Storage configuration on AFF/FAS systems

Enterprise applications

NetApp
January 12, 2026

Table of Contents

Storage configuration on AFF/FAS systems

FC SAN

LUN Alignment

LUN alignment refers to optimizing I/O with respect to the underlying file system layout.

On a ONTAP system, storage is organized in 4KB units. A database or file system 8KB block should map to exactly two 4KB blocks. If an error in LUN configuration shifts the alignment by 1KB in either direction, each 8KB block would exist on three different 4KB storage blocks rather than two. This arrangement would cause increased latency and cause additional I/O to be performed within the storage system.

Alignment also affects LVM architectures. If a physical volume within a logical volume group is defined on the whole drive device (no partitions are created), the first 4KB block on the LUN aligns with the first 4KB block on the storage system. This is a correct alignment. Problems arise with partitions because they shift the starting location where the OS uses the LUN. As long as the offset is shifted in whole units of 4KB, the LUN is aligned.

In Linux environments, build logical volume groups on the whole drive device. When a partition is required, check alignment by running `fdisk -u` and verifying that the start of each partition is a multiple of eight. This means that the partition starts at a multiple of eight 512-byte sectors, which is 4KB.

Also see the discussion about compression block alignment in the section Efficiency. Any layout that is aligned with 8KB compression block boundaries is also aligned with 4KB boundaries.

Misalignment warnings

Database redo/transaction logging normally generates unaligned I/O that can cause misleading warnings about misaligned LUNs on ONTAP.

Logging performs a sequential write of the log file with writes of varying size. A log write operation that does not align to 4KB boundaries does not ordinarily cause performance problems because the next log write operation completes the block. The result is that ONTAP is able to process almost all writes as complete 4KB blocks, even though the data in some 4KB blocks was written in two separate operations.

Verify alignment by using by using utilities such as `sio` or `dd` that can generate I/O at a defined block size. The I/O alignment statistics on the storage system can be viewed with the `stats` command. See WAFL alignment verification for more information.

Alignment in Solaris environments is more complicated. Refer to ONTAP SAN Host Configuration for more information.

Caution
In Solaris x86 environments, take additional care about proper alignment because most configurations have several layers of partitions. Solaris x86 partition slices usually exist on top of a standard master boot record partition table.

LUN sizing and LUN count

Selecting the optimal LUN size and the number of LUNs to be used is critical for optimal performance and manageability of Oracle databases.

A LUN is a virtualized object on ONTAP that exists across all of the drives in the hosting aggregate. As a result, the performance of the LUN is unaffected by its size because the LUN draws on the full performance potential of the aggregate no matter which size is chosen.

As a matter of convenience, customers might wish to use a LUN of a particular size. For example, if a database is built on an LVM or Oracle ASM diskgroup composed of two LUNs of 1TB each, then that diskgroup must be grown in increments of 1TB. It might be preferable to build the diskgroup from eight LUNs of 500GB each so that the diskgroup can be increased in smaller increments.

The practice of establishing a universal standard LUN size is discouraged because doing so can complicate manageability. For example, a standard LUN size of 100GB might work well when a database or datastore is in the range of 1TB to 2TB, but a database or datastore of 20TB in size would require 200 LUNs. This means that server reboot times are longer, there are more objects to manage in the various UIs, and products such as SnapCenter must perform discovery on many objects. Using fewer, larger LUNs avoids such problems.

- The LUN count is more important than the LUN size.
- LUN size is mostly controlled by LUN count requirements.
- Avoid creating more LUNs than required.

LUN count

Unlike the LUN size, the LUN count does affect performance. Application performance often depends on the ability to perform parallel I/O through the SCSI layer. As a result, two LUNs offer better performance than a single LUN. Using an LVM such as Veritas VxVM, Linux LVM2, or Oracle ASM is the simplest method to increase parallelism.

NetApp customers have generally experienced minimal benefit from increasing the number of LUNs beyond sixteen, although the testing of 100%-SSD environments with very heavy random I/O has demonstrated further improvement up to 64 LUNs.

NetApp recommends the following:

In general, four to sixteen LUNs are sufficient to support the I/O needs of any given database workload. Less than four LUNs might create performance limitations because of limitations in host SCSI implementations.

LUN placement

Optimal placement of database LUNs within ONTAP volumes primarily depends on how various ONTAP features will be used.

Volumes

One common point of confusion with customers new to ONTAP is the use of FlexVols, commonly referred to as simply "volumes".

A volume is not a LUN. These terms are used synonymously with many other vendor products, including cloud providers. ONTAP volumes are simply management containers. They do not serve data by themselves, nor do they occupy space. They are containers for files or LUNs and exist to improve and simplify manageability, especially at scale.

Volumes and LUNs

Related LUNs are normally co-located in a single volume. For example, a database that requires 10 LUNs would typically have all 10 LUNs placed on the same volume.

> - Using a 1:1 ratio of LUNs to volumes, meaning one LUN per volume, is **not** a formal best practice.
> - Instead, volumes should be viewed as containers for workloads or datasets. There may be a single LUN per volume, or there could be many. The right answer depends on manageability requirements.
> - Scattering LUNs across an unnecessary number of volumes can lead to additional overhead and scheduling problems for operations such as snapshot operations, excessive numbers of objects displayed in the UI, and result in reaching platform volume limits before the LUN limit is reached.

Volumes, LUNs, and snapshots

Snapshot policies and schedules are placed on the volume, not the LUN. A dataset that consists of 10 LUNs would require only a single snapshot policy when those LUNs are co-located in the same volume.

Additionally, co-locating all related LUNs for a given dataset in a single volume delivers atomic snapshot operations. For example, a database that resided on 10 LUNs, or a VMware-based application environment consisting of 10 different OSs could be protected as a single, consistent object if the underlying LUNs are all placed on a single volume. If they are placed on different volumes, the snapshots may or may not be 100% in sync, even if scheduled at the same time.

In some cases, a related set of LUNs might need to be split into two different volumes because of recovery requirements. For example, a database might have four LUNs for datafiles and two LUNs for logs. In this case, a datafile volume with 4 LUNs and a log volume with 2 LUNs might be the best option. The reason is independent recoverability. For example, the datafile volume could be selectively restored to an earlier state, meaning all four LUNs would be reverted to the state of the snapshot, while the log volume with its critical data would be unaffected.

Volumes, LUNs, and SnapMirror

SnapMirror policies and operations are, like snapshot operations, performed on the volume, not the LUN.

Co-locating related LUNs in a single volume allows you to create a single SnapMirror relationship and update all contained data with a single update. As with snapshots, the update will also be an atomic operation. The SnapMirror destination would be guaranteed to have a single point-in-time replica of the source LUNs. If the LUNs were spread across multiple volumes, the replicas may or may not be consistent with one another.

Volumes, LUNs, and QoS

While QoS can be selectively applied to individual LUNs, it is usually easier to set it at the volume level. For example, all of the LUNs used by the guests in a given ESX server could be placed on a single volume, and then an ONTAP adaptive QoS policy could be applied. The result is a self-scaling IOPS-per-TB limit that applies to all LUNs.

Likewise, if a database required 100K IOPS and occupied 10 LUNs, it would be easier to set a single 100K IOPS limit on a single volume than to set 10 individual 10K IOPS limits, one on each LUN.

Multi-volume layouts

There are some cases where distributing LUNs across multiple volumes may be beneficial. The primary reason is controller striping. For example, an HA storage system might be hosting a single database where the full processing and caching potential of each controller is required. In this case, a typical design would be to place half of the LUNs in a single volume on controller 1, and the other half of the LUNs in a single volume on controller 2.

Similarly, controller striping might be used for load balancing. An HA system that hosted 100 databases of 10 LUNs each might be designed where each database receives a 5-LUN volume on each of the two controllers. The result is guaranteed symmetric loading of each controller as additional databases are provisioned.

None of these examples involve a 1:1 volume to LUN ratio, though. The goal remains to optimize manageability by co-locating related LUNs in volumes.

One example where a 1:1 LUN to volume ratio makes sense is containerization, where each LUN might really represent a single workload and need to be each managed on an individual basis. In such cases, a 1:1 ratio may be optimal.

LUN resizing and LVM resizing

When a SAN-based file system has reached its capacity limit, there are two options for increasing the space available:

- Increase the size of the LUNs
- Add a LUN to an existing volume group and grow the contained logical volume

Although LUN resizing is an option to increase capacity, it is generally better to use an LVM, including Oracle ASM. One of the principal reasons LVMs exist is to avoid the need for a LUN resize. With an LVM, multiple LUNs are bonded together into a virtual pool of storage. The logical volumes carved out of this pool are managed by the LVM and can be easily resized. An additional benefit is the avoidance of hotspots on a particular drive by distributing a given logical volume across all available LUNs. Transparent migration can usually be performed by using the volume manager to relocate the underlying extents of a logical volume to new LUNs.

LVM striping

LVM striping refers to distributing data across multiple LUNs. The result is dramatically improved performance for many databases.

Before the era of flash drives, striping was used to help overcome the performance limitations of spinning drives. For example, if an OS needs to perform a 1MB read operation, reading that 1MB of data from a single drive would require a lot of drive head seeking and reading as the 1MB is slowly transferred. If that 1MB of data was striped across 8 LUNs, the OS could issue eight 128K read operations in parallel and reduce the time required to complete the 1MB transfer.

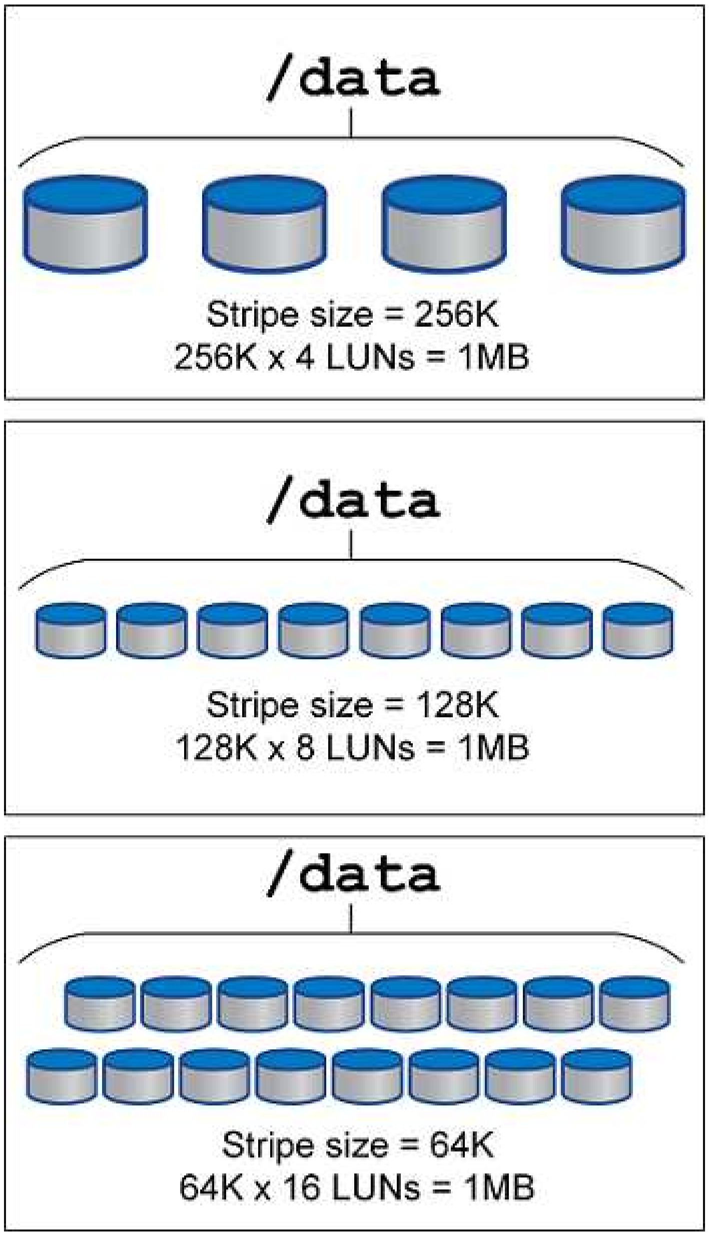
Striping with spinning drives was more difficult because the I/O pattern had to be known in advance. If the striping wasn't correctly tuned for the true I/O patterns, striped configurations could damage performance. With Oracle databases, and especially with all-flash configurations, striping is much easier to configure and has been proven to dramatically improve performance.

Logical volume managers such as Oracle ASM stripe by default, but native OS LVM do not. Some of them bond multiple LUNs together as a concatenated device, which results in datafiles that exist on one and only one LUN device. This causes hot spots. Other LVM implementations default to distributed extents. This is similar to striping, but it’s coarser. The LUNs in the volume group are sliced into large pieces, called extents and typically measured in many megabytes, and the logical volumes are then distributed across those extents. The result is random I/O against a file should be well distributed across LUNs, but sequential I/O operations are not as efficient as they could be.

Performance-intensive application I/O is nearly always either (a) in units of the basic block size or (b) one megabyte.

The primary goal of a striped configuration is to ensure that single-file I/O can be performed as a single unit, and multiblock I/Os, which should be 1MB in size, can be parallelized evenly across all LUNs in the striped volume. This means that the stripe size must not be smaller than the database block size, and the stripe size multiplied by the number of LUNs should be 1MB.

The following figure shows three possible options for stripe size and width tuning. The number of LUNs is selected to meet performance requirements as described above, but in all cases the total data within a single stripe is 1MB.



NFS

Overview

NetApp has been providing enterprise-grade NFS storage for over 30 years, and its use is growing with the push toward cloud-based infrastructures because of it's simplicity.

The NFS protocol includes multiple versions with varying requirements. For a complete description of NFS configuration with ONTAP, please see TR-4067 NFS on ONTAP Best Practices. The following sections cover some of the more critical requirements and common user errors.

NFS versions

The operating system NFS client must be supported by NetApp.

- NFSv3 is supported with OSs that follow the NFSv3 standard.
- NFSv3 is supported with the Oracle dNFS client.
- NFSv4 is supported with all OSs that follow the NFSv4 standard.
- NFSv4.1 and NFSv4.2 require specific OS support. Consult the NetApp IMT for supported OSs.
- Oracle dNFS support for NFSv4.1 requires Oracle 12.2.0.2 or higher.

The NetApp support matrix for NFSv3 and NFSv4 does not include specific operating systems. All OSs that obey the RFC are generally supported. When searching the online IMT for NFSv3 or NFSv4 support, do not select a specific OS because there will be no matches displayed. All OSs are implicitly supported by the general policy.

Linux NFSv3 TCP slot tables

TCP slot tables are the NFSv3 equivalent of host bus adapter (HBA) queue depth. These tables control the number of NFS operations that can be outstanding at any one time. The default value is usually 16, which is far too low for optimum performance. The opposite problem occurs on newer Linux kernels, which can automatically increase the TCP slot table limit to a level that saturates the NFS server with requests.

For optimum performance and to prevent performance problems, adjust the kernel parameters that control the TCP slot tables.

Run the `sysctl -a | grep tcp.*.slot_table` command, and observe the following parameters:

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

All Linux systems should include `sunrpc.tcp_slot_table_entries`, but only some include `sunrpc.tcp_max_slot_table_entries`. They should both be set to 128.

Failure to set these parameters may have significant effects on performance. In some cases, performance is limited because the linux OS is not issuing sufficient I/O. In other cases, I/O latencies increases as the linux OS attempts to issue more I/O than can be serviced.

ADR and NFS

Some customers have reported performance problems resulting from an excessive amount of I/O on data in the `ADR` location. The problem does not generally occur until a lot of performance data has accumulated. The reason for the excessive I/O is unknown, but this problem appears to be a result of Oracle processes repeatedly scanning the target directory for changes.

Removal of the `noac` and/or `actimeo=0` mount options allows host OS caching to occur and reduces storage I/O levels.

NetApp recommends to not place `ADR` data on a file system with `noac` or `actimeo=0` because performance problems are likely. Separate `ADR` data into a different mount point if necessary.

nfs-rootonly and mount-rootonly

ONTAP includes an NFS option called `nfs-rootonly` that controls whether the server accepts NFS traffic connections from high ports. As a security measure, only the root user is permitted to open TCP/IP connections using a source port below 1024 because such ports are normally reserved for OS use, not user processes. This restriction helps ensure that NFS traffic is from an actual operating system NFS client, and not a malicious process emulating an NFS client. The Oracle dNFS client is a userspace driver, but the process runs as root, so it is generally not required to change the value of `nfs-rootonly`. The connections is made from low ports.

The `mount-rootonly` option only applies to NFSv3. It controls whether the RPC MOUNT call be accepted from ports greater than 1024. When dNFS is used, the client is again running as root, so it able to open ports below 1024. This parameter has no effect.

Processes opening connections with dNFS over NFS versions 4.0 and higher do not run as root and therefore require ports over 1024. The `nfs-rootonly` parameter must be set to disabled for dNFS to complete the connection.

If `nfs-rootonly` is enabled, the result is a hang during the mount phase opening dNFS connections. The sqlplus output looks similar to this:

```
SQL>startup
ORACLE instance started.
Total System Global Area 4294963272 bytes
Fixed Size                  8904776 bytes
Variable Size             822083584 bytes
Database Buffers         3456106496 bytes
Redo Buffers                7868416 bytes
```

The parameter can be changed as follows:

```
Cluster01::> nfs server modify -nfs-rootonly disabled
```

In rare situations, you might need to change both nfs-rootonly and mount-rootonly to disabled. If a server is managing an extremely large number of TCP connections, it is possible that no ports below 1024 is available, and the OS is forced to use higher ports. These two ONTAP parameters would need to be changed to allow the connection to complete.

NFS export polices: superuser and setuid

If Oracle binaries are located on an NFS share, the export policy must include superuser and setuid permissions.

Shared NFS exports used for generic file services such as user home directories usually squash the root user. This means a request from the root user on a host that has mounted a filesystem is remapped as a different user with lower privileges. This helps secure data by preventing a root user on a particular server from accessing data on the shared server. The setuid bit can also be a security risk on a shared environment. The setuid bit allows a process to be run as a different user than the user invoking the command. For example, a shell script that was owned by root with the setuid bit runs as root. If that shell script could be changed by other users, any non-root user could issue a command as root by updating the script.

The Oracle binaries include files owned by root and use the setuid bit. If Oracle binaries are installed on an NFS share, the export policy must include the appropriate superuser and setuid permissions. In the example below, the rule includes both `allow-suid` and permits `superuser` (root) access for NFS clients using system authentication.

```
Cluster01::> export-policy rule show -vserver vserver1 -policyname orabin
-fields allow-suid,superuser
vserver   policyname ruleindex superuser allow-suid
--------- ---------- --------- --------- ----------
vserver1  orabin     1         sys       true
```

NFSv4/4.1 configuration

For most applications, there is very little difference between NFSv3 and NFSv4. Application I/O is usually very simple I/O and does not benefit significantly from some of the advanced features available in NFSv4. Higher versions of NFS should not be viewed as an “upgrade” from a database storage perspective, but instead as versions of NFS that include additional features. For example, if the end-to-end security of kerberos privacy mode (krb5p) is required, then NFSv4 is required.

NetApp recommends using NFSv4.1 if NFSv4 capabilities are required. There are some functional enhancements to the NFSv4 protocol in NFSv4.1 that improve resiliency in certain edge cases.

Switching to NFSv4 is more complicated than simply changing the mount options from vers=3 to vers=4.1. A more complete explanation of NFSv4 configuration with ONTAP, including guidance on configuring the OS, see TR-4067 NFS on ONTAP best practices. The following sections of this TR explain some of the basic requirements for using NFSv4.

NFSv4 domain

A complete explanation of NFSv4/4.1 configuration is beyond the scope of this document, but one commonly encountered problem is a mismatch in domain mapping. From a sysadmin point of view, the NFS file systems appear to behave normally, but applications report errors about permissions and/or setuid on the certain files.

In some cases, administrators have incorrectly concluded that the permissions of the application binaries have been damaged and have run chown or chmod commands when the actual problem was the domain name.

The NFSv4 domain name is set on the ONTAP SVM:

```
Cluster01::> nfs server show -fields v4-id-domain
vserver   v4-id-domain
--------- ------------
vserver1  my.lab
```

The NFSv4 domain name on the host is set in `/etc/idmap.cfg`

```
[root@host1 etc]# head /etc/idmapd.conf
[General]
#Verbosity = 0
# The following should be set to the local NFSv4 domain name
# The default is the host's DNS domain name.
Domain = my.lab
```

The domain names must match. If they do not, mapping errors similar to the following appear in `/var/log/messages`:

```
Apr 12 11:43:08 host1 nfsidmap[16298]: nss_getpwnam: name 'root@my.lab'
does not map into domain 'default.com'
```

Application binaries, such as Oracle database binaries, include files owned by root with the setuid bit, which means a mismatch in the NFSv4 domain names causes failures with Oracle startup and a warning about the ownership or permissions of a file called `oradism`, which is located in the `$ORACLE_HOME/bin` directory. It should appear as follows:

```
[root@host1 etc]# ls -l /orabin/product/19.3.0.0/dbhome_1/bin/oradism
-rwsr-x--- 1 root oinstall 147848 Apr 17  2019
/orabin/product/19.3.0.0/dbhome_1/bin/oradism
```

If this file appears with ownership of nobody, there may be an NFSv4 domain mapping problem.

```
[root@host1 bin]# ls -l oradism
-rwsr-x--- 1 nobody oinstall 147848 Apr 17  2019 oradism
```

To fix this, check the `/etc/idmap.cfg` file against the v4-id-domain setting on ONTAP and ensure they are consistent. If they are not, make the required changes, run `nfsidmap -c`, and wait a moment for the changes to propagate. The file ownership should then be properly recognized as root. If a user had attempted to run `chown root` on this file before the NFS domains configure was corrected, it might be necessary to run `chown` `root` again.

Oracle direct NFS (dNFS)

Oracle databases can use NFS in two ways.

First, it can use a filesystem mounted using the native NFS client that is part of the operating system. This is sometimes called kernel NFS, or kNFS. The NFS filesystem is mounted and used by the Oracle database exactly the same as any other application would use an NFS filesystem.

The second method is Oracle Direct NFS (dNFS). This is an implementation of the NFS standard within the Oracle database software. It does not change the way Oracle databases are configured or managed by the DBA. As long as the storage system itself has the correct settings, the use of dNFS should be transparent to the DBA team and end users.

A database with the dNFS feature enabled still has the usual NFS filesystems mounted. Once the database is open, the Oracle database opens a set of TCP/IP sessions and performs NFS operations directly.

Direct NFS

The primary value of Oracle's Direct NFS is to bypass the host NFS client and perform NFS file operations directly on an NFS server. Enabling it only requires changing the Oracle Disk Manager (ODM) library. Instructions for this process are provided in the Oracle documentation.

Using dNFS results in a significant improvement in I/O performance and decreases the load on the host and the storage system because I/O is performed in the most efficient way possible.

In addition, Oracle dNFS includes an **option** for network interface multipathing and fault-tolerance. For example, two 10Gb interfaces can be bound together to offer 20Gb of bandwidth. A failure of one interface results in I/O being retried on the other interface. The overall operation is very similar to FC multipathing. Multipathing was common years ago when 1Gb ethernet was the most common standard. A 10Gb NIC is sufficient for most Oracle workloads, but if more is required 10Gb NICs can be bonded.

When dNFS is used, it is critical that all patches described in Oracle Doc 1495104.1 are installed. If a patch cannot be installed, the environment must be evaluated to make sure that the bugs described in that document do not cause problems. In some cases, an inability to install the required patches prevents the use of dNFS.

Do not use dNFS with any type of round-robin name resolution, including DNS, DDNS, NIS or any other method. This includes the DNS load balancing feature available in ONTAP. When an Oracle database using dNFS resolves a host name to an IP address it must not change on subsequent lookups. This can result in Oracle database crashes and possible data corruption.

Enabling dNFS

Oracle dNFS can work with NFSv3 with no configuration required beyond enabling the dNFS library (See Oracle documentation for the specific command required) but if dNFS is unable to establish connectivity, it can silently revert back to the kernel NFS client. If this happens, performance can be severely affected.

If you wish to use dNFS multiplexing across multiple interface, with NFSv4.X, or use encryption, you must configure an oranfstab file. The syntax is extremely strict. Small errors in the file can result in startup hanging or bypassing the oranfstab file.

At the time of writing, dNFS multipathing does not work with NFSv4.1 with recent versions of Oracle Database. An oranfstab file that specifies NFSv4.1 as a protocol can only use a single path statement for a given export. The reason is ONTAP does not support clientID trunking. Oracle Database patches to resolve this limitation may be available in the future.

The only way to be certain dNFS is operating as expected is to query the v$dnfs tables.

Below is a sample oranfstab file located at /etc. This is one of multiple locations an oranfstab file can be placed.

```
[root@jfs11 trace]# cat /etc/oranfstab
server: NFSv3test
path: jfs_svmdr-nfs1
path: jfs_svmdr-nfs2
export: /dbf mount: /oradata
export: /logs mount: /logs
nfs_version: NFSv3
```

The first step is to check that dNFS is operational for the specified filesystems:

```
SQL> select dirname,nfsversion from v$dnfs_servers;

DIRNAME
------------------------------------
NFSVERSION
----------------
/logs
NFSv3.0

/dbf
NFSv3.0
```

This output indicates that dNFS is in use with these two filesystems, but it does **not** mean that oranfstab is operational. If an error was present, dNFS would have autodiscovered the host’s NFS filesystems and you may still see the same output from this command.

Multipathing can be checked as follows:

```
SQL> select svrname,path,ch_id from v$dnfs_channels;

SVRNAME
------------------------------------
PATH
------------------------------------
     CH_ID
----------
NFSv3test
jfs_svmdr-nfs1
         0
```

```
NFSv3test
jfs_svmdr-nfs2
         1

SVRNAME
------------------------------------
PATH
------------------------------------
     CH_ID
----------

NFSv3test
jfs_svmdr-nfs1
         0

NFSv3test
jfs_svmdr-nfs2

[output truncated]

SVRNAME
------------------------------------
PATH
------------------------------------
     CH_ID
----------
NFSv3test
jfs_svmdr-nfs2
         1

NFSv3test
jfs_svmdr-nfs1
         0

SVRNAME
------------------------------------
PATH
------------------------------------
     CH_ID
----------

NFSv3test
jfs_svmdr-nfs2
         1
```

```
66 rows selected.
```

These are the connections that dNFS is using. Two paths and channels are visible for each SVRNAME entry. This means multipathing is working, which means the oranfstab file was recognized and processed.

Direct NFS and host file system access

Using dNFS can occasionally cause problems for applications or user activities that rely on the visible file systems mounted on the host because the dNFS client accesses the file system out of band from the host OS. The dNFS client can create, delete, and modify files without the knowledge of the OS.

When the mount options for single-instance databases are used, they enable caching of file and directory attributes, which also means that the contents of a directory are cached. Therefore, dNFS can create a file, and there is a short lag before the OS rereads the directory contents and the file becomes visible to the user. This is not generally a problem, but, on rare occasions, utilities such as SAP BR*Tools might have issues. If this happens, address the problem by changing the mount options to use the recommendations for Oracle RAC. This change results in the disabling of all host caching.

Only change mount options when (a) dNFS is used and (b) a problem results from a lag in file visibility. If dNFS is not in use, using Oracle RAC mount options on a single-instance database results in degraded performance.

See the note about `nosharecache` in Linux NFS mount options for a Linux-specific dNFS issue that can produce unusual results.

NFS leases and locks

NFSv3 is stateless. That effectively means that the NFS server (ONTAP) doesn’t keep track of which file systems are mounted, by whom, or which locks are truly in place.

ONTAP does have some features that will record mount attempts so you have an idea which clients may be accessing data, and there may be advisory locks present, but that information isn’t guaranteed to be 100% complete. It can’t be complete, because tracking NFS client state is not part of the NFSv3 standard.

NFSv4 statefulness

In contrast, NFSv4 is stateful. The NFSv4 server tracks which clients are using which file systems, which files exist, which files and/or regions of files are locked, etc. This means there needs to be regular communication between an NFSv4 server to keep the state data current.

The most important states being managed by the NFS server are NFSv4 Locks and NFSv4 Leases, and they are very much intertwined. You need to understand how each works by itself, and how they relate to one another.

NFSv4 locks

With NFSv3, locks are advisory. An NFS client can still modify or delete a "locked" file. An NFSv3 lock doesn’t expire by itself, it must be removed. This creates problems. For example, if you have a clustered application that creates NFSv3 locks, and one of the nodes fails, what do you do? You can code the application on the surviving nodes to remove the locks, but how do you know that’s safe? Maybe the "failed" node is operational, but isn’t communicating with the rest of the cluster?

With NFSv4, locks have a limited duration. As long as the client holding the locks continues to check in with the

NFSv4 server, no other client is permitted to acquire those locks. If a client fails to check in with the NFSv4, the locks eventually get revoked by the server and other clients will be able to request and obtain locks.

NFsv4 leases

NFSv4 locks are associated with an NFSv4 lease. When an NFSv4 client establishes a connection with an NFSv4 server, it gets a lease. If the client obtains a lock (there are many types of locks) then the lock is associated with the lease.

This lease has a defined timeout. By default, ONTAP will set the timeout value to 30 seconds:

```
Cluster01::*> nfs server show -vserver vserver1 -fields v4-lease-seconds

vserver   v4-lease-seconds
--------- ----------------
vserver1  30
```

This means that an NFSv4 client needs to check in with the NFSv4 server every 30 seconds to renew its leases.

The lease is automatically renewed by any activity, so if the client is doing work there’s no need to perform addition operations. If an application becomes quiet and is not doing real work, it’s going to need to perform a sort of keep-alive operation (called a SEQUENCE) instead. It’s essentially just saying "I’m still here, please refresh my leases."

```
*Question:* What happens if you lose network connectivity for 31 seconds?
```

NFSv3 is stateless. It’s not expecting communication from the clients. NFSv4 is stateful, and once that lease period elapses, the lease expires, and locks are revoked and the locked files are made available to other clients.

With NFSv3, you could move network cables around, reboot network switches, make configuration changes, and be fairly sure that nothing bad would happen. Applications would normally just wait patiently for the network connection to work again.

With NFSv4, you have 30 seconds (unless you’ve increased the value of that parameter within ONTAP) to complete your work. If you exceed that, your leases time out. Normally this results in application crashes.

As an example, if you have an Oracle database, and you experience a loss of network connectivity (sometimes called a "network partition") that exceeds the lease timeout, you will crash the database.

Here’s an example of what happens in the Oracle alert log if this happens:

```
2022-10-11T15:52:55.206231-04:00
Errors in file /orabin/diag/rdbms/ntap/NTAP/trace/NTAP_ckpt_25444.trc:
ORA-00202: control file: '/redo0/NTAP/ctrl/control01.ctl'
ORA-27072: File I/O error
Linux-x86_64 Error: 5: Input/output error
Additional information: 4
Additional information: 1
Additional information: 4294967295
2022-10-11T15:52:59.842508-04:00
Errors in file /orabin/diag/rdbms/ntap/NTAP/trace/NTAP_ckpt_25444.trc:
ORA-00206: error in writing (block 3, # blocks 1) of control file
ORA-00202: control file: '/redo1/NTAP/ctrl/control02.ctl'
ORA-27061: waiting for async I/Os failed
```

If you look at the syslogs, you should see several of these errors:

```
Oct 11 15:52:55 host1 kernel: NFS: nfs4_reclaim_open_state: Lock reclaim
failed!
Oct 11 15:52:55 host1 kernel: NFS: nfs4_reclaim_open_state: Lock reclaim
failed!
Oct 11 15:52:55 host1 kernel: NFS: nfs4_reclaim_open_state: Lock reclaim
failed!
```

The log messages are usually the first sign of a problem, other than the application freeze. Typically, you see nothing at all during the network outage because processes and the OS itself are blocked attempting to access the NFS file system.

The errors appear after the network is operational again. In the example above, once connectivity was reestablished, the OS attempted to reacquire the locks, but it was too late. The lease had expired and the locks were removed. That results in an error that propagates up to the Oracle layer and causes the message in the alert log. You might see variations on these patterns depending on the version and configuration of the database.

In summary, NFSv3 tolerates network interruption, but NFSv4 is more sensitive and imposes a defined lease period.

What if a 30 second timeout isn't acceptable? What if you manage a dynamically changing network where switches are rebooted or cables are relocated and the result is the occasional network interruption? You could choose to extend the lease period, but whether you want to do that requires an explanation of NFSv4 grace periods.

NFSv4 grace periods

If an NFSv3 server is rebooted, it's ready to serve IO almost instantly. It was not maintaining any sort of state about clients. The result is that an ONTAP takeover operation often appears to be close to instantaneous. The moment a controller is ready to start serving data it will send an ARP to the network that signals the change in topology. Clients normally detect this almost instantly and data resumes flowing.

NFSv4, however, will produce a brief pause. It’s just part of how NFSv4 works.

The following sections are current as of ONTAP 9.15.1, but the lease and lock behavior as well as tuning options can change from version to version. If you need to tune NFSv4 lease/lock timeouts, please consult NetApp support for the latest information.

NFSv4 servers need to track the leases, locks, and who’s using what data. If an NFS server panics and reboots, or loses power for a moment, or is restarted during maintenance activity, the result is the lease/lock and other client information is lost. The server needs to figure out which client is using what data before resuming operations. This is where the grace period comes in.

If you suddenly power cycle your NFSv4 server. When it comes back up, clients that attempt to resume IO will get a response that essentially says, "I have lost lease/lock information. Would you like to re-register your locks?" That’s the start of the grace period. It defaults to 45 seconds on ONTAP:

```
Cluster01::> nfs server show -vserver vserver1 -fields v4-grace-seconds

vserver   v4-grace-seconds
--------- ----------------
vserver1  45
```

The result is that, after a restart, a controller will pause IO while all the clients reclaim their leases and locks. Once the grace period ends, the server will resume IO operations.

This grace period controls lease reclamation during network interface changes, but there is a second grace period that controls reclamation during storage failover, `locking.grace_lease_seconds`. This is a node-level option.

```
cluster01::> node run [node names or *] options
locking.grace_lease_seconds
```

For example, if you frequently needed to perform LIF failovers, and needed to reduce the grace period, you would change `v4-grace-seconds`. If you wanted to improve the IO resumption time during controller failover, you would need to change `locking.grace_lease_seconds`.

Only alter these values with caution and after fully understanding the risks and consequences. The IO pauses involved with failover and migration operations with NFSv4.X cannot be avoided entirely. Lock, lease, and grace periods are part of the NFS RFC. For many customers, NFSv3 is preferable because failover times are faster.

Lease timeouts vs grace periods

The grace period and the lease period are connected. As mentioned above, the default lease timeout is 30 seconds, which means NFSv4 clients must check in with the server at least every 30 seconds or they lose their leases and, in turn, their locks. The grace period exists to allow an NFS server to rebuild lease/lock data, and it defaults to 45 seconds. The grace period must be longer than the lease period. This ensures that an NFS client environment that is designed to renew leases at least every 30 seconds will have the ability to check in with the server after a restart. A grace period of 45 seconds ensures that all those clients that expect to renew their leases at least every 30 seconds definitely have the opportunity to do so.

If a 30 second timeout isn’t acceptable, you could choose to extend the lease period.

If you want to increase the lease timeout to 60 seconds in order to withstand a 60 second network outage, you’re going to have to increase the grace period as well. That means you’re going to experience longer IO pauses during controller failover.

This shouldn’t normally be a problem. Typical users only update ONTAP controllers once or twice per year, and unplanned failover due to hardware failures are extremely rare. In addition, if you had a network where a 60-second network outage was a concerning possibility, and you needed to the lease timeout to 60 seconds, then you probably wouldn’t object to rare storage system failover resulting in a 61 second pause either. You’ve already acknowledged you have a network that’s pausing for 60+ seconds rather frequently.

NFS caching

The presence of any of the following mount options causes host caching to be disabled:

```
cio, actimeo=0, noac, forcedirectio
```

These settings can have a severe negative effect on the speed of software installation, patching, and backup/restore operations. In some cases, especially with clustered applications, these options are required as an inevitable result of the need to deliver cache-coherency across all nodes in the cluster. In other cases, customers mistakenly use these parameters and the result is unnecessary performance damage.

Many customers temporarily remove these mount options during installation or patching of the application binaries. This removal can be performed safely if the user verifies that no other processes are actively using the target directory during the installation or patching process.

NFS transfer sizes

By default, ONTAP limits NFS I/O sizes to 64K.

Random I/O with an most applications and databases uses a much smaller block size which is well below the 64K maximum. Large-block I/O is usually parallelized, so the 64K maximum is also not a limitation to obtaining maximum bandwidth.

There are some workloads where the 64K maximum does create a limitation. In particular, single-threaded operations such as backup or recovery operation or a database full table scan run faster and more efficiently if the database can perform fewer but larger I/Os. The optimum I/O handling size for ONTAP is 256K.

The maximum transfer size for a given ONTAP SVM can be changed as follows:

```
Cluster01::> set advanced
Warning: These advanced commands are potentially dangerous; use them only
when directed to do so by NetApp personnel.
Do you want to continue? {y|n}: y
Cluster01::*> nfs server modify -vserver vserver1 -tcp-max-xfer-size
262144
Cluster01::*>
```

Never decrease the maximum allowable transfer size on ONTAP below the value of rsize/wsize of currently mounted NFS file systems. This can create hangs or even data corruption with some operating systems. For example, if NFS clients are currently set at an rsize/wsize of 65536, then the ONTAP maximum transfer size could be adjusted between 65536 and 1048576 with no effect because the clients themselves are limited. Reducing the maximum transfer size below 65536 can damage availability or data.

NVFAIL

NVFAIL is a feature within ONTAP that ensures the integrity during catastrophic failover scenarios.

Databases are vulnerable to corruption during storage failover events because they maintain large internal caches. If a catastrophic event requires forcing an ONTAP failover or forcing MetroCluster switchover, irrespective of the health of the overall configuration, the result is previously acknowledged changes may be effectively discarded. The contents of the storage array jump backward in time, and the state of the database cache no longer reflects the state of the data on disk. This inconsistency results in data corruption.

Caching can occur at the application or server layer. For example, an Oracle Real Application Cluster (RAC) configuration with servers active on both a primary and a remote site caches data within the Oracle SGA. A forced switchover operation that resulted in lost data would put the database at risk of corruption because the blocks stored in the SGA might not match the blocks on disk.

A less obvious use of caching is at the OS file system layer. Blocks from a mounted NFS file system might be cached in the OS. Alternatively, a clustered file system based on LUNs located on the primary site could be mounted on servers at the remote site, and once again data could be cached. A failure of NVRAM or a forced takeover or forced switchover in these situations could result in file system corruption.

ONTAP protects databases and operating systems from this scenario with NVFAIL and its associated settings.

ASM Reclamation Utility (ASMRU)

ONTAP efficiently removes zeroed blocks written to a file or LUN when inline compression is enabled. Utilities such as the Oracle ASM Reclamation Utility (ASRU) work by writing zeros to unused ASM extents.

This allows DBAs to reclaim space on the storage array after data is deleted. ONTAP intercepts the zeros and deallocates the space from the LUN. The reclamation process is extremely fast because no data is being written within the storage system.

From a database perspective, the ASM diskgroup contains zeros, and reading those regions of the LUNs would result in a stream of zeros, but ONTAP does not store the zeros on drives. Instead, simple metadata changes are made that internally mark the zeroed regions of the LUN as empty of any data.

For similar reasons, performance testing involving zeroed data is not valid since blocks of zeros are not actually processed as writes within the storage array.

When using ASRU, ensure that all Oracle-recommended patches are installed.