FC SAN

Enterprise applications

NetApp
February 10, 2026

Table of Contents

FC SAN

LUN Alignment

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

ASA r2 systems use the same ONTAP architecture as AFF/FAS but with a simplified configuration model. ASA r2 systems use Storage Availability Zones (SAZ) instead of aggregates, but the alignment principles remain the same because ONTAP manages block layout consistently across platforms. However, note these ASA-specific points:

- ASA r2 systems provide active-active symmetric paths for all LUNs, which eliminates path asymmetry concerns during alignment.
- Storage units (LUNs) are thin-provisioned by default; alignment does not change this behavior.
- Snapshot reserve and automatic snapshot deletion can be configured during LUN creation (ONTAP 9.18.1 and later).

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 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.

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.

Additional best practices:

- Verify HBA firmware and OS settings against the NetApp Interoperability Matrix Tool (IMT).
- Use sanlun utilities to confirm path health and alignment.
- For Oracle ASM and LVM, ensure configuration files (/etc/lvm/lvm.conf, /etc/sysconfig/oracleasm) are properly set to avoid alignment issues.

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 Storage Availability Zone (SAZ) on ASA r2 systems. As a result, the performance of the LUN is unaffected by its size because the LUN draws on the full performance potential of the SAZ 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.

ASA r2 considerations:

- Maximum LUN size for ASA r2 is 128TB, which allows for fewer, larger LUNs without performance impact.
- ASA r2 uses Storage Availability Zones (SAZ) instead of aggregates, but this does not change LUN sizing logic for Oracle workloads.
- Thin provisioning is enabled by default; resizing LUNs is non-disruptive and does not require taking them offline.

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.

With ASA r2, the principles for LUN count remain the same as AFF/FAS because ONTAP handles parallel I/O similarly across platforms. However, ASA r2’s SAN-only architecture and active-active symmetric paths ensure consistent performance across all LUNs.

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 Oracle database workload. Less than four LUNs might create performance limitations because of limitations in host SCSI implementations. Increasing beyond sixteen LUNs rarely improves performance except in extreme cases (such as very high random I/O SSD workloads).

LUN placement

Optimal placement of database LUNs within ASA r2 systems primarily depends on how various ONTAP features will be used.

In ASA r2 systems, storage units (LUNs or NVMe namespaces) are created from a simplified storage layer called Storage Availability Zones (SAZs), which act as common pools of storage for an HA pair.

> There is typically only one storage availability zone (SAZ) per HA pair.

Storage Availability Zones (SAZ)

In ASA r2 systems, volumes are still there, but they are automatically created when storage units are created. Storage units (LUNs or NVMe namespaces) are provisioned directly within the automatically created volumes in Storage Availability Zones (SAZs). This design eliminates the need for manual volume management and makes provisioning more direct and streamlined for block workloads like Oracle databases.

SAZs and Storage units

Related storage units (LUNs or NVMe namespaces) are normally co-located within a single Storage Availability Zone (SAZ). For example, a database that requires 10 storage units (LUNs) would typically have all 10 units placed in the same SAZ for simplicity and performance.

> - Using a 1:1 ratio of storage units to volumes, meaning one storage unit (LUN) per volume, is the ASA r2 default behavior.
> - In case of more than one HA pair in the ASA r2 system, storage units (LUNs) for a given database can be distributed across multiple SAZs to optimize controller utilization and performance.

> In context of FC SAN, here storage unit refers to LUN.

Consistency Groups (CGs), LUNs, and snapshots

In ASA r2, snapshot policies and schedules are applied at the Consistency Group level, which is a logical construct that groups multiple LUNs or NVMe namespaces for coordinated data protection. A dataset that consists of 10 LUNs would require only a single snapshot policy when those LUNs are part of the same Consistency Group.

Consistency Groups ensure atomic snapshot operations across all included LUNs. For example, a database that resides on 10 LUNs, or a VMware-based application environment consisting of 10 different OSs, can be protected as a single, consistent object if the underlying LUNs are grouped in the same consistency group. If they are placed in different consistency groups, snapshots may or may not be perfectly synchronized, even if scheduled at the same time.

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

CGs, LUNs, and SnapMirror

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

Co-locating related LUNs in a single consistency group 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 consistency groups, the replicas may or may not be consistent with one another.

> SnapMirror replication on ASA r2 systems has the following limitations:
>
> - SnapMirror synchronous replication is not supported.
> - SnapMirror active sync is supported only between two ASA r2 systems.
> - SnapMirror asynchronous replication is supported only between two ASA r2 systems.
> - SnapMirror asynchronous replication is not supported between an ASA r2 system and an ASA, AFF or FAS system or the cloud.
>
> Learn more about SnapMirror replication policies supported on ASA r2 systems.

CGs, LUNs, and QoS

While QoS can be selectively applied to individual LUNs, it is usually easier to set it at the consistency group level. For example, all of the LUNs used by the guests in a given ESX server could be placed in a single consistency group, and then an ONTAP adaptive QoS policy could be applied. The result is a self-scaling IOPS-per-TiB 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 consistency group than to set 10 individual 10K IOPS limits, one on each LUN.

Multiple CG layouts

There are some cases where distributing LUNs across multiple consistency groups may be beneficial. The primary reason is controller striping. For example, an HA ASA r2 storage system might be hosting a single Oracle 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 consistency group on controller 1, and the other half of the LUNs in a single consistency group on controller 2.

Similarly, for environments hosting many databases, distributing LUNs across multiple consistency groups can ensure balanced controller utilization. For example, an HA system hosting 100 databases of 10 LUNs each might assign 5 LUNs to a consistency group on controller 1 and 5 LUNs to a consistency group on controller 2 per database. This guarantees symmetric loading as additional databases are provisioned.

None of these examples involve a 1:1 LUN-to-consistency group ratio, though. The goal remains to optimize manageability by grouping related LUNs logically in consistency group.

One example where a 1:1 LUN to consistency group ratio makes sense is containerized workloads, where each LUN might really represent a single workload requiring separate snapshot and replication policies and thus need to be 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 or Oracle ASM disk group reaches its capacity limit on ASA r2, there are two options for increasing available space:

- Increase the size of the existing LUNs (storage units)
- Add a new LUN to an existing ASM disk group or LVM volume group and grow the contained logical volume

Although LUN resizing is supported on ASA r2, it is generally better to use a Logical Volume Manager (LVM) such as Oracle ASM. One of the principal reasons LVMs exist is to avoid the need for frequent LUN resizing. With an LVM, multiple LUNs are combined into a virtual pool of storage. Logical volumes carved from this pool can be easily resized without impacting the underlying storage configuration.

Additional benefits of using LVM or ASM include:

- Performance optimization: Distributes I/O across multiple LUNs, reducing hotspots.
- Flexibility: Add new LUNs without disrupting existing workloads.
- Transparent migration: ASM or LVM can relocate extents to new LUNs for balancing or tiering without host downtime.

> Key ASA r2 considerations:
>
> - LUN resizing is performed at the storage unit level within a Storage VM (SVM) using capacity from the Storage Availability Zone (SAZ).
> - For Oracle, best practice is to add LUNs to ASM disk groups rather than resizing existing LUNs, to maintain striping and parallelism.

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 storage 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.

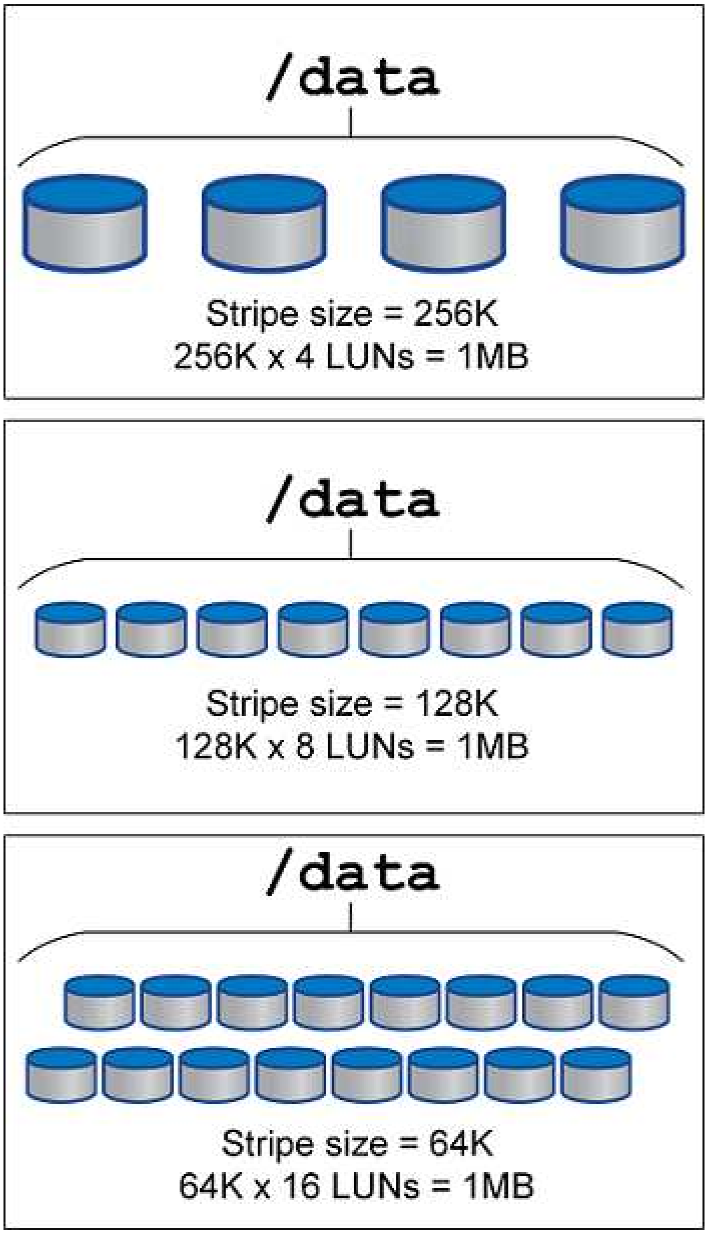
Performance-intensive application I/O is nearly always either (a) in units of the basic block size or (b) one megabyte.

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.

Best practice for LVM striping with Oracle database:

- Stripe size ≥ database block size.
- Stripe size * number of LUNs ≈ 1MB for optimal parallelism.
- Use multiple LUNs per ASM disk group to maximize throughput and avoid hotspots.

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.


/data
Stripe size = 256K
256K x 4 LUNs = 1MB
/data
Stripe size = 128K
128K x 8 LUNs = 1MB
/data
Stripe size = 64K
64K x 16 LUNs = 1MB