FC SAN

Enterprise applications

NetApp
March 13, 2024

Table of Contents

FC SAN

LUN Alignment for Oracle database I/O

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.

Oracle database 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.

Oracle database LUN resizing and LVM-based 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 with Oracle databases

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.

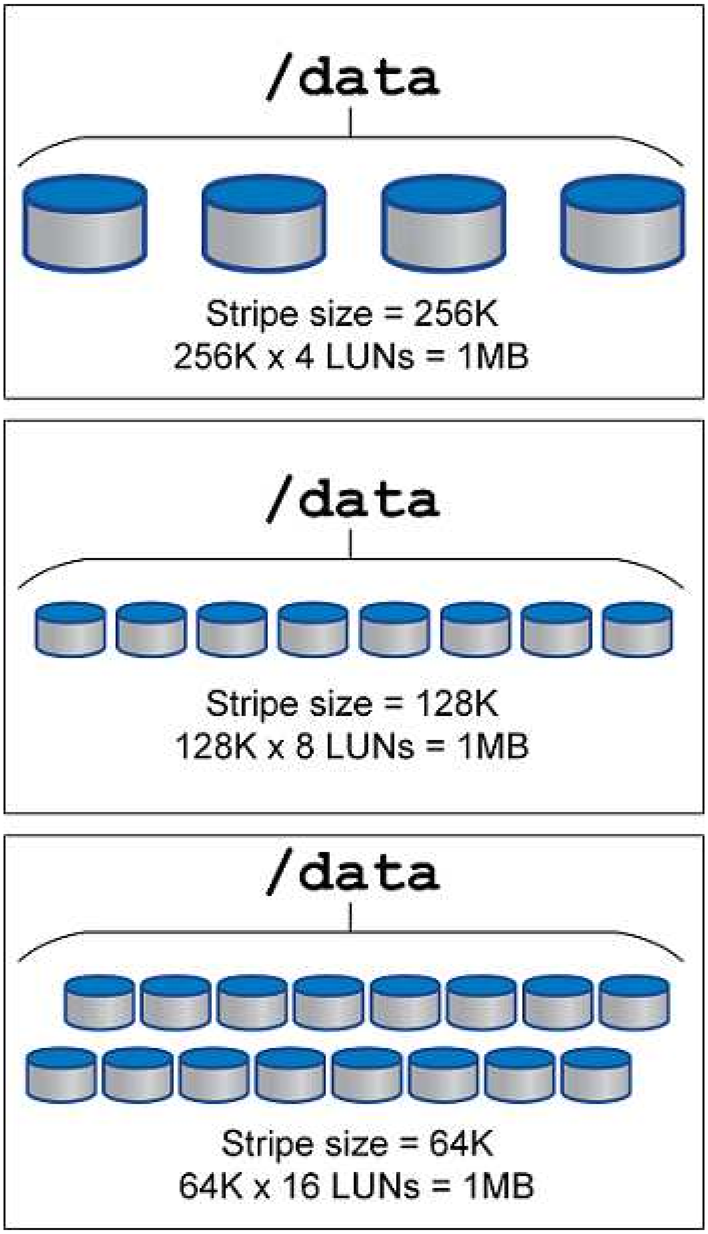
Striping with spinning drives was more difficult because the I/O pattern had to be known in advance. If the striping wasn't correctly tuned for the true I/O patterns, striped configurations could damage performance. With Oracle databases, and especially with all-flash configurations, striping is much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


/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