ONTAP configuration

Enterprise applications

NetApp
May 08, 2024

Table of Contents

ONTAP configuration

RAID and Oracle databases

RAID refers to the use of redundancy to protect data against the loss of a drive.

Questions occasionally arise concerning RAID levels in the configuration of NetApp storage used for Oracle databases and other enterprise applications. Many legacy Oracle best practices regarding storage array configuration contain warnings about using RAID mirroring and/or avoiding certain types of RAID. Although they raise valid points, these sources do not apply to RAID 4 and the NetApp RAID DP and RAID-TEC technologies used in ONTAP.

RAID 4, RAID 5, RAID 6, RAID DP, and RAID-TEC all use parity to ensure drive failure does not result in data loss. These RAID options offer much better storage utilization in comparison to mirroring, but most RAID implementations have a drawback that affects write operations. Completion of a write operation on other RAID implementations may require multiple drive reads to regenerate the parity data, a process commonly called the RAID penalty.

ONTAP, however, does not incur this RAID penalty. This is because of the integration of NetApp WAFL (Write Anywhere File Layout) with the RAID layer. Write operations are coalesced in RAM and prepared as a complete RAID stripe, including parity generation. ONTAP does not need to perform a read in order to complete a write, which means that ONTAP and WAFL avoid the RAID penalty. Performance for latency-critical operations, such as redo logging, is unimpeded, and random data-file writes do not incur any RAID penalty resulting from a need to regenerate parity.

With respect to statistical reliability, even RAID DP offers better protection than RAID mirroring. The primary problem is the demand made on drives during a RAID rebuild. With a mirrored RAID set, the risk of data loss from a drive failing while rebuilding to its partner in the RAID set is much greater than the risk of a triple- drive failure in a RAID DP set.

Oracle databases and storage capacity management

Managing a database or other enterprise application with predictable, manageable, high performance enterprise storage requires some free space on the drives for data and metadata management. The amount of free space required depends on the type of drive used, and business processes.

Free space is defined as any space that is not used for actual data and includes unallocated space on the aggregate itself and unused space within the constituent volumes. Thin provisioning must also be considered. For example, a volume might contain a 1TB LUN of which only 50% is utilized by real data. In a thin provisioned environment, this would correctly appear to be consuming 500GB of space. However, in a fully provisioned environment, the full capacity of 1TB appears to be in use. The 500GB of unallocated space is hidden. This space is unused by actual data and should therefore be included in the calculation of total free space.

NetApp recommendations for storage systems used for enterprise applications are as follows:

SSD aggregates, including AFF systems

NetApp recommends a minimum of 10% free space. This includes all unused space, including free space within the aggregate or a volume and any free space that is allocated due to the use of full provisioning but is not used by actual data. Logical space is unimportant, the question is how much actual free physical space is available for data storage.

The recommendation of 10% free space is very conservative. SSD aggregates can support workloads at even higher levels of utilization without any effect on performance. However, as the utilization of the aggregate increases, the risk of running out of space also increases if utilization is not monitored carefully. Furthermore, while running a system at 99% capacity may not incur a performance penalty, but it would likely incur management effort trying to keep it from filling up completely while additional hardware is ordered, and it may take some time to procure and install additional drives.

HDD aggregates, including Flash Pool aggregates

NetApp recommends a minimum of 15% free space when spinning drives are used. This includes all unused space, including free space within the aggregate or a volume and any free space that is allocated due to the use of full provisioning but is not used by actual data. Performance will be impacted at free speech approaches 10%.

Oracle databases and Storage Virtual Machines

Oracle database storage management is centralized on a Storage Virtual Machine (SVM)

An SVM, known as a vserver at the ONTAP CLI, is a basic functional unit of storage, and it is useful to compare an SVM to a guest on a VMware ESX server.

When first installed, ESX has no pre-configured capabilities, such as hosting a guest OS or supporting an end-user application. It is an empty container until a virtual machine (VM) is defined. ONTAP is similar. When ONTAP is first installed, it has no data-serving capabilities until an SVM is created. It is the SVM personality that defines the data services.

As with other aspects of storage architecture, the best options for SVM and logical interface (LIF) design depend heavily on scaling requirements and business needs.

SVMs

There is no official best practice for provisioning SVMs for ONTAP. The right approach depends on management and security requirements.

Most customers operate one primary SVM for most of their day-to-day requirements but then create a small number of SVMs for special needs. For example, you might wish to create:

- An SVM for a critical business database managed by a specialist team
- An SVM for a development group to whom complete administrative control has been given so that they can manage their own storage independently
- An SVM for sensitive business data, such as human resources or financial reporting data, for which the administrative team must be limited

In a multi-tenant environment, each tenant’s data can be given a dedicated SVM. The limit for the number of SVMs and LIFs per cluster, HA pair, and node are dependant on the protocol being used, the node model, and the version of ONTAP. Consult the NetApp Hardware Universe for these limits.

Oracle database performance management with ONTAP QoS

Safely and efficiently managing multiple Oracle databases requires an effective QoS strategy. The reason is the ever-increasing performance capabilities of a modern storage system.

Specifically, the increased adoption of all-flash storage has enabled the consolidation of workloads. Storage arrays relying on spinning media tended to support only a limited number of I/O-intensive workloads because of the limited IOPS capabilities of older rotational drive technology. One or two highly active databases would saturate the underlying drives long before the storage controllers reached their limits. This has changed. The performance capability of a relatively small number of SSD drives can saturate even the most powerful storage controllers. This means the full capabilities of the controllers can be leveraged without the fear of sudden collapse of performance as spinning media latency spikes.

As a reference example, a simple two-node HA AFF A800 system is capable of servicing up to one million random IOPS before latency climbs above one millisecond. Very few single workloads would be expected to reach such levels. Fully utilizing this AFF A800 system array will involve hosting multiple workloads, and doing this safely while ensuring predictability requires QoS controls.

There are two types of quality of service (QoS) in ONTAP: IOPS and bandwidth. QoS controls can be applied to SVMs, volumes, LUNs, and files.

IOPS QoS

An IOPS QoS control is obviously based on the total IOPS of a given resource, but there are a number of aspects of IOPS QoS that might not be intuitive. A few customers were initially puzzled by the apparent increase in latency when an IOPS threshold is reached. Increasing latency is the natural result of limiting IOPS. Logically, it functions similarly to a token system. For example, if a given volume containing datafiles has a 10K IOPS limit, each I/O that arrives must first receive a token to continue processing. So long as no more than 10K tokens have been consumed in a given second, no delays are present. If IO operations must wait to receive their token, this wait appears as additional latency. The harder a workload pushes up against the QoS limit, the longer each IO must wait in the queue for its turn to be processed, which appears to the user as higher latency.

> ⓘ Be cautious when applying QoS controls to database transaction/redo log data. While the performance demands of redo logging are normally much, much lower than datafiles, the redo log activity is bursty. The IO happens in brief pulses, and a QoS limit that appears appropriate for average redo IO levels may be too low for the actual requirements. The result can be severe performance limitations as QoS engages with each redo log burst. In general, redo and archive logging should not be limited by QoS.

Bandwidth QoS

Not all I/O sizes are the same. For example, a database might be performing a large number of small block reads which would result in the IOPS threshold being reached, but databases might also be performing a full table scan operation which would consist of a very small number of large block reads, consuming a very large amount of bandwidth but relatively few IOPS.

Likewise, a VMware environment might drive a very high number of random IOPS during boot-up, but would perform fewer but larger IOs during an external backup.

Sometimes effectively managing performance require either IOPS or bandwidth QoS limits, or even both.

Minimum/guaranteed QoS

Many customers seek a solution that includes guaranteed QoS, which is more difficult to achieve than it might seem and is potentially quite wasteful. For example, placing 10 databases with a 10K IOPS guarantee requires sizing a system for a scenario in which all 10 databases are simultaneously running at 10K IOPS, for a total of 100K.

The best use for minimum QoS controls is to protect critical workloads. For example, consider an ONTAP controller with a maximum possible IOPS of 500K and a mix of production and development workloads. You should apply maximum QoS policies to development workloads to prevent any given database from monopolizing the controller. You would then apply minimum QoS policies to production workloads to make sure that they always have the required IOPS available when needed.

Adaptive QoS

Adaptive QoS refers to the ONTAP feature where the QoS limit is based on the capacity of the storage object. It is rarely used with databases because there is not usually any link between the size of a database and its performance requirements. Large databases can be nearly inert, while smaller databases can be the most IOPS-intensive.

Adaptive QoS can be very useful with virtualization datastores because the IOPS requirements of such datasets do tend to correlate to the total size of the database. A newer datastore containing 1TB of VMDK files is likely to need about half the performance as a 2TB datastore. Adaptive QoS allows you to grow the QoS limits automatically as the datastore becomes populated with data.

Oracle databases and ONTAP efficiency features

ONTAP space efficiency features are optimized for Oracle databases. In almost all cases, the best approach is to leave the defaults in place with all efficiency features enabled.

Space efficiency features, such as compression, compaction, and deduplication are designed to increase the amount of logical data that fits on a given amount of physical storage. The result is lower costs and management overhead.

At a high level, compression is a mathematical process whereby patterns in data are detected and encoded in a way that reduces space requirements. In contrast, deduplication detects actual repeated blocks of data and removes the extraneous copies. Compaction allows multiple logical blocks of data to share the same physical block on media.

ⓘ See the sections below on thin provisioning for an explanation of the interaction between storage efficiency and fractional reservation.

Compression

Prior to the availability of all-flash storage systems, array-based compression was of limited value because most I/O-intensive workloads required a very large number of spindles to provide acceptable performance. Storage systems invariably contained much more capacity than required as a side effect of the large number of drives. The situation has changed with the rise of solid-state storage. There is no longer a need to vastly overprovision drives purely to obtain good performance. The drive space in a storage system can be matched to actual capacity needs.

The increased IOPS capability of solid-state drives (SSDs) almost always yields cost savings compared to spinning drives, but compression can achieve further savings by increasing the effective capacity of solid-state media.

There are several ways to compress data. Many databases include their own compression capabilities, but this is rarely observed in customer environments. The reason is usually the performance penalty for a **change** to compressed data, plus with some applications there are high licensing costs for database-level compression. Finally, there is the overall performance consequences to database operations. It makes little sense to pay a high per-CPU license cost for a CPU that performs data compression and decompression rather than real database work. A better option is to offload the compression work on to the storage system.

Adaptive compression

Adaptive compression has been thoroughly tested with enterprise workloads with no observed effect on performance, even in an all-flash environment in which latency is measured in microseconds. Some customers have even reported a performance increase with the use of compression because the data remains compressed in cache, effectively increasing the amount of available cache in a controller.

ONTAP manages physical blocks in 4KB units. Adaptive compression uses a default compression block size of 8KB, which means data is compressed in 8KB units. This matches the 8KB block size most often used by relational databases. Compression algorithms become more efficient as more data is compressed as a single unit. A 32KB compression block size would be more space-efficient than an 8KB compression block unit. This does mean that adaptive compression using the default 8KB block size does lead to slightly lower efficiency rates, but there is also a significant benefit to using a smaller compression block size. Database workloads include a large amount of overwrite activity. Overwriting a 8KB of a compressed 32KB block of data requires reading back the entire 32KB of logical data, decompressing it, updating the required 8KB region, recompressing, and then writing the entire 32KB back to the drives. This is a very expensive operation for a storage system and is the reason some competing storage arrays based on larger compression block sizes also incur a significant performance penalty with database workloads.

> ⓘ The block size used by adaptive compression can be increased up to 32KB. This may improve storage efficiency and should be considered for quiescent files such as transaction logs and backup files when a substantial amount of such data is stored on the array. In some situations, active databases that use a 16KB or 32KB block size may also benefit from increasing the block size of adaptive compression to match. Consult a NetApp or partner representative for guidance on whether this is appropriate for your workload.

> ❗ Compression block sizes larger than 8KB should not be used alongside deduplication on streaming backup destinations. The reason is small changes to the backed-up data affect the 32KB compression window. If the window shifts, the resulting compressed data differs across the entire file. Deduplication occurs after compression, which means the deduplication engine sees each compressed backup differently. If deduplication of streaming backups is required, only 8KB block adaptive compression should be used. Adaptive compression is preferable, because it works at a smaller block size and does not disrupt deduplication efficiency. For similar reasons, host-side compression also interferes with deduplication efficiency.

Compression alignment

Adaptive compression in a database environment requires some consideration of compression block alignment. Doing so is only a concern for data that is subject to random overwrites of very specific blocks. This approach is similar in concept to overall file system alignment, where the start of a filesystem must be aligned to a 4K device boundary and the blocksize of a filesystem must be a multiple of 4K.

For example, an 8KB write to a file is compressed only if it aligns with an 8KB boundary within the file system itself. This point means that it must fall on the first 8KB of the file, the second 8KB of the file, and so forth. The simplest way to ensure correct alignment is to use the correct LUN type, any partition created should have an offset from the start of the device that is a multiple of 8K, and use a filesystem block size that is a multiple of the database block size.

Data such as backups or transaction logs are sequentially written operations that span multiple blocks, all of which are compressed. Therefore, there is no need to consider alignment. The only I/O pattern of concern is the random overwrites of files.

Data compaction

Data compaction is a technology that improves compression efficiency. As stated previously, adaptive compression alone can provide at best 2:1 savings because it is limited to storing an 8KB I/O in a 4KB WAFL block. Compression methods with larger block sizes deliver better efficiency. However, they are not suitable for data that is subject to small block overwrites. Decompressing 32KB units of data, updating an 8KB portion, recompressing, and writing back to the drives creates overhead.

Data compaction works by allowing multiple logical blocks to be stored within physical blocks. For example, a database with highly compressible data such as text or partially full blocks may compress from 8KB to 1KB. Without compaction, that 1KB of data would still occupy an entire 4KB block. Inline data compaction allows that 1KB of compressed data to be stored in just 1KB of physical space alongside other compressed data. It is not a compression technology; it is simply a more efficient way of allocating space on the drives and therefore should not create any detectable performance effect.

The degree of savings obtained vary. Data that is already compressed or encrypted cannot generally be further compressed, and therefore such datasets do not benefit from compaction. In contrast, newly initialized datafiles that contain little more than block metadata and zeros compress up to 80:1.

Temperature sensitive storage efficiency

Temperature sensitive storage efficiency (TSSE) is a available in ONTAP 9.8 and later that relies on block access heat maps to identify infrequently accessed blocks and compress them with greater efficiency.

Deduplication

Deduplication is the removal of duplicate block sizes from a dataset. For example, if the same 4KB block existed in 10 different files, deduplication would redirect that 4KB block within all 10 files to the same 4KB physical block. The result would be a 10:1 improvement in efficiency for that data.

Data such as VMware guest boot LUNs usually deduplicate extremely well because they consist of multiple copies of the same operating system files. Efficiency of 100:1 and greater have been observed.

Some data does not contain duplicate data. For example, an Oracle block contains a header that is globally unique to the database and a trailer that is nearly unique. As a result, deduplication of an Oracle database rarely delivers more than 1% savings. Deduplication with MS SQL databases is slightly better, but unique metadata at the block level is still a limitation.

Space savings of up to 15% in databases with 16KB and large block sizes have been observed in a few cases. The initial 4KB of each block contains the globally unique header, and the final 4KB block contains the nearly unique trailer. The internal blocks are candidates for deduplication, although in practice this is almost entirely attributed to the deduplication of zeroed data.

Many competing arrays claim the ability to deduplicate databases based on the presumption that a database is copied multiple times. In this respect, NetApp deduplication could also be used, but ONTAP offers a better option: NetApp FlexClone technology. The end result is the same; multiple copies of a database that share most of the underlying physical blocks are created. Using FlexClone is much more efficient than taking the time to copy database files and then deduplicating them. It is, in effect, nonduplication rather than deduplication, because a duplicate is never created in the first place.

Efficiency and thin provisioning

Efficiency features are forms of thin provisioning. For example, a 100GB LUN occupying a 100GB volume might compress down to 50GB. There are no actual savings realized yet because the volume is still 100GB. The volume must first be reduced in size so that the space saved can be used elsewhere on the system. If later changes to the 100GB LUN result in the data becoming less compressible, then the LUN grows in size and the volume could fill up.

Thin provisioning is strongly recommended because it can simplify management while delivering a substantial improvement in usable capacity with associated cost savings. The reason is simple - database environments frequently include a lot of empty space, a large number of volumes and LUNs, and compressible data. Thick provisioning results in the reservation of space on storage for volumes and LUNs just in case they someday become 100% full and contain 100% uncompressible data. That is unlikely to ever occur. Thin provisioning allows that space to be reclaimed and used elsewhere and allows capacity management to be based on the storage system itself rather than many smaller volumes and LUNs.

Some customers prefer to use thick provisioning, either for specific workloads or generally based on established operational and procurement practices.

Caution: If a volume is thick provisioned, care must be taken to completely disable all efficiency features for that volume, including decompression and the removal of deduplication using the `sis undo` command. The volume should not appear in `volume efficiency show` output. If it does, the volume is still partially configured for efficiency features. As a result, overwrite guarantees work differently, which increases the chance that configuration oversights cause the volume to unexpectedly run out of space, resulting in database I/O errors.

Efficiency best practices

NetApp recommends the following:

AFF defaults

Volumes created on ONTAP running on an all-flash AFF system are thin provisioned with all inline efficiency features enabled. Although databases generally do not benefit from deduplication and may include uncompressible data, the default settings are nevertheless appropriate for almost all workloads. ONTAP is designed to efficiently process all types of data and I/O patterns, whether or not they result in savings. Defaults should only be changed if the reasons are fully understood and there is a benefit to deviating.

General recommendations

- If volumes and/or LUNs are not thin provisioned, you should must disable all efficiency settings because using these features provides no savings and the combination of thick provisioning with space efficiency enabled can cause unexpected behavior, including out-of-space errors.
- If data is not subject to overwrites, such as with backups or database transaction logs, you can achieve greater efficiency by enabling TSSE with a low cooling period.
- Some files might contain a significant amount of uncompressible data, for example when compression is already enabled at the application level of files are encrypted. If any of these scenarios are true, consider disabling compression to allow more efficient operation on other volumes containing compressible data.

- Do not use both 32KB compression and deduplication with database backups. See the section Adaptive compression for details.

Thin provisioning with Oracle databases

Thin provisioning for an Oracle database requires careful planning because the result is configuring more space on a storage system than is necessarily physically available. It is very much worth the effort because, when done correctly, the result is significant cost savings and improvements in manageability.

Thin provisioning comes in many forms and is integral to many features that ONTAP offers to an enterprise application environment. Thin provisioning is also closely related to efficiency technologies for the same reason: efficiency features allow more logical data to be stored than what technically exists on the storage system.

Almost any use of snapshots involves thin provisioning. For example, a typical 10TB database on NetApp storage includes around 30 days of snapshots. This arrangement results in approximately 10TB of data visible in the active file system and 300TB dedicated to snapshots. The total 310TB of storage usually resides on approximately 12TB to 15TB of space. The active database consumes 10TB, and the remaining 300TB of data only requires 2TB to 5TB of space because only the changes to the original data are stored.

Cloning is also an example of thin provisioning. A major NetApp customer created 40 clones of an 80TB database for use by development. If all 40 developers using these clones overwrote every block in every datafile, over 3.2PB of storage would be required. In practice, turnover is low, and the collective space requirement is closer to 40TB because only changes are stored on the drives.

Space management

Some care must be taken with thin provisioning an application environment because data change rates can increase unexpectedly. For example, space consumption due to snapshots can grow rapidly if database tables are reindexed, or wide-scale patching is applied to VMware guests. A misplaced backup can write a large amount of data in a very short time. Finally, it can be difficult to recover some applications if a file system runs out of free space unexpectedly.

Fortunately, these risks can be addressed with careful configuration of `volume-autogrow` and `snapshot-autodelete` policies. As their names imply, these options enable a user to create policies that automatically clear space consumed by snapshots or grow a volume to accommodate additional data. Many options are available and needs vary by customer.

See the logical storage management documentation for a complete discussion of these features.

Fractional reservations

Fractional reserve refers to the behavior of a LUN in a volume with respect to space efficiency. When the option `fractional-reserve` is set to 100%, all data in the volume can experience 100% turnover with any data pattern without exhausting space on the volume.

For example, consider a database on a single 250GB LUN in a 1TB volume. Creating a snapshot would immediately result in the reservation of an additional 250GB of space in the volume to guarantee that the volume does not run out of space for any reason. Using fractional reserves is generally wasteful because it is extremely unlikely that every byte in the database volume would need to be overwritten. There is no reason to reserve space for an event that never happens. Still, if a customer cannot monitor space consumption in a storage system and must be certain that space never runs out, 100% fractional reservations would be required to use snapshots.

Compression and deduplication

Compression and deduplication are both forms of thin provisioning. For example, a 50TB data footprint might compress to 30TB, resulting in a savings of 20TB. For compression to yield any benefits, some of that 20TB must be used for other data, or the storage system must be purchased with less than 50TB. The result is storing more data than is technically available on the storage system. From the data point of view, there is 50TB of data, even though it occupies only 30TB on the drives.

There is always a possibility that the compressibility of a dataset changes, which would result in increased consumption of real space. This increase in consumption means that compression must be managed as with other forms of thin provisioning in terms of monitoring and using `volume-autogrow` and `snapshot-autodelete`.

Compression and deduplication are discussed in further detail in the section xref:./oracle/efficiency.html

Compression and fractional reservations

Compression is a form of thin provisioning. Fractional reservations affect the use of compression, with one important note; space is reserved in advance of the snapshot creation. Normally, fractional reserve is only important if a snapshot exists. If there is no snapshot, fractional reserve is not important. This is not the case with compression. If a LUN is created on a volume with compression, ONTAP preserves space to accommodate a snapshot. This behavior can be confusing during configuration, but it is expected.

As an example, consider a 10GB volume with a 5GB LUN that has been compressed down to 2.5GB with no snapshots. Consider these two scenarios:

- Fractional reserve = 100 results in 7.5GB utilization
- Fractional reserve = 0 results in 2.5GB utilization

The first scenario includes 2.5GB of space consumption for current data and 5GB of space to account for 100% turnover of the source in anticipation of snapshot use. The second scenario reserves no extra space.

Although this situation might seem confusing, it is unlikely to be encountered in practice. Compression implies thin provisioning, and thin provisioning in a LUN environment requires fractional reservations. It is always possible for compressed data to be overwritten by something uncompressible, which means a volume must be thin provisioned for compression to result in any savings.

NetApp recommends the following reserve configurations:

- Set `fractional-reserve` to 0 when basic capacity monitoring is in place along with `volume-autogrow` and `snapshot-autodelete`.
- Set `fractional-reserve` to 100 if there is no monitoring ability or if it is impossible to exhaust space under any circumstance.

Free space and LVM space allocation

The efficiency of thin provisioning of active LUNs in a file system environment can be lost over time as data is deleted. Unless that deleted data is either overwritten with zeros (see also ASMRU or the space is released with TRIM/UNMAP space reclamation, the "erased" data occupies more and more unallocated whitespace in the file system. Furthermore, thin provisioning of active LUNs is of limited use in many database environments because datafiles are initialized to their full size at the time of creation.

Careful planning of LVM configuration can improve efficiency and minimize the need for storage provisioning and LUN resizing. When an LVM such as Veritas VxVM or Oracle ASM is used, the underlying LUNs are divided into extents that are only used when needed. For example, if a dataset begins at 2TB in size but could grow to 10TB over time, this dataset could be placed on 10TB of thin-provisioned LUNs organized in an LVM diskgroup. It would occupy only 2TB of space at the time of creation and would only claim additional space as extents are allocated to accommodate data growth. This process is safe as long as space is monitored.

Oracle databases and ONTAP controller failover/switchover

An understanding of storage takeover and switchover functions is required to ensure that Oracle database operations are not disrupted by these operations. In addition, the arguments used by takeover and switchover operations can affect data integrity if used incorrectly.

- Under normal conditions, incoming writes to a given controller are synchronously mirrored to its partner. In a NetApp MetroCluster environment, writes are also mirrored to a remote controller. Until a write is stored in nonvolatile media in all locations, it is not acknowledged to the host application.
- The media storing the write data is called nonvolatile memory or NVMEM. It is also sometimes referred to nonvolatile random- access memory (NVRAM), and it can be thought of as a write cache although it functions as a journal. In a normal operation, the data from NVMEM is not read; it is only used to protect data in the event of a software or hardware failure. When data is written to drives, the data is transferred from the RAM in the system, not from NVMEM.
- During a takeover operation, one node in a high availability (HA) pair takes over the operations from its partner. A switchover is essentially the same, but it applies to MetroCluster configurations in which a remote node takes over the functions of a local node.

During routine maintenance operations, a storage takeover or switchover operation should be transparent, other than for a potential brief pause in operations as the network paths change. Networking can be complicated, however, and it is easy to make errors, so NetApp strongly recommends testing takeover and switchover operations thoroughly before putting a storage system into production. Doing so is the only way to be sure that all network paths are configured correctly. In a SAN environment, carefully check the output of the command `sanlun lun show -p` to make sure that all expected primary and secondary paths are available.

Care must be taken when issuing a forced takeover or switchover. Forcing a change to storage configuration with these options means that the state of the controller that owns the drives is disregarded and the alternative node forcibly takes control of the drives. Incorrect forcing of a takeover can result in data loss or corruption. This is because a forced takeover or switchover can discard the contents of NVMEM. After the takeover or switchover is complete, the loss of that data means that the data stored on the drives might revert to a slightly older state from the point of view of the database.

A forced takeover with a normal HA pair should rarely be required. In almost all failure scenarios, a node shut downs and informs the partner so that an automatic failover takes place. There are some edge cases, such as a rolling failure in which the interconnect between nodes is lost and then one controller is lost, in which a forced takeover is required. In such a situation, the mirroring between nodes is lost before the controller failure, which means that the surviving controller would have no longer has a copy of the writes in progress. The takeover then needs to be forced, which means that data potentially is lost.

The same logic applies to a MetroCluster switchover. In normal conditions, a switchover is nearly transparent. However, a disaster can result in a loss of connectivity between the surviving site and the disaster site. From the point of view of the surviving site, the problem could be nothing more than an interruption in connectivity between sites, and the original site might still be processing data. If a node cannot verify the state of the primary controller, only a forced switchover is possible.

NetApp recommends taking the following precautions:

- Be very careful to not accidentally force a takeover or a switchover. Normally, forcing should not be required, and forcing the change can cause data loss.
- If a forced takeover or switchover is required, make sure that the applications are shut down, all file systems are dismounted and logical volume manager (LVM) volume groups are varyoffed. ASM diskgroups must be unmounted.
- In the event of a forced MetroCluster switchover, fence off the failed node from all surviving storage resources. For more information, see the MetroCluster Management and Disaster Recovery Guide for the relevant version of ONTAP.

MetroCluster and multiple aggregates

MetroCluster is a synchronous replication technology that switches to asynchronous mode if connectivity is interrupted. This is the most common request from customers, because guaranteed synchronous replication means that interruption in site connectivity leads to a complete stall of database I/O, taking the database out of service.

With MetroCluster, aggregates rapidly resynchronize after connectivity is restored. Unlike other storage technologies, MetroCluster should never require a complete remirroring after site failure. Only delta changes must be shipped.

In datasets that span aggregates, there is a small risk that additional data recovery steps would be required in a rolling disaster scenario. Specifically, if (a) connectivity between sites is interrupted, (b) connectivity is restored, (c) the aggregates reach a state in which some are synchronized and some are not, and then (d) the primary site is lost, the result is a surviving site in which the aggregates are not synchronized with one another. If this happens, parts of the dataset are synchronized with one another and it is not possible to bring up applications, databases, or datastores without recovery. If a dataset spans aggregates, NetApp strongly recommends leveraging snapshot-based backups with one of the many available tools to verify rapid recoverability in this unusual scenario.

Copyright information

Trademark information