS does not attempt to localize a mount path. Instead, it establishes a metadata server using a mount path and then localizes the data path dynamically as needed.

NFSv4.1 referrals can be used with pNFS, but the functionality is unnecessary. Enabling referrals with pNFS will not show noticeable results.

Enable or disable NFSv4 referrals

Interaction of pNFS with advanced capacity balancing

Advanced capacity balancing in ONTAP writes portions of file data across constituent volumes of a FlexGroup volume (not supported with single FlexVol volumes). As a file grows, ONTAP decides to begin writing data to a new multipart inode on a different constituent volume which might be on the same node or a different node. Writes, reads, and metadata operations to these multi-inode files are transparent and non-disruptive to clients. Advanced capacity balancing improves space management among the FlexGroup constituent volumes which provides for more consistent performance.

pNFS can redirect data IO to a localized network path depending on the file layout information stored in the NFS server. When a single large file is created in parts across multiple constituent volumes that can potentially span multiple nodes in the cluster, pNFS in ONTAP can still provide localized traffic to each file part because ONTAP maintains the file layout information for all of the file parts as well. When a file is read, the data path locality will change as needed.

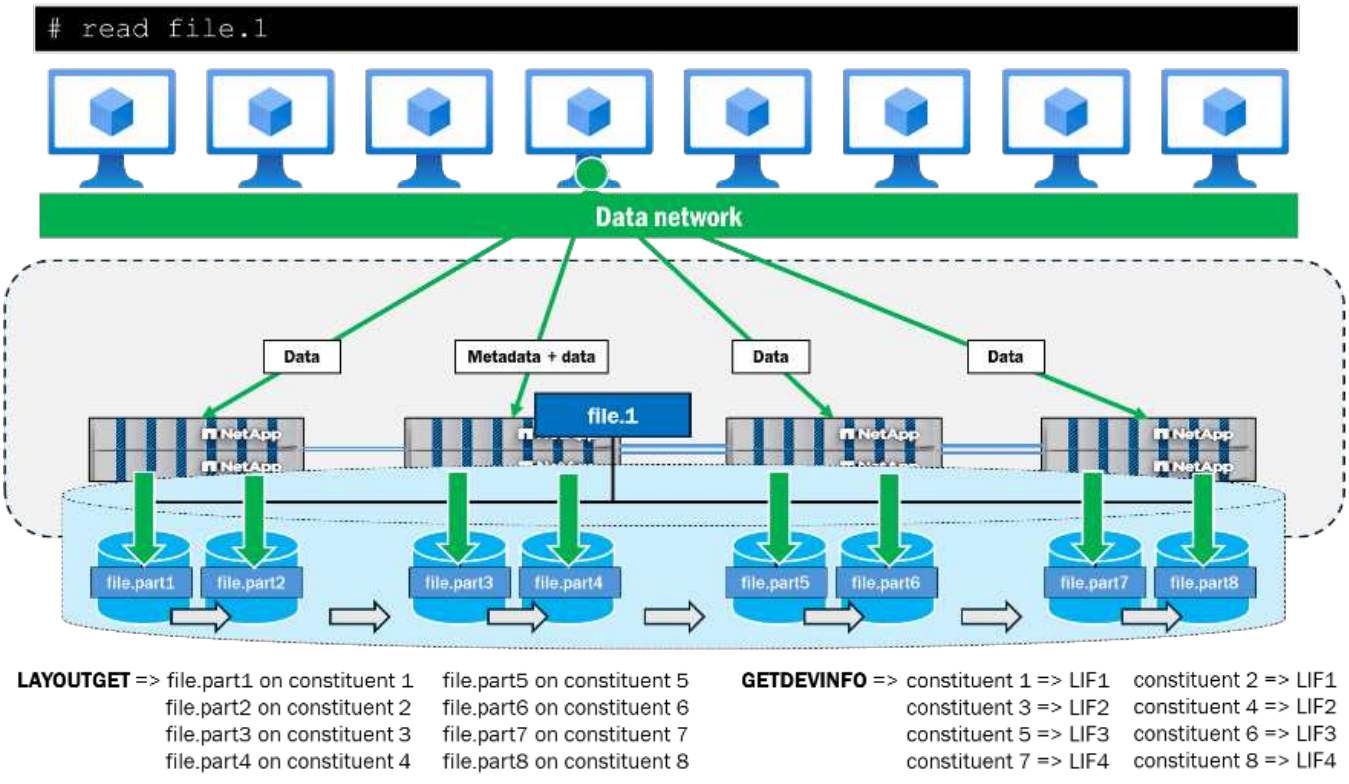


Figure 12. Advanced capacity balancing with pNFS

Related information

- FlexGroup volume configuration

pNFS deployment strategy in ONTAP

pNFS was introduced to improve upon traditional NFS by separating metadata and data paths, providing data localization, and enabling parallel operations.

Challenges of traditional NFS and benefits of pNFS

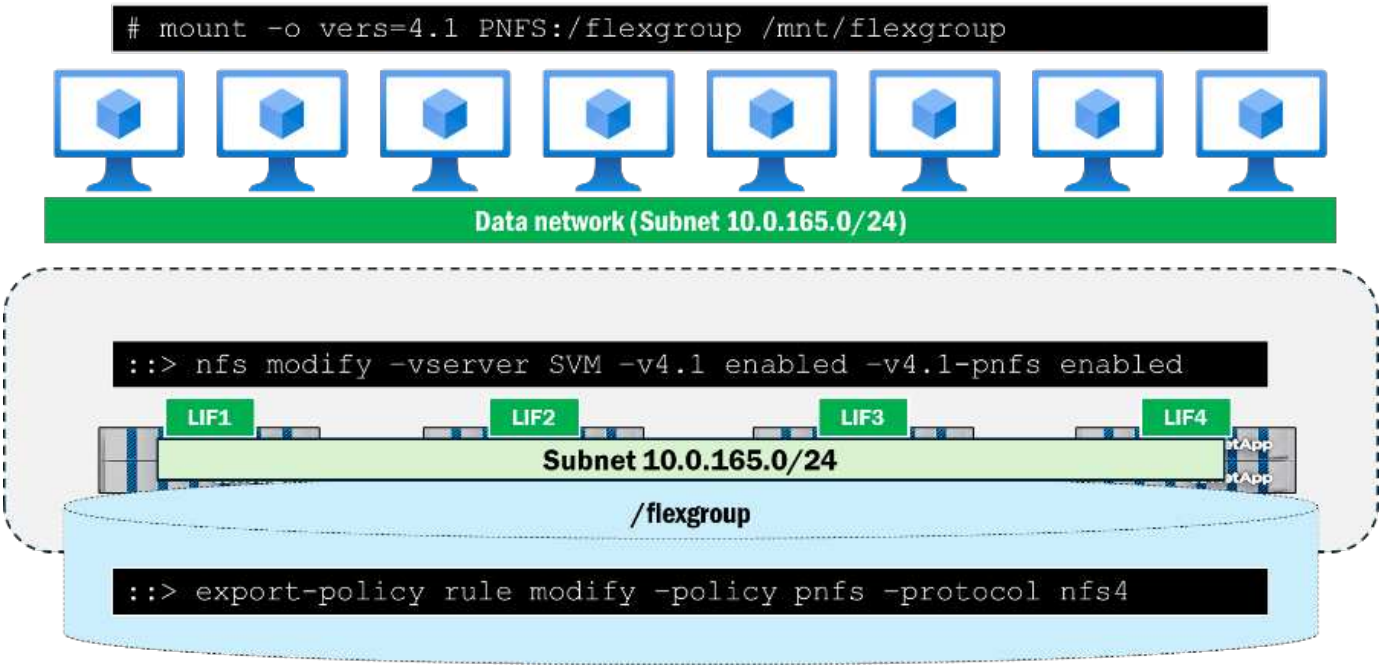
The following table shows the challenges of traditional NFS and explains how pNFS in ONTAP addresses them.

Challenge	pNFS benefit
Same path for metadata and data In traditional NFS, metadata and data traverse the same path, which can saturate both network and CPU, as a single path will attach to a single hardware node in the cluster. This is exacerbated when many users are attempting to access the same NFS export.	**Metadata and data paths are separate, data paths are parallelized** By separating the metadata and data paths for NFS traffic and providing multiple network paths for data paths, CPU and network resources are maximized in an ONTAP cluster, thus providing improved scale for workloads.
Workload distribution challenges In an ONTAP NAS cluster, you can have up to 24 nodes, each of which can have its own set of data volumes and network interfaces. Each volume can host its own workload, or a subset of a workload, and with a FlexGroup volume that workload can exist across multiple nodes that access a single namespace for simplicity. When a client mounts an NFS export, network traffic will be established on a single node. When the data being accessed resides on a separate node in the cluster, remote traffic will occur, which can add latency to a workload and complexity in administration.	**Local, parallel paths to data structures** Because pNFS splits out the data paths from metadata and provides multiple parallel data paths depending on the locality of the volume in the cluster, latency can be reduced by reducing the distance of network traffic in the cluster, as well as leveraging multiple hardware resources in a cluster. Also, since pNFS in ONTAP redirects data traffic automatically, administrators have less need to manage multiple export paths and locations.
Relocation of NFS mount points After a mount point is established, it would be disruptive to unmount and remount the volume. ONTAP offers the ability to migrate network interfaces between nodes, but that adds management overhead and is disruptive for stateful NFS connections using NFSv4.x. Some of the reasons for relocating a mount point are tied to the data locality challenges.	**Automatic path relocation** With pNFS, the NFS server maintains a table of the locations of network interfaces and volumes. When a data structure is requested from a client across the metadata path in pNFS, the server will deliver an optimized network path to the client, which will then use that path for data operations. This drastically reduces the management overhead for workloads and can improve performance in some cases.

Configuration requirements

Configuration of pNFS in NetApp ONTAP requires the following:

- An NFS client that supports pNFS and is mounted with NFSv4.1 or later
- NFSv4.1 enabled on the NFS server in ONTAP (`nfs modify -v4.1 enabled`; off by default)
- pNFS enabled in the NFS server in ONTAP (`nfs modify -v4.1-pnfs enabled`; disabled by default)
- At least one network interface per node, routable to the NFS clients
- Data volumes in the SVM that have export policies and rules that allow NFSv4



After the above configuration requirements are met, pNFS will simply work on its own.

Related information

- NFS configuration
- ONTAP support for NFSv4.1
- Network interface connectivity for pNFS

Plan

Plan for pNFS deployment

Before deploying pNFS in your environment, ensure you meet the prerequisites and understand the interoperability requirements and configuration limits.

Prerequisites

Before enabling and using pNFS in ONTAP, ensure the following requirements are met:

- NFSv4.1 or later is enabled on the NFS server

- At least one data LIF exists per node in the cluster for the SVM hosting the NFS server
- All data LIFs in the SVM are routable to NFS clients
- NFS clients support pNFS (most modern Linux distributions from 2014 and later)
- Network connectivity between clients and all data LIFs in the SVM is functional
- DNS resolution (if using hostnames) is configured properly for all data LIFs
- FlexGroup volumes are configured (recommended for best results)
- NFSv4.x ID domains match between clients and ONTAP
- NFS Kerberos (if used) is enabled on all data LIFs in the SVM

Best practices summary

When implementing pNFS in your environment, follow these best practices:

- Use FlexGroup volumes for best performance and capacity scaling
- Ensure all network interfaces in the SVM are routable to clients
- Disable NFSv4.0 to ensure clients use NFSv4.1 or later
- Spread mount points across multiple network interfaces and nodes
- Use round robin DNS for load balancing metadata servers
- Verify NFSv4.x ID domains match on clients and servers
- Conduct network interface migrations and storage failovers during maintenance windows
- Enable NFS Kerberos on all data LIFs if using Kerberos security
- Avoid using NFSv4.1 referrals when using pNFS
- Test nconnect settings carefully to avoid overwhelming TCP connection limits
- Consider session trunking as an alternative to nconnect (do not use both together)
- Verify client OS vendor support for pNFS before deployment

Interoperability

pNFS in ONTAP is designed to work with RFC-compliant NFS clients. The following considerations apply:

- Most modern Linux distributions from 2014 and later support pNFS (RHEL 6.4, Fedora 17, and later)
- Verify with your client OS vendor that pNFS is supported
- pNFS works with both FlexVol and FlexGroup volumes
- pNFS is supported with NFSv4.1 and NFSv4.2
- pNFS can be used with NFS Kerberos (krb5, krb5i, krb5p), but performance might be impacted
- pNFS can be used alongside nconnect or session trunking (but not both simultaneously)
- pNFS does not work over NFSv4.0

Limits

The following limits apply to pNFS in ONTAP:

- TCP connection limits per node vary by platform (check the NetApp Hardware Universe for specific limits)

- Maximum file size: Depends on the volume type and ONTAP version
- Maximum file count: Up to 200 billion files with FlexGroup volumes
- Maximum capacity: Up to 60 PB with FlexGroup volumes
- Network interface count: At least one data LIF per node is required; more might be needed for load balancing

When using nconnect with pNFS, be aware that TCP connection counts multiply quickly:

- Each client mount with nconnect creates multiple TCP connections per data LIF
- With many clients using high nconnect values, TCP connection limits can be exceeded
- Exceeding TCP connection limits prevents new connections until existing connections are freed

Related information

- Network interface connectivity for pNFS
- Enable or disable NFSv4.1
- ONTAP support for NFSv4.1
- ONTAP support for NFSv4.2
- NetApp Hardware Universe

pNFS tuning and performance best practices

When using pNFS in ONTAP, use these considerations and best practices for best results.

Volume type recommendations

pNFS in ONTAP works with both FlexVol volumes and FlexGroup volumes, but for the best overall results, use FlexGroup volumes.

FlexGroup volumes provide:

- A single mount point that can span multiple hardware resources in a cluster while allowing pNFS to localize data traffic
- Massive capacity possibilities (up to 60 PB) and high file counts (up to 200 billion files)
- Support for multipart files for capacity balancing and potential performance benefits
- Parallel access to volumes and hardware supporting a single workload

Learn about FlexGroup volumes management

Client recommendations

Not all NFS clients support pNFS, but most modern clients do. RHEL 6.4 and Fedora 17 were the first supported pNFS clients (roughly in 2014), so it is reasonable to assume that client versions released in the past few years fully support the feature. ONTAP's NFS support stance is one of "if the client supports the feature and is RFC compliant, and we support the feature, then the combination is supported." However, it is a best practice to ensure that pNFS is supported by the client OS vendor.

Volume moves

ONTAP provides the ability to nondisruptively move volumes across nodes or aggregates in the same cluster to provide capacity and performance balance flexibility. When a volume move takes place in ONTAP, the pNFS device mappings are automatically updated to inform clients to use the new volume-to-interface relationship if necessary.

Learn about moving a volume

Network interface migration

ONTAP provides the ability to move network interfaces across nodes in the same cluster to provide performance balance and maintenance flexibility. Like volume moves, when a network interface migration takes place in ONTAP, the pNFS device mappings are automatically updated to inform clients to use the new volume-to-interface relationship if necessary.

However, because NFSv4.1 is a stateful protocol, a network interface migration can be disruptive to clients that are actively using the NFS mount. It is a best practice to conduct network interface migrations in a maintenance window and notify clients of potential network disruptions.

Storage failovers/givebacks

pNFS follows the same storage failover considerations as NFSv4.1. These are covered in detail in NetApp Technical Report 4067: NFS Best Practice and Implementation Guide. In general, any storage failovers/givebacks involving pNFS should be done in a maintenance window, with potential storage disruptions expected due to the statefulness of the protocol.

Metadata workloads

Metadata operations are small in size and can be large in numbers depending on the workload (Are you creating a large number of files? Are you running "find" commands?) and total file count. As a result, workloads that are high in metadata calls can be taxing on the CPU of the NFS server and can potentially bottleneck over a single connection. pNFS (and NFSv4.x in general) is not suited for performance-dependent high metadata workloads, as the statefulness, the locking mechanisms, and some of the security features of the protocol version can negatively impact CPU utilization and latency. These workload types (such as high GETATTR or SETATTR) generally fare better with NFSv3.

Metadata server

The metadata server in pNFS is established at the initial mount of an NFS export. When the mount point is established, it remains in place until it is remounted or the data interface is moved. Because of this, it is a best practice to ensure that multiple clients accessing the same volume mount to different nodes and data interfaces across the SVM. This approach provides load balancing of the metadata servers across nodes and CPU resources while maximizing the network interfaces in the cluster. One way to accomplish this is to establish a round robin DNS setup, which is covered in NetApp Technical Report 4523: DNS Load Balancing in ONTAP.

NFSv4.x ID domains

NFSv4.x provides security functionality in many ways (covered in detail in NetApp Technical Report 4067: NFS Best Practice and Implementation Guide). NFSv4.x ID domains is one of those ways, where a client and server must agree on the ID domains when attempting to authenticate users and groups in an NFS export. One of the side effects of an ID domain mismatch would be the user or group showing up as an anonymized user (essentially squashed) to prevent unwanted access. With NFSv4.x (and also pNFS), it is a best practice to ensure the NFSv4.x ID domains match on client and server.

nconnect

As mentioned previously, nconnect in ONTAP can help improve performance in some workloads. With pNFS, it is important to understand that while nconnect can improve performance by greatly increasing the total number of TCP connections to the storage system, it can also create issues when many clients are leveraging the mount option by overwhelming the TCP connections on the storage. The NetApp Hardware Universe covers the TCP connection limits per node.

When a node's TCP connection limits are exceeded, no new TCP connections are allowed until existing connections are freed. This can create complications in environments that might experience mount storms.

The following table shows how pNFS with nconnect might overwhelm TCP connection limits:

Client count	nconnect value	Total potential TCP connections per mount, per node
1	4	4
100	4	400
1000	8	8000
10000	8	80000
10000	16	160000 [1]

[1] Exceeds most ONTAP single node TCP connection limits

NFSv4.1 session trunking

Session trunking in ONTAP can be used to increase throughput and path resiliency to NFSv4.x mounts. When used with pNFS, each node in a cluster can establish a session trunk. However, session trunks require at least two interfaces per node, and pNFS requires at least one interface per node to work as intended. Additionally, all interfaces in the SVM must be routable to the NFS clients. Session trunking and pNFS do not work properly when also leveraging nconnect. Consider nconnect and session trunking as mutually exclusive features.

Learn about NFS trunking

Network interface connectivity

pNFS requires a routable network interface on each node in a cluster to function properly. If other network interfaces that are not routable to NFS clients exist in the same SVM as the NFS server hosting pNFS, ONTAP will still advertise those interfaces in the device mapping to clients. When the NFS client attempts to access data via the interfaces in a different subnet, they will not be able to connect, and it will create an outage. It is a best practice to only allow network interfaces in an SVM that can be accessed by clients when using pNFS.

By default, pNFS requires any data LIF in the SVM to be routable to interfaces on the NFS clients because pNFS device lists will be populated with any data LIF in the SVM. As a result, non-routable data LIFs could be selected, which can create outage scenarios. As a best practice, only configure routable data LIFs when using pNFS.

Beginning in ONTAP 9.18.1 RC1 and later, you can specify which interfaces are eligible for pNFS traffic by subnet, allowing for mixing of routable and non-routable interfaces. Contact NetApp support for information on the commands.

NFSv4.0

NFSv4.0 is an option that can be enabled in an ONTAP NFS server alongside NFSv4.1. However, pNFS does not work over NFSv4.0. If NFSv4.0 is enabled in the NFS server, clients can potentially unknowingly mount that protocol version and will not be able to leverage pNFS. As a result, it is a best practice to explicitly disable NFSv4.0 when using pNFS. NFSv4.1 must still be enabled and can work independently of NFSv4.0.

NFSv4.1 referrals

NFSv4.1 referrals will localize the mount path from a client to the network interface on the node that owns a volume. pNFS localizes the data path, and the mount path becomes a metadata server.

While the two features can feasibly be used together, using NFSv4.1 referrals with pNFS might result in the undesired effect of stacking multiple metadata servers on the same node and reducing the ability to spread metadata servers across multiple cluster nodes. If metadata servers are not spread evenly across a cluster when using pNFS, then a single node's CPU can get overwhelmed with metadata requests and create a performance bottleneck.

As such, it is a best practice to avoid using NFSv4.1 referrals when using pNFS. Instead, spread the mount points across multiple network interfaces and nodes in the cluster.

Learn about enabling or disabling NFSv4 referrals

NFS Kerberos

With NFS Kerberos, it is possible to encrypt authentication with krb5 and to further encrypt data packets with krb5i and krb5p. This is enabled on a per-network interface basis in a SVM and is covered in full detail in NetApp Technical Report 4616: NFS Kerberos in ONTAP with Microsoft Active Directory.

Because pNFS can redirect data traffic across nodes and network interfaces in the SVM, NFS Kerberos must be enabled and functional on each network interface in the SVM. If any network interface in the SVM is not enabled for Kerberos, then pNFS will not be able to function properly when attempting to access data volumes on those interfaces.

For example, when running a read test using parallel dd on a pNFS-enabled SVM with two network interfaces (only one enabled for Kerberos), the files located on the Kerberos-enabled interface performed well, while the files on the node with the interface without Kerberos enabled never were able to complete their reads. When Kerberos was enabled on both interfaces, all files were able to perform as expected.

NFS Kerberos can be used with pNFS provided NFS Kerberos is enabled on all network interfaces in the SVM. Keep in mind that NFS Kerberos can incur a performance penalty due to encryption/decryption of the packets, so it is a best practice to test pNFS with NFS Kerberos thoroughly with your workloads to ensure that any performance hit is not overly impactful to the workload.

Below is an example of parallel read performance when using krb5 (authentication) and krb5p (end to end encryption) with pNFS on a RHEL 9.5 client. Krb5p saw a 70% performance degradation in this test.

Kerberos flavor	MB/s	Completion time
krb5	• File1–243 • File2–243 • File3–238 • File4–238	• File1–43 • File2–43.1 • File3–44 • File4–44.1

Kerberos flavor	MB/s	Completion time
krb5p	• File1–72.9 • File2–72.8 • File3–71.4 • File4–71.2	• File1–143.9 • File2–144.1 • File3–146.9 • File4–147.3

Learn about Kerberos with NFS for strong security

NFSv4.2

NFSv4.2 was added to ONTAP 9.8 and is the latest NFSv4.x version available (RFC-7862). NFSv4.2 does not have an explicit option to enable/disable it. Instead, it is enabled/disabled alongside NFSv4.1 (`-4.1 enabled`). If a client supports NFSv4.2, it will negotiate the highest supported version of NFS during the mount command if not specified otherwise with the `minorversion=2` mount option.

NFSv4.2 in ONTAP supports the following functionality:

- Security labels (MAC labels)
- Extended attributes
- Sparse file ops (FALLOCATE)

pNFS was introduced with NFSv4.1, but is also supported with NFSv4.2, as well as its accompanying features.

Learn about ONTAP support for NFSv4.2

pNFS commands, statistics and event logs

These ONTAP CLI commands pertain specifically to pNFS. You can use them to configure, troubleshoot, and gather statistics.

Enable NFSv4.1

```
nfs modify -vserver SVM -v4.1 enabled
```

Enable pNFS

```
nfs modify -vserver SVM -v4.1-pnfs enabled
```

Show pNFS devices (advanced privileges)

```
pnfs devices show -vserver SVM
```

```
Vserver Name     Mapping ID      Volume MSID     Mapping Status
Generation
---------------  --------------- --------------- ---------------
-------------
SVM              17              2157024470      notavailable    2
SVM              18              2157024463      notavailable    2
SVM              19              2157024469      available       3
SVM              20              2157024465      available       4
SVM              21              2157024467      available       3
SVM              22              2157024462      available       1
```

Show pNFS device mappings (advanced privileges)

```
pnfs devices mappings show -vserver SVM
```

```
Vserver Name    Mapping ID      Dsid            LIF IP
--------------  --------------- --------------- --------------------
SVM             19              2449            10.x.x.x
SVM             20              2512            10.x.x.y
SVM             21              2447            10.x.x.x
SVM             22              2442            10.x.x.y
```

Capture pNFS-specific performance counters (advanced privileges)

```
statistics start -object nfsv4_1 -vserver SVM -sample-id [optional-name]
```

View pNFS-specific performance counters (advanced privileges)

```
statistics show -object nfsv4_1 -vserver SVM
```

View list of pNFS-specific counters (advanced privileges)

```
statistics catalog counter show -object nfsv4_1 -counter *layout*|*device*
```

```
Object: nfsv4_1

Counter                     Description
--------------------------- ----------------------------------------------
getdeviceinfo_avg_latency   Average latency of NFSv4.1 GETDEVICEINFO
```

```
operations.
getdeviceinfo_error         The number of failed NFSv4.1 GETDEVICEINFO
operations.
getdeviceinfo_percent       Percentage of NFSv4.1 GETDEVICEINFO
operations.
getdeviceinfo_success       The number of successful NFSv4.1 GETDEVICEINFO
operations.
getdeviceinfo_total         Total number of NFSv4.1 GETDEVICEINFO
operations.
getdevicelist_avg_latency   Average latency of NFSv4.1 GETDEVICELIST
operations.
getdevicelist_error         The number of failed NFSv4.1 GETDEVICELIST
operations.
getdevicelist_percent       Percentage of NFSv4.1 GETDEVICELIST
operations.
getdevicelist_success       The number of successful NFSv4.1 GETDEVICELIST
operations.
getdevicelist_total         Total number of NFSv4.1 GETDEVICELIST
operations.
layoutcommit_avg_latency    Average latency of NFSv4.1 LAYOUTCOMMIT
operations.
layoutcommit_error          The number of failed NFSv4.1 LAYOUTCOMMIT
operations.
layoutcommit_percent        Percentage of NFSv4.1 LAYOUTCOMMIT operations.
layoutcommit_success        The number of successful NFSv4.1 LAYOUTCOMMIT
operations.
layoutcommit_total          Total number of NFSv4.1 LAYOUTCOMMIT
operations.
layoutget_avg_latency       Average latency of NFSv4.1 LAYOUTGET
operations.
layoutget_error             The number of failed NFSv4.1 LAYOUTGET
operations.
layoutget_percent           Percentage of NFSv4.1 LAYOUTGET operations.
layoutget_success           The number of successful NFSv4.1 LAYOUTGET
operations.
layoutget_total             Total number of NFSv4.1 LAYOUTGET operations.
layoutreturn_avg_latency    Average latency of NFSv4.1 LAYOUTRETURN
operations.
layoutreturn_error          The number of failed NFSv4.1 LAYOUTRETURN
operations.
layoutreturn_percent        Percentage of NFSv4.1 LAYOUTRETURN operations.
layoutreturn_success        The number of successful NFSv4.1 LAYOUTRETURN
operations.
layoutreturn_total          Total number of NFSv4.1 LAYOUTRETURN
operations.
```

View active network connections for NFS

You can verify if multiple TCP connections are being made to the SVM with the `network connections active show` command.

For instance, if you want to see NFS session trunks, look for connections from the same clients across different interfaces per node:

```
cluster::*> network connections active show -node cluster-0* -vserver PNFS
              Vserver   Interface         Remote
      CID Ctx Name      Name:Local Port   Host:Port
Protocol/Service
--------- --- --------- ----------------- --------------------
----------------
Node: node-01
2304333128 14 PNFS      data1:2049        ubuntu22-224:740     TCP/nfs
2304333144 10 PNFS      data3:2049        ubuntu22-224:864     TCP/nfs
2304333151  5 PNFS      data1:2049        ubuntu22-226:848     TCP/nfs
2304333167 15 PNFS      data3:2049        ubuntu22-226:684     TCP/nfs
Node: node-02
2497668321 12 PNFS      data2:2049        ubuntu22-224:963     TCP/nfs
2497668337 18 PNFS      data4:2049        ubuntu22-224:859     TCP/nfs
2497668344 14 PNFS      data2:2049        ubuntu22-226:675     TCP/nfs
2497668360  7 PNFS      data4:2049        ubuntu22-226:903     TCP/nfs
```

View NFS version information for connected clients

You can also view NFS connections with the `nfs connected-clients show` command. Keep in mind that the list of clients shown are clients that have had active NFS traffic in the past 48 hours. Idle NFS clients (even if still mounted) might not show up until the mount is accessed. You can filter these to show only more recently accessed clients by specifying the `-idle-time` feature.

For example, to see clients with activity in the past 10 minutes for the pNFS SVM:

```
cluster::*> nfs connected-clients show -vserver PNFS -idle-time <10m>

 Node: node-01

Vserver: PNFS Data-Ip: 10.x.x.x Local Remote Client-Ip Protocol Volume
Policy Idle-Time Reqs Reqs Trunking

10.x.x.a nfs4.2 PNFS_root default 9m 10s 0 149 false 10.x.x.a nfs4.2
FG_0001 default 9m 10s 135847 0 false 10.x.x.b nfs4.2 PNFS_root default 8m
12s 0 157 false 10.x.x.b nfs4.2 FG_0001 default 8m 12s 52111 0 false
```

Related information

- Learn about parallel NFS (pNFS) in ONTAP

Copyright information

Trademark information