Technical Report

Database Data Protection: Backup, Recovery, Replication, and DR

Jeffrey Steiner, NetApp

July 2017 | TR-4591

Important

Consult the Interoperability Matrix Tool (IMT) to determine whether the environment, configurations, and versions specified in this report support your environment

TABLE OF CONTENTS

1 Database Data Protection with ONTAP

The most mission-critical data is usually found in databases. An enterprise cannot operate without access to its data, and in some cases, the data defines the business. This data must be protected; however, data protection is more than just ensuring a usable backup—it is about performing the backups quickly and reliably as well as storing backups safely. The other side of data protection is data recovery. When data is inaccessible, the enterprise is affected and may be inoperative until data is restored. This process must be quick and reliable. Finally, most databases must be protected against disasters, which means maintaining a replica of the database. The replica must be sufficiently up to date, and it must be quick and simple to make the replica a fully operational database.

2 Database Data Protection

A database data protection architecture should be defined by business requirements. These requirements include factors such as the speed of recovery, the maximum permissible data loss, and backup retention needs. The data protection plan must also take into consideration various regulatory requirements for data retention and restoration. Finally, different data recovery scenarios must be considered, ranging from the typical and foreseeable recovery resulting from user or application errors up to disaster recovery scenarios that include the complete loss of a site.

Small changes in data protection and recovery policies can have a significant effect on the overall architecture of storage, backup, and recovery. It is critical to define and document standards before starting design work to avoid complicating a data protection architecture. Unnecessary features or levels of protection lead to unnecessary costs and management overhead, and an initially overlooked requirement can lead a project in the wrong direction or require last-minute design changes.

2.1 Recovery Time Objective

The recovery time objective (RTO) defines the maximum time allowed for the recovery of a service. For example, a human resources database might have an RTO of 24 hours because, although it would be very inconvenient to lose access to this data during the workday, the business can still operate. In contrast, a database supporting the general ledger of a bank would have an RTO measured in minutes or even seconds. An RTO of zero is not possible, because there must be a way to differentiate between an actual service outage and a routine event such as a lost network packet. However, a near-zero RTO is a typical requirement.

2.2 Recovery Point Objective

The recovery point objective (RPO) defines the maximum tolerable data loss. In a database context, the RPO is usually a question of how much log data can be lost in a specific situation. In a typical recovery scenario in which a database is damaged due to a product bug or user error, the RPO should be zero, meaning there should be no data loss. The recovery procedure involves restoring an earlier copy of the database files and then replaying the log files to bring the database state up to the desired point in time. The log files required for this operation should already be in place in the original location.

In unusual scenarios, log data might be lost. For example, an accidental or malicious `rm -rf *` of database files could result in the deletion of all data. The only option would be to restore from backup, including log files, and some data would inevitably be lost. The only option to improve the RPO in a traditional backup environment would be to perform repeated backups of the log data. This has limitations, however, because of the constant data movement and the difficulty maintaining a backup system as a constantly running service. One of the benefits of advanced storage systems is the ability to protect data from accidental or malicious damage to files and thus deliver a better RPO without data movement.

2.3 Disaster Recovery

Disaster recovery includes the IT architecture, policies, and procedures required to recover a service in the event of a physical disaster. This can include floods, fire, or person acting with malicious or negligent intent.

Disaster recovery is more than just a set of recovery procedures. It is the complete process of identifying the various risks, defining the data recovery and service continuity requirements, and delivering the right architecture with associated procedures.

When establishing data protection requirements, it is critical to differentiate between typical RPO and RTO requirements and the RPO and RTO requirements needed for disaster recovery. Some database environments require an RPO of zero and a near-zero RTO for data loss situations ranging from a relatively normal user error right up to a fire that destroys a data center. However, there are cost and administrative consequences for these high levels of protection.

In general, nondisaster data recovery requirements should be strict for two reasons. First, application bugs and user errors that damage a database are foreseeable to the point they are almost inevitable. Second, it is not difficult to design a backup strategy that can deliver an RPO of zero and a low RTO as long as the storage system is not destroyed. There is no reason not to address a significant risk that is easily remedied, which is why the RPO and RTO targets for local recovery should be aggressive.

Disaster recovery RTO and RPO requirements vary more widely based on the likelihood of a disaster and the consequences of the associated data loss or disruption to a business. RPO and RTO requirements should be based on the actual business needs and not on general principles. They must account for multiple logical and physical disaster scenarios.

Logical Disasters

Logical disasters include data corruption caused by users, application or OS bugs, and software malfunctions. Logical disasters can also include malicious attacks by outside parties with viruses or worms or by exploiting application vulnerabilities. In these cases, the physical infrastructure is undamaged but the underlying data is no longer valid.

An increasingly common type of logical disaster is known as ransomware, in which an attack vector is used to encrypt data. Encryption does not damage the data, but it makes it unavailable until payment is made to a third party. An increasing number of enterprises are being specifically targeted with ransomware hacks.

Physical Disasters

Physical disasters include the failure of components of an infrastructure that exceeds its redundancy capabilities and result in a loss of data or an extended loss of service. For example, RAID protection provides disk-drive redundancy, and the use of HBAs provides FC port and FC cable redundancy. Hardware failures of such components is foreseeable and does not impact availability.

In a database environment, it is generally possible to protect the infrastructure of an entire site with redundant components to the point where the only foreseeable physical disaster scenario is complete loss of the site. Disaster recovery planning then depends on site-to-site replication.

Synchronous and Asynchronous Data Protection

In an ideal world, all data would be synchronously replicated across geographically dispersed sites. Such replication is not always feasible or even possible for several reasons:

- Synchronous replication unavoidably increases write latency because all changes must be replicated to both locations before the application or database can proceed. The resulting performance effect is frequently unacceptable, ruling out the use of synchronous mirroring.

- The increased adoption of 100% SSD storage means that additional write latency is more likely to be noticed because performance expectations include hundreds of thousands of IOPS and submillisecond latency. Gaining the full benefits of using 100% SSDs can require revisiting the disaster recovery strategy.
- Datasets continue to grow in terms of bytes, creating challenges with ensuring sufficient bandwidth to sustain synchronous replication.
- Datasets also grow in terms of complexity, creating challenges with the management of large-scale synchronous replication.
- Cloud-based strategies frequently involve greater replication distances and latency, further precluding the use of synchronous mirroring.

NetApp® offers solutions that include both synchronous replication for the most exacting data recovery demands and asynchronous solutions that allow for better database performance and flexibility. In addition, NetApp technology integrates seamlessly with many third-party replication solutions, such as Oracle DataGuard and SQL Server AlwaysOn.

2.4 Retention Time

The final aspect of a data protection strategy is the data retention time, which can vary dramatically.

- A typical requirement is 14 days of nightly backups on the primary site and 90 days of backups stored on a secondary site.
- Many customers create standalone quarterly archives stored on different media.
- A constantly updated database might have no need for historical data, and backups need only be retained for a few days.

Regulatory requirements might require recoverability to the point of any arbitrary transaction in a 365-day window.

3 NetApp ONTAP Data Protection Fundamentals

3.1 Data Protection with NetApp Snapshot Copies

The foundation of NetApp ONTAP® data protection software is NetApp Snapshot® technology. The key values are as follows:

- **Simplicity.** A Snapshot copy is a read-only copy of the contents of a container of data at a specific point in time.
- **Efficiency.** Snapshot copies require no space at the moment of creation. Space is only consumed when data is changed.
- **Manageability.** A backup strategy based on Snapshot copies is easy to configure and manage because Snapshot copies are a native part of the storage OS. If the storage system is powered on, it is ready to create backups.
- **Scalability.** Up to 255 backups of a single container of files and LUNs can be preserved. For complex datasets, multiple containers of data can be protected by a single, consistent set of Snapshot copies.
- Performance is unaffected, whether a volume contains 250 Snapshot copies or none.

As a result, protecting a database running on ONTAP is simple and highly scalable. Database backups do not require movement of data, therefore a backup strategy can be tailored to the needs of the business rather than the limitations of network transfer rates, large number of tape drives, or disk staging areas.

3.2 Data Restoration with ONTAP SnapRestore

Rapid data restoration in ONTAP from a Snapshot copy is delivered by NetApp SnapRestore® technology. The key values are as follows:

- Individual files or LUNs can be restored in seconds, whether it is a 2TB LUN or a 4KB file.
- An entire container (a NetApp FlexVol® volume) of LUNs and/or files can be restored in seconds, whether it is 10GB or 100TB of data.

When a critical database is down, critical business operations are down. Tapes can break, and even restores from disk-based backups can be slow to transfer across the network. SnapRestore avoids these problems by delivering near instantaneous restoration of databases. Even petabyte-scale databases can be completely restored with just a few minutes of effort.

3.3 Data Replication and Disaster Recovery

Nearly every database requires data replication. At the most basic level, replication can mean a copy on tape stored offsite or database-level replication to a standby database. Disaster recovery refers to the use of those replica copies to bring a service online in the event of catastrophic loss of service.

ONTAP offers multiple replication options to address a variety of requirements natively within the storage array, covering a complete spectrum of needs. These options can include simple replication of backups to a remote site up to a synchronous, fully automated solution that delivers both disaster recovery and high availability in the same platform.

The primary ONTAP replication technologies applicable to databases are the NetApp SnapMirror® and NetApp SyncMirror® technologies. These are not add-on products; rather they are fully integrated into ONTAP and are activated by the simple addition of a license key. Storage-level replication is not the only option either. Database-level replication, such as with Oracle DataGuard or Microsoft SQL Server AlwaysOn, can also integrate into a data protection strategy based on ONTAP.

The right choice depends on the specific replication, recovery, and retention requirements.

ONTAP SnapMirror

SnapMirror is the NetApp asynchronous replication solution, ideally suited for protecting large, complicated, and dynamic datasets such as databases and their associated applications. Its key values are as follows:

- **Manageability.** SnapMirror is easy to configure and manage because it is a native part of the storage software. No add-on products are required. Replication relationships can be established in minutes and can be managed directly on the storage system.
- **Simplicity.** Replication is based on FlexVol volumes, which are containers of LUNs or files that are replicated as a single consistent group.
- **Efficiency.** After the initial replication relationship is established, only changes are replicated. Furthermore, efficiency features such as deduplication and compression are preserved, further reducing the amount of data that must be transferred to a remote site.
- **Flexibility.** Mirrors can be temporarily broken to allow testing of disaster recovery procedures, and then the mirroring can be easily reestablished with no need for a complete remirroring. Only the changed data must be applied to bring the mirrors back into sync. Mirroring can also be reversed to allow a rapid resync after the disaster concludes and the original site is back in service. Finally, read-write clones of replicated data are available for testing and development.

MetroCluster and SyncMirror

Synchronous replication in ONTAP is delivered by SyncMirror. At the simplest layer, SyncMirror creates two complete sets of RAID-protected data in two different locations. They could be in adjoining rooms within a data center, or they could be located many kilometers apart.

SyncMirror is fully integrated with ONTAP and operates just above the RAID level. Therefore, all the usual ONTAP features, such as Snapshot copies, SnapRestore, and NetApp FlexClone®, work seamlessly. It is still ONTAP, it just includes an additional layer of synchronous data mirroring.

A collection of ONTAP controllers managing SyncMirror data is called a NetApp MetroCluster™ configuration. The primary purpose of MetroCluster is to provide high availability access to synchronously mirrored data in a variety of typical and disaster recovery failure scenarios.

The key values of data protection with MetroCluster and SyncMirror are as follows:

- In normal operations, SyncMirror delivers guaranteed synchronous mirroring across locations. A write operation is not acknowledged until it is present on nonvolatile media on both sites.
- If connectivity between sites fails, SyncMirror automatically switches into asynchronous mode to keep the primary site serving data until connectivity is restored. When restored, it delivers rapid resynchronization by efficiently updating the changes that have accumulated on the primary site. Full reinitialization is not required.

SnapMirror is also fully compatible with systems based on SyncMirror. For example, a primary database might be running on a MetroCluster cluster spread across two geographic sites. This database can also replicate backups to a third site as long-term archives or for the creation of clones in a DevOps environment.

4 SnapCenter Software and Other Management Tools

The primary value of ONTAP in a database environment comes from the core ONTAP technologies such as instant Snapshot copies, simple SnapMirror replication, and efficient creation of FlexClone volumes. In some cases, simple configuration of these features within ONTAP will meet requirements, but more complicated needs require an orchestration layer.

The purpose of this TR is to explain the principles of data protection on ONTAP with a goal of making a database storage architecture "snapshot-friendly" and ready.

4.1 SnapCenter

SnapCenter is the flagship NetApp data protection product. At a very low level, it is similar to the SnapManager products in terms of how it executes database backups, but it was built from the ground up to deliver a single-pane-of-glass for data protection management on NetApp storage systems.

SnapCenter includes the basic functions such as Snapshot based backups and restores, SnapMirror and SnapVault replication, and other features required to operate at scale for large enterprises. These advanced features include an expanded role-based access control (RBAC) capability, RESTful APIs to integrate with third-party orchestration products, nondisruptive central management of SnapCenter plug-ins on database hosts, and a user interface designed for cloud-scale environments.

At present, it supports Oracle and SQL Server out-of-the-box. For custom needs, users can define their workflows with custom plug-ins. For more information, contact your NetApp representative or visit SnapCenter on the NetApp site.

4.2 SnapManager

The SnapManager products include SnapManager for Oracle and SnapManager for SQL server. They are used to manage database data protection on ONTAP storage systems in a manner similar to SnapCenter, but are based on an older architecture and are gradually being replaced with SnapCenter with most customers.

4.3 Snap Creator

NetApp Snap Creator Framework was originally developed to address requirements that were not covered by one of the SnapManager products. The framework allows users to create customized workflows and includes many plug-ins for products such as Oracle or MySQL. Its primary benefits are flexibility and simplicity, which allow for great scalability; however, it does not include the rich GUI of SnapCenter or the SnapManager products. Although it is possible to use Snap Creator for advanced workflows such as the creation of a fully operational database clone, this would require scripting, whereas SnapCenter includes full cloning automation with full support for these advanced workflows.

4.4 CommVault IntelliSnap for NetApp

CommVault IntelliSnap for NetApp integrates with ONTAP to provide data protection for Oracle, DB2, MySQL, and Microsoft SQL Server. This includes snapshot-based backups and restores, cloning using FlexClone, and SnapVault/SnapMirror replication. It also features the ability to work with traditional tape and direct-to-disk streaming backups to address backup needs across mixed environments. In addition, existing CommVault installations can be licensed for advanced IntelliSnap functionality.

5 ONTAP Platforms

ONTAP software is the foundation for advanced data protection and management. However, ONTAP only refers to software. There are several ONTAP hardware platforms from which to choose:

- ONTAP on All Flash FAS (AFF) and FAS
- NetApp Private Storage for Cloud
- ONTAP Select
- ONTAP Cloud

The key concept is that ONTAP is ONTAP. Some hardware options offer better performance, others offer lower costs, and some run within hyperscaler clouds. The core functions of ONTAP are unchanged, with multiple replication options available to bind different ONTAP platforms into a single solution. As a result, data protection and disaster recovery strategies can be built on real-world needs, such as performance requirements, capex/opex considerations, and overall cloud strategy. The underlying storage technology runs anywhere in any environment.

5.1 ONTAP with All Flash FAS and FAS Controllers

For maximum performance and control of data, ONTAP on a physical AFF or FAS controller remains the leading solution. This option is the standard on which thousands of customers have relied for more than 20 years. ONTAP delivers solutions for any environment, ranging from three mission-critical databases to 60,000-database service provider deployments, instant restores of petabyte-scale databases, and DBaaS involving hundreds of clones of a single database.

5.2 NetApp Private Storage for Cloud

NetApp introduced the NetApp Private Storage (NPS) option to address the needs of data-intensive workloads in the public cloud. Although many public cloud storage options exist, most of them have limitations in terms of performance, control, or scale. With respect to database workloads, the primary limitations are as follows:

- Many public cloud storage options do not scale to the IOPS levels required by modern database workloads in terms of cost, efficiency, or manageability.
- Even when the raw IOPS capabilities of a public cloud provider meet requirements, the I/O latencies are frequently unacceptable for database workloads. This has become even more true as databases have migrated to all-flash storage arrays, and businesses have begun to measure latency in terms of microseconds, not milliseconds.
- Although public cloud storage availability is good overall, it does not yet meet the demands of most mission-critical environments.
- Backup and recovery capabilities exist within public cloud storage services, but they generally cannot meet the zero RPO and near-zero RTO requirements of most databases. Data protection requires true instant snapshot-based backup and recovery, not streaming backup and recovery to and from elsewhere in a cloud.
- Hybrid cloud environments must move data between on-premises and cloud storage systems, mandating a common foundation for storage management.
- Many governments have strict data sovereignty laws that prohibit relocating data outside national borders.

NetApp Private Storage systems deliver maximum storage performance, control, and flexibility to public cloud providers, including Amazon AWS, Microsoft Azure, and IBM SoftLayer. This capability is delivered by AFF and FAS systems, including MetroCluster options, in data centers connected directly to public clouds. The full power of the hyperscaler compute layer can be used without the limitations of hyperscaler storage. Furthermore, NPS enables cloud-independent and multicloud architectures because the data, such as application binaries, databases, database backups, and archives, all remain wholly within the NPS system. There is no need to expend time, bandwidth, or money moving data between cloud providers.

Notably, some NetApp customers have used the NPS model on their own initiative. In many locations, high-speed access to one of the hyperscaler providers is readily available to customer data center facilities. In other cases, customers use a colocation facility that is already capable of providing high-speed access to hyperscaler cloud providers. This had led to the use of Amazon AWS, Azure, and SoftLayer as essentially on-demand, consumption-based sources of virtualized servers. In some cases, nothing has changed about the customers' day-to-day operations. They simply use the hyperscaler services as a more powerful, flexible, and cost-efficient replacement for their traditional virtualization infrastructure.

Options are also available for NPS as a service (NPSaaS). In many cases, the demands of database environments are substantial enough to warrant purchasing an NPS system at a colocation facility. However, in some cases, customers prefer to utilize both cloud servers and cloud storage as an operational expense rather than a capital expense. In these cases, they want to use storage resources purely as an as-needed, on-demand service. Several providers now offer NPS as a service for such customers.

5.3 ONTAP Select

ONTAP Select runs on a customer’s virtualization infrastructure and delivers ONTAP intelligence and data fabric connectivity to the drives inside of white box hardware. ONTAP Select allows ONTAP and guest operating systems to share the same physical hardware as a highly-converged infrastructure. The best practices for running Oracle on ONTAP are not affected. The primary consideration is performance, but ONTAP Select should not be underestimated.

An ONTAP Select environment does not match the peak performance of a high-end AFF system; however, most databases do not require 300K IOPS. Typical databases only require around 5K to 10K IOPS, a target that can be met by ONTAP Select. Furthermore, most databases are limited more by storage latency than storage IOPS, a problem that can be addressed by deploying ONTAP Select on SSD drives.

5.4 ONTAP Cloud

ONTAP Cloud is similar to ONTAP Select, except that it runs in a hyperscaler cloud environment, bringing intelligence and data fabric connectivity to hyperscaler storage volumes. The best practices for running Oracle on ONTAP are not affected. The primary considerations are performance and to a lesser extent, cost.

ONTAP Cloud is partially limited by the performance of the underlying volumes managed by the cloud provider. The result is more manageable storage, and sometimes the caching capability of ONTAP Cloud offers a performance improvement. However, there are always some limitations in terms of IOPS and latency due to the reliance on the public cloud provider. These limitations do not mean that database performance is unacceptable. It simply means that the performance ceiling is lower than options such as an actual physical AFF system. Furthermore, the performance of storage volumes offered by the various cloud providers that are utilized by ONTAP Cloud are continuously improving.

The prime use case for ONTAP Cloud is currently for development and testing work, but some customers have used ONTAP Cloud for production activity as well. One particularly notable report was the use of the Oracle Database In-Memory feature to mitigate storage performance limitations. This allows more data to be stored in RAM on the virtual machine hosting the database server, thus reducing performance demands on storage.

6 NetApp Snapshot

A Snapshot copy is the fundamental building block of data protection within ONTAP. It is a read-only image of the data within a FlexVol volume. It is atomic and preserves write order across all file and LUN data within the FlexVol volume. The term "consistency group" is used to refer to a grouping of storage objects that are managed as a consistent collection of data. A FlexVol volume is the basic consistency group available on ONTAP, and a Snapshot copy constitutes a consistency group backup.

Although many storage vendors offer snapshot technology, the Snapshot technology within ONTAP is unique and offers significant benefits to enterprise application and database environments:

- ONTAP Snapshot copies are part of the underlying Write-Anywhere File Layout (WAFL®). They are not an add-on or external technology. This simplifies management because the storage system *is* the backup system.
- ONTAP Snapshot copies do not affect performance, except for some edge cases such as when so much data is stored in Snapshot copies that the underlying storage system fills up.

ONTAP Snapshot copies scale better than competing technology. Customers can store 5, 50, or 250 Snapshot copies without affecting performance. The maximum number of Snapshot copies currently allowed in a volume is 255. If additional Snapshot copy retention is required, there are options to cascade the Snapshot copies to additional volumes.

7 Consistency Groups

The term "consistency group" refers to the ability of a storage array to manage multiple storage resources as a single image. For example, a database might consist of 10 LUNs. The array must be able to back up, restore, and replicate those 10 LUNs in a consistent manner. Restoration will not be possible if the images of the LUNs were not consistent at the point of backup. Replicating those 10 LUNs requires that all the replicas are perfectly synchronized with each other.

The term "consistency group" is not often used when discussing ONTAP because consistency has always been a basic function of the volume and aggregate architecture within ONTAP. Many other storage arrays manage LUNs or file systems as individual units. They could then be optionally configured as a "consistency group" for purposes of data protection, but this is an extra step in the configuration.

ONTAP has always been able to capture consistent local and replicated images of data. Although the various volumes on an ONTAP system are not usually formally described as a consistency group, that is what they are. A Snapshot copy of that volume is a consistency group image, restoration for that Snapshot copy is a consistency group restoration, and both SnapMirror and SnapVault offer consistency group replication.

7.1 Dependent Write Order

From a technical point of view, the key to a consistency group is preserving write order and, specifically, dependent write order. For example, a database writing to 10 LUNs writes simultaneously to all of them. Many writes are issued asynchronously, meaning that the order in which they are completed is unimportant and the actual order they are completed varies based on operating system and network behavior.

Some write operations must be present on disk before the database can proceed with additional writes. These critical write operations are called dependent writes. Subsequent write I/O depends on the presence of these writes on disk. Any snapshot, recovery, or replication of these 10 LUNs must make sure that dependent write order is guaranteed. File system updates are another example of write-order dependent writes. The order in which file system changes are made must be preserved or the entire file system could become corrupt.

Note: Some storage systems can be configured for "async" operations, meaning that they can acknowledge a write before it has been committed to persistent media. This can improve performance, but it virtually guarantees data loss in the event of a storage system crash or power failure. ONTAP cannot be configured like this; it always operates in a synchronous mode. If a write has been acknowledged by ONTAP, it has been stored in persistent media.

7.2 Consistency Group Snapshots

Consistency group snapshots (cg-snapshots) are an extension of the basic ONTAP Snapshot technology. A standard snapshot operation creates a consistent image of all data within a single volume, but sometimes it is necessary to create a consistent set of snapshots across multiple volumes and even across multiple storage systems. The result is a set of snapshots that can be used in the same way as a snapshot of just one individual volume. They can be used for local data recovery, replicated for disaster recovery purposes, or cloned as a single consistent unit.

The largest known use of cg-snapshots is for a database environment of approximately 1PB in size spanning 12 controllers. The cg-snapshots created on this system have been used for backup, recovery and cloning.

Most of the time, when a data set spans volumes and write order must be preserved, a cg-snapshot is automatically used by the chosen management software. There is no need to understand the technical details of cg-snapshots in such cases. However, there are situations in which complicated data protection requirements require detailed control over the data protection and replication process. Automation workflows or the use of custom scripts to call the cg-snapshot APIs are some of options. Understanding the best option and the role of cg-snapshot requires a more detailed explanation of the technology.

Creation of a set of cg-snapshots is a two-step process:

1. Establish write fencing on all target volumes.
2. Create snapshots of those volumes while in the fenced state.

Write fencing is established serially. This mean that as the fencing process is set up across multiple volumes, write I/O is frozen on the first volumes in the sequence as it continues to be committed to volumes that appear later. This might initially appear to violate the requirement for write order to be preserved, but that only applies to I/O that is issued asynchronously on the host and does not depend on any other writes.

For example, a database might issue a lot of asynchronous datafile updates and allow the OS to reorder the I/O and complete them according to its own scheduler configuration. The order of this type of I/O cannot be guaranteed because the application and operating system have already released the requirement to preserve write order.

As a counter example, most database logging activity is synchronous. The database does not proceed with further log writes until the I/O is acknowledged, and the order of those writes must be preserved. If a log I/O arrives on a fenced volume, it is not acknowledged and the application blocks on further writes. Likewise, file system metadata I/O is usually synchronous. For example, a file deletion operation must not be lost. If an operating system with an xfs file system deleted a file and the I/O that updated the xfs file system metadata to remove the reference to that file landed on a fenced volume, then the file system activity would pause. This guarantees the integrity of the file system during cg-snapshot operations.

After write fencing is set up across the target volumes, they are ready for snapshot creation. The snapshots need not be created at precisely the same time because the state of the volumes is frozen from a dependent write point of view. To guard against a flaw in the application creating the cg snapshots, the initial write fencing includes a configurable timeout in which ONTAP automatically releases the fencing and resumes write processing after a defined number of seconds. If all the snapshots are created before the timeout period lapses, then the resulting set of snapshots are a valid consistency group.

8 SnapRestore

SnapRestore enables instantaneous restoration of Snapshot data. The reason SnapRestore works so quickly and efficiently is due to the nature of a Snapshot copy, which is essentially a parallel read-only view of the contents of a volume at a specific point in time. The active blocks are the real blocks that can be changed, while the Snapshot copy is a read-only view into the state of the blocks that constitute the files and LUNs at the time the Snapshot copy was created.

ONTAP only permits read-only access to Snapshot data, but the data can be reactivated with SnapRestore. The Snapshot copy is reenabled as a read-write view of the data, returning the data to its prior state. SnapRestore can operate at the volume or the file level. The technology is essentially the same with a few minor differences in behavior.

8.1 Volume SnapRestore

Volume-based SnapRestore returns the entire volume of data to an earlier state. This operation does not require data movement, meaning that the restore process is essentially instantaneous, although the API or CLI operation might take a few seconds to be processed. Restoring 1GB of data is no more complicated or time-consuming than restoring 1PB of data. This capability is the primary reason many database customers migrate to ONTAP storage systems. It delivers an RTO measured in seconds for even the largest datasets.

One drawback to volume-based SnapRestore is caused by the fact that changes within a volume are cumulative over time. Therefore, each Snapshot copy and the active file data are dependent on the changes leading up to that point. Reverting a volume to an earlier state means discarding all the subsequent changes that had been made to the data. What is less obvious, however, is that this includes subsequently created Snapshot copies. This is not always desirable.

For example, a data retention SLA might specify 30 days of nightly backups. Restoring a database to a Snapshot copy created five days ago with volume SnapRestore would discard all the Snapshot copies created on the previous five days, violating the SLA.

There are a number of options available to address this limitation:

1. Data can be copied from a prior Snapshot copy, as opposed to performing a SnapRestore of the entire volume. This method works best with smaller datasets.
2. A Snapshot copy can be cloned rather than restored. The limitation to this approach is that the source Snapshot copy is a dependency of the clone. Therefore, it cannot be deleted unless the clone is also deleted or is split into an independent volume.
3. Use of file-based SnapRestore

8.2 File SnapRestore

File-based SnapRestore is a more granular snapshot-based restoration process. Rather than reverting the state of an entire volume, the state of an individual file or LUN is reverted. No Snapshot copies need to be deleted, nor does this operation create any dependency on a prior Snapshot copy. The file or LUN becomes immediately available in the active volume.

No data movement is required during a SnapRestore restore of a file or LUN. However, some internal metadata updates are required to reflect the fact that the underlying blocks in a file or LUN now exist in both a Snapshot copy and the active volume. There should be no effect on performance, but this process blocks the creation of Snapshot copies until it is complete. The processing rate is approximately 5GBps (18TB/hour) based on the total size of the files restored.

9 SnapMirror

ONTAP offers several different replication technologies, but the most flexible is SnapMirror, a volume-to-volume asynchronous mirroring option.

As mentioned before, a FlexVol volume is the basic unit of management for Snapshot-based backups and SnapRestore-based recovery. A FlexVol volume is also the basic unit for SnapMirror-based replication. The first step is establishing the baseline mirror of the source volume to the destination volume. After this mirror relationship is initialized, all subsequent operations are based on replication of the changed data alone.

From a database recovery perspective, the key values of SnapMirror are as follows:

- SnapMirror operations are simple to understand and can be easily automated.
- A simple update of a SnapMirror replica requires that only the delta changes are replicated, reducing demands on bandwidth and allowing more frequent updates.
- SnapMirror is highly granular. It is based on simple volume-to-volume relationships, allowing for the creation of hundreds of independently managed replicas and replication intervals. Replication does not need to be one-size-fits-all.
- The mirroring direction can be easily reversed while preserving the ability to update the relationship based on the changes alone. This delivers rapid failback capability after the primary site is restored to service after a disaster such as a power failure. Only the changes must be synchronized back to the source.

- Mirrors can easily be broken and efficiently resynced to permit rehearsal of disaster recovery procedures.
- SnapMirror operating in full block-level replication mode replicates not just the data in a volume, but also the Snapshot copies. This capability provides both a copy of the data and a complete set of backups on the disaster recovery site.

SnapMirror operating in version-flexible mode allows for replication of specific Snapshot copies, permitting different retention times at the primary and secondary sites.

10 MetroCluster

MetroCluster is a synchronous replication solution for sites up to 300 km apart.

For a complete description of using Oracle on MetroCluster, see TR-4592. Although this TR primarily targets Oracle and Oracle RAC databases, the basic principles are applicable to most relational databases.

11 ONTAP Logical Architecture

There are two fundamental requirements for any storage system: make sure that data is protected and make sure that data is available. A complete explanation of ONTAP data protection technologies is beyond the scope of this document, but a review of the layers is required to fully understand what happens in various fault scenarios.

11.1 Data Protection

Logical data protection within ONTAP consists of three key requirements:

- Data must be protected against data corruption.
- Data must be protected against drive failure.
- Changes to data must be protected against loss

These three needs are discussed in the following sections.

Network Corruption: Checksums

The most basic level of data protection is the checksum, which is a special error-detecting code stored alongside data. Corruption of data during network transmission is detected with the use of a checksum and, in some instances, multiple checksums.

For example, an FC frame includes a form of checksum called a cyclic redundancy check (CRC) to make sure that the payload is not corrupted in transit. The transmitter sends both the data and the CRC of the data. The receiver of an FC frame recalculates the CRC of the received data to make sure that it matches the transmitted CRC. If the newly computed CRC does not match the CRC attached to the frame, the data is corrupt and the FC frame is discarded or rejected. An iSCSI I/O operation includes checksums at the TCP/IP and Ethernet layers, and, for extra protection, it can also include optional CRC protection at the SCSI layer. Any bit corruption on the wire is detected by the TCP layer or IP layer, which results in retransmission of the packet. As with FC, errors in the SCSI CRC result in a discard or rejection of the operation.

Drive Corruption: Checksums

Checksums are also used to verify the integrity of data stored on drives. Data blocks written to drives are stored with a checksum function that yields an unpredictable number that is tied to the original data. When data is read from the drive, the checksum is recomputed and compared to the stored checksum. If it does not match, then the data has become corrupt and must be recovered by the RAID layer.

Data Corruption: Lost Writes

One of the most difficult types of corruption to detect is a lost or a misplaced write. When a write is acknowledged, it must be written to the media in the correct location. In-place data corruption is relatively easy to detect by using a simple checksum stored with the data. However, if the write is simply lost, then the prior version of data might still exist and the checksum would be correct. If the write is placed at the wrong physical location, the associated checksum would once again be valid for the stored data, even though the write has destroyed other data.

The solution to this challenge is as follows:

- A write operation must include metadata that indicates the location where the write is expected to be found.
- A write operation must include some sort of version identifier.

When ONTAP writes a block, it includes data on where the block belongs. If a subsequent read identifies a block, but the metadata indicates that it belongs at location 123 when it was found at location 456, then the write has been misplaced.

Detecting a wholly lost write is more difficult. The explanation is very complicated, but essentially ONTAP is storing metadata in a way that a write operation results in updates to two different locations on the drives. If a write is lost, a subsequent read of the data and associated metadata shows two different version identities. This indicates that the write was not completed by the drive.

Lost and misplaced write corruption is exceedingly rare, but, as drives continue to grow and datasets push into exabyte scale, the risk increases. Lost write detection should be included in any storage system supporting database workloads.

Drive Failures: RAID, RAID DP, and RAID-TEC

If a block of data on a drive is discovered to be corrupt, or the entire drive fails and is wholly unavailable, the data must be reconstituted. This is done in ONTAP by using parity drives. Data is striped across multiple data drives, and then parity data is generated. This is stored separately from the original data.

ONTAP originally used RAID 4, which uses a single parity drive for each group of data drives. The result was that any one drive in the group could fail without resulting in data loss. If the parity drive failed, no data was damaged and a new parity drive could be constructed. If a single data drive failed, the remaining drives could be used with the parity drive to regenerate the missing data.

When drives were small, the statistical chance of two drives failing simultaneously was negligible. As drive capacities have grown, so has the time required to reconstruct data after a drive failure. This has increased the window in which a second drive failure would result in data loss. In addition, the rebuild process creates a lot of additional I/O on the surviving drives. As drives age, the risk of the additional load leading to a second drive failure also increases. Finally, even if the risk of data loss did not increase with the continued use of RAID 4, the consequences of data loss would become more severe. The more data that would be lost in the event of a RAID-group failure, the longer it would take to recover the data, extending business disruption.

These issues led NetApp to develop the NetApp RAID DP® technology, a variant of RAID 6. This solution includes two parity drives, meaning that any two drives in a RAID group can fail without creating data loss. Drives have continued to grow in size, which eventually led NetApp to develop the NetApp RAID-TEC™ technology, which introduces a third parity drive.

Some historical database best practices recommend the use of RAID-10, also known as striped mirroring. This offers less data protection than even RAID DP because there are multiple two-disk failure scenarios, whereas in RAID DP there are none.

There are also some historical database best practices that indicate RAID-10 is preferred to RAID-4/5/6 options due to performance concerns. These recommendations sometimes refer to a RAID penalty.

Although these recommendations are generally correct, they are inapplicable to the implementations of RAID within ONTAP. The performance concern is related to parity regeneration. With traditional RAID implementations, processing the routine random writes performed by a database requires multiple disk reads to regenerate the parity data and complete the write. The penalty is defined as the additional read IOPS required to perform write operations.

ONTAP does not incur a RAID penalty because writes are staged in memory where parity is generated and then written to disk as a single RAID stripe. No reads are required to complete the write operation.

In summary, when compared to RAID 10, RAID DP and RAID-TEC deliver much more usable capacity, better protection against drive failure, and no performance sacrifice.

Hardware Failure Protection: NVRAM

Any storage array servicing a database workload must service write operations as quickly as possible. Furthermore, a write operation must be protected from loss from an unexpected event such as a power failure. This means any write operation must be safely stored in at least two locations.

AFF and FAS systems rely on NVRAM to meet these requirements. The write process works as follows:

1. The inbound write data is stored in RAM.
2. The changes that must be made to data on disk are journaled into NVRAM on both the local and partner node. NVRAM is not a write cache; rather it is a journal similar to a database redo log. Under normal conditions, it is not read. It is only used for recovery, such as after a power failure during I/O processing.
3. The write is then acknowledged to the host

The write process at this stage is complete from the application point of view, and the data is protected against loss because it is stored in two different locations. Eventually, the changes are written to disk, but this process is out-of-band from the application point of view because it occurs after the write is acknowledged and therefore does not affect latency. This process is once again similar to database logging. A change to the database is recorded in the redo logs as quickly as possible, and the change is then acknowledged as committed. The updates to the datafiles occur much later and do not directly affect the speed of processing.

In the event of a controller failure, the partner controller takes ownership of the required disks and replays the logged data in NVRAM to recover any I/O operations that were in-flight when the failure occurred.

Site and Shelf Failure Protection: SyncMirror and Plexes

SyncMirror is a mirroring technology that enhances, but does not replace, RAID DP or RAID-TEC. It mirrors the contents of two independent RAID groups. The logical configuration is as follows:

- Drives are configured into two pools based on location. One pool is composed of all drives on site A, and the second pool is composed of all drives on site B.
- A common pool of storage, known as an aggregate, is then created based on mirrored sets of RAID groups. An equal number of drives is drawn from each site. For example, a 20-drive SyncMirror aggregate would be composed of 10 drives from site A and 10 drives from site B.
- Each set of drives on a given site is automatically configured as one or more fully redundant RAID-DP or RAID-TEC groups, independent of the use of mirroring. This provides continuous data protection, even after the loss of a site.

Figure 1) SyncMirror.

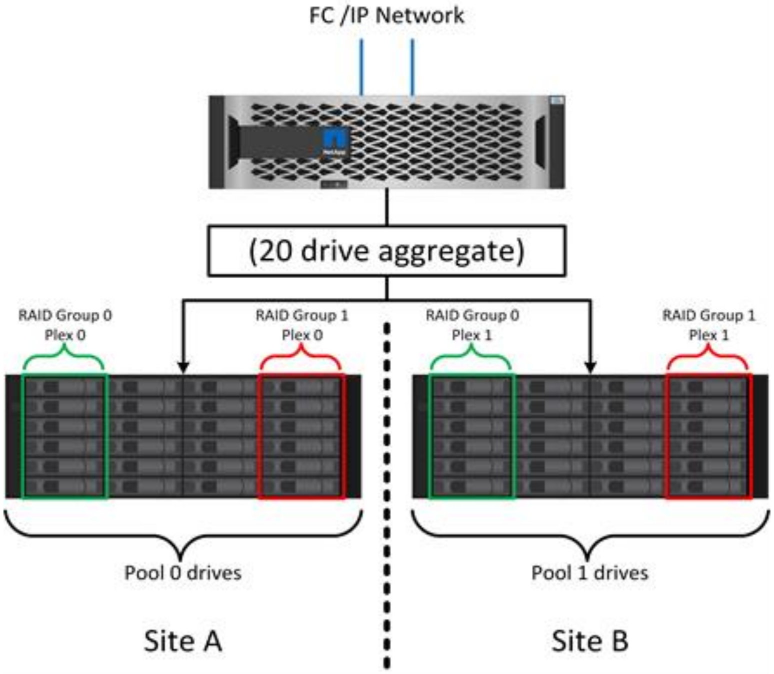


Figure 1 illustrates a sample SyncMirror configuration. A 24-drive aggregate was created on the controller with 12 drives from a shelf allocated on Site A and 12 drives from a shelf allocated on Site B. The drives were grouped into two mirrored RAID groups. RAID Group 0 includes a 6-drive plex on Site A mirrored to a 6-drive plex on Site B. Likewise, RAID Group 1 includes a 6-drive plex on Site A mirrored to a 6-drive plex on Site B.

SyncMirror is normally used to provide remote mirroring with MetroCluster systems, with one copy of the data at each site. On occasion, it has been used to provide an extra level of redundancy in a single system. In particular, it provides shelf-level redundancy. A drive shelf already contains dual power supplies and controllers and is overall little more than sheet metal, but in some cases the extra protection might be warranted. For example, one NetApp customer has deployed SyncMirror for a mobile real-time analytics platform used during automotive testing. The system was separated into two physical racks suppled supplied by independent power feeds from independent UPS systems.

11.2 High Availability

A complete description of ONTAP high availability features is beyond the scope of this document. However, as with data protection, a basic understanding of this functionality is important when designing a database infrastructure.

HA Pairs

The basic unit of high availability is the HA pair. Each pair contains redundant links to support replication of NVRAM data. NVRAM is not write cache. The RAM inside a controller serves as the write cache. The purpose of NVRAM is to temporarily journal data as a safeguard against unexpected system failure. In this respect, it is similar to a database redo log.

Both NVRAM and a database redo log are used to store data quickly, allowing changes to data to be committed as quickly as possible. The update to the persistent data on drives (or datafiles) does not take place until later during a process called a checkpoint on both ONTAP and most databases platforms. Neither NVRAM data nor database redo logs are read during normal operations.

If a controller fails abruptly, there are likely to be pending changes stored in NVRAM that have not yet been written to the drives. The partner controller detects the failure, take control of the drives, and applies the required changes that have been stored in NVRAM.

Takeover and Giveback

Takeover and giveback refers to the process of transferring responsibility for storage resources between nodes in an HA pair. There are two aspects to takeover and giveback:

- Management of the network connectivity that allows access to the drives
- Management of the drives themselves

Network interfaces supporting CIFS and NFS traffic are configured with both a home and failover location. A takeover includes moving the network interfaces to their temporary home on a physical interface located on the same subnet(s) as the original location. A giveback includes moving the network interfaces back to their original locations. The exact behavior can be tuned as required.

Network interfaces supporting SAN block protocols such as iSCSI and FC are not relocated during takeover and giveback. Instead, LUNs should be provisioned with paths that includes a complete HA pair which results in a primary path and a secondary path.

Note: Additional paths to additional controllers can also be configured to support relocating data between nodes in a larger cluster, but this is not part of the HA process.

The second aspect of takeover and giveback is the transfer of disk ownership. The exact process depends on multiple factors including the reason for the takeover/giveback and the command line options issued. The goal is to perform the operation as efficiently as possible. Although the overall process might appear to require several minutes, the actual moment in which ownership of the drive is transitioned from node to node can generally be measured in seconds.

Takeover Time

Host I/O experiences a short pause in I/O during takeover and giveback operations, but there should not be application disruption in a correctly configured environment. The actual transition process in which I/O is delayed is generally measured in seconds, but the host might require additional time to recognize the change in data paths and resubmit I/O operations.

The nature of the disruption depends on the protocol:

- A network interface supporting NFS and CIFS traffic issues an ARP (Address Resolution Protocol) request to the network after the transition to a new physical location. This causes the network switches to update their MAC address tables and resume processing I/O. Disruption in the case of planned takeover and giveback is usually measured in seconds and in many cases is not detectable. Some networks might be slower to fully recognize the change in network path, and some OSs might queue up a lot of I/O in a very short time that must be retried. This can extend the time required to resume I/O.
- A network interface supporting SAN protocols does not transition to a new location. A host OS must change the path or paths in use. The pause in I/O observed by the host depends on multiple factors. From a storage system point of view, the period where I/O cannot be served is just a few seconds. However, different host OSs might require additional time to allow an I/O to time out before retry. Newer OSs are better able to recognize a path change much more quickly, but older OSs typically require up to 30 seconds to recognize a change.

The expected takeover times during which the storage system cannot serve data to a database environment are shown in Table 1.

Table 1) Expected takeover times.

	NAS	SAN Optimized OS	SAN
Planned Takeover	15sec	2sec-10sec	2sec-10sec
Unplanned Takeover	30sec	2sec-15sec	30sec

12 Local Database Data Protection Architecture

The right database data protection architecture depends on the business requirements surrounding data retention, recoverability, and tolerance for disruption during various events.

For example, consider the number of databases in scope. Building a backup strategy for a single database and ensuring compliance with typical SLAs is fairly straightforward because there are not many objects to manage. There are only one set of datafiles and one set of log files. As the number of databases increase, monitoring becomes more complicated and database administrators (DBAs) might be forced to spend an increasing amount of time addressing backup failures. As a database environment reaches cloud and service provider scales, a wholly different approach is needed.

Database size also affects strategy. Many options exist for backup and recovery with a 100GB database because the data set is so small. Simply copying the data from backup media with traditional tools typically delivers a sufficient RTO for recovery. A 100TB database normally needs a completely different strategy unless the RTO allows for a multiday outage, in which case a traditional copy-based backup and recovery procedure might be acceptable.

Finally, there are factors outside the backup and recovery process itself. For example, are the databases supporting critical production activities, making recovery a rare event that is only performed by skilled DBAs? Alternatively, are the databases part of a large development environment in which recovery is a frequent occurrence and managed by a generalist IT team?

These considerations affect the choice of data protection strategy. The following sections explain the basic principles that apply to relational database platforms.

12.1 Is a Snapshot a Backup?

One commonly raised objection to the use of snapshots as a data protection strategy is the fact that the "real" data and the snapshot data are located on the same drives. Loss of those drives would result in the loss of both the primary data and the backup.

This is a valid concern. Local snapshots are used for day-to-day backup and recovery needs, and in that respect the snapshot is a backup. Close to 99% of all recovery scenarios in NetApp environments rely on snapshots to meet even the most aggressive RTO requirements.

Local snapshots should, however, never be the only backup strategy, which is why NetApp offers technology such as SnapMirror and SnapVault replication to quickly and efficiently replicate snapshots to an independent set of drives. In a properly architected solution with snapshots plus snapshot replication, the use of tape can be minimized to perhaps a quarterly archive or eliminated entirely.

12.2 Backup and Recovery Overview

The basic options and their benefits and limitations for local database protection are summarized in Table 2. Note that this table does not address synchronous mirroring data protection. For this requirement, see the NetApp MetroCluster documentation including TR-4592: Oracle on MetroCluster.

Table 2) Local data protection overview.

	Consistency Group	Log Backup and Replay
Local recovery RPO	One hour (15 minutes possible)	Seconds
Local recovery RTO	Seconds	Minutes
Scalability	Best option for large numbers of databases for which a high RPO is acceptable	Maximum flexibility for very large databases

The following sections explain these options in detail.

12.3 Consistency Group Backups

A consistency group backup involves capturing the state of a database (or multiple databases) and any associated applications at a single atomic point in time. This includes all database components, such as datafiles, log files, and other files directly associated with the database. This works with almost all relational database products, including Oracle RDBMS, Microsoft SQL Server, SAP HANA, PostgreSQL, MySQL, and MariaDB.

Creation of a snapshot of an entire database environment is essentially simulating a crash, which is why such backups are frequently called crash-consistent backups. There are sometimes concerns with support for recovery scenarios, but it is important to understand that no recovery procedure is required. When the database starts up after restoring a consistency group backup, it performs the usual log recovery process to replay any I/O that was in-flight at the point of the backup. The database then starts.

Essentially, any database that can withstand a power failure or server crash without data corruption can be protected in this way. The fact that this works can also be demonstrated by the huge number of databases protected with synchronous and asynchronous mirroring products from many different vendors. If a disaster suddenly strikes the primary site, then the replica site contains a consistent image of the original database at the moment the disaster occurred. Once again, no special recovery procedure is required. Starting the database using the surviving copy of the database results in automatic log replay and the database then opens.

The RPO for this approach is usually limited to the point of the backup. As a general rule, the minimum RPO for single-volume database snapshots is one hour. For example, 48 hourly snapshots plus another 30 days of nightly snapshots is reasonable and would not require the storage of an excessive number of snapshots. An RPO lower than one hour becomes more difficult to achieve and it is not recommended without first consulting NetApp Professional Services to understand the environment, scale, and data protection requirements.

The RTO can usually be measured in seconds. A database is shut down, the volume is restored, and the database is restarted. A small mount of log replay occurs and the database is then online.

The retention time is tied to the chosen RPO. Maintaining a granular RPO in combination with a long retention time results in a large number of snapshots and could exceed the limits of the storage platform. See the prior paragraph explaining RPO limitations for additional detail.

The simplest approach is to place all the files or LUNS in a single volume consistency group, which allows a snapshot creation to be scheduled directly in ONTAP. Where a database must span volumes, a consistency group snapshot copy (cg-snapshot) is required. Both SnapCenter and the previous generation Snap Center are capable of creating a simple consistency group snapshot on a defined list of volumes. Both also includes scheduling, pre/post operation scripting abilities, and replication management.

Where a database must span volumes, a consistency group snapshot copy (cg-snapshot) is required. The most common software used to create consistent snapshots is Snap Creator, which is available at no charge for any controller with an active support contract. Snap Creator also includes scheduling, pre/post operation scripting abilities, and replication management. Products like the SnapCenter Plug-in for Oracle Database also natively perform cg-snapshots when required by the underlying data set. Lastly, cg-snapshots can be easily scripted by using the ONTAP Software Development Kit with a variety of scripting languages.

12.4 Log Backup and Replay

The log-based approach to backups is the best option for important databases that require the lowest possible RPO with point-in-time recoverability. The process should be familiar to most DBAs because it is based on traditional tape or file-based backup processes. The difference is that a snapshot replaces the process of copying the datafiles. In addition to being a nearly instantaneous process, it eliminates the load resulting from data movement on the database server, the storage system, and the network.

The exact backup process for a given database platform is described in the following sections, but it usually follows the same procedure:

1. The database is made ready for a backup procedure. The procedures vary.
2. A snapshot is created for the datafiles. If the datafiles span volumes, a consistency group snapshot might be required.
3. A snapshot of the log files is created.

The result is a set of snapshots containing:

- A snapshot containing a recoverable image of the datafiles.
- A snapshot of the log files required to make the database consistent.

The RPO of this approach is zero under normal situations. Most database recovery situations result from user or application errors that damage the database, or less likely an actual corruption in the database. Recovery requires restoring the datafiles only and then using the log files that are still present on disk to bring the database to the desired point. This point is the current state for an RPO of zero.

If the log files are damaged as well, then an increased frequency of log file snapshots can minimize data loss. It is impossible to completely eliminate the possibility of data loss from a rogue administrator aggressively trying to delete files, but the damage can be minimized.

For example, if an `rm -rf /` deleted both the datafiles and the log files, then both snapshots need to be recovered. If the snapshot frequency of the log files was set at one hour, then the RPO in this near-disaster situation is one hour. It is difficult to match this level of data protection without a technology like snapshots that does not require a lot of data movement.

The RTO is effectively controlled by the frequency of the datafile snapshots. For example, if datafile snapshots were created every 24 hours, then then worst-case RTO scenario would be a failure 23 hours and 59 minutes after the previous snapshot. Nearly 24 hours of log files would have to be applied to the backup to fully recover the database. This could require anything from five minutes to 24 hours to complete, depending on the volume of logs generated and the particular relational database management system used. If the time required to replay data logs is unacceptable, the datafile snapshot frequency can be increased.

There are two aspects to the retention time because there are two independently controlled backups, the full database backup and the log file backups. In general, databases require point-in-time recoverability for a limited time, but point-of-the-backup recovery is broader. As a typical example, a database might be backed up nightly, with those nightly snapshots being retained for 90 days. In addition, log files might be retained for seven days. The result is a database with 90 days of retention time, but specific point-in-time recovery is only possible within the immediately prior seven-day window.

13 Replication and Disaster Recovery Architecture

Table 3 addresses remote data protection, for which data is replicated to a remote site for the purposes of secure offsite storage and disaster recovery. Note that these tables do not address synchronous mirroring data protection. For this requirement, see the NetApp MetroCluster documentation including TR-4592 Oracle on MetroCluster.

Table 3) Replication and disaster recovery.

	Consistency Group	Log Replication	Database Replication
Disaster recovery RPO	One hour (15 minutes possible)	Seconds	Zero to minutes
Disaster recovery RTO	Seconds	Minutes	Seconds
Scalability	Best option for large numbers of databases for which a high RPO is acceptable	Maximum flexibility for very large databases	Good for small numbers of databases with low RPO, but scales poorly

Consistency group replication is the process of replicating a consistency group backup. The consistency group must include all database components, including datafiles, log files, and other files directly associated with the database. It can also include application data.

The RPO is limited by the available network bandwidth and the total size of the databases being protected. After the initial baseline transfer is created, the updates are only based on the changed data, which typically is a low percentage of the total database size. As a general principle, updating a database once per hour is achievable. There are limitations based on the available bandwidth.

For example, a 10TB database with a 10% weekly change rate averages approximately 6GB per hour of total changes. With 10Gb of connectivity, this database requires approximately six minutes to transfer. The change rate varies with fluctuation in the database change rate, but overall a 15-minute update interval and thus a 15-minute RPO should be achievable. If there are 100 such databases, then 600 minutes is required to transfer the data. Therefore, an RPO of one-hour is not possible. Likewise, a replica of a single database 100TB in size with a 10% weekly change rate cannot be updated reliably in one hour.

Additional factors can affect replication, such as replication overhead and limitations on the number of concurrent replication operations. However, overall planning for a single-volume replication strategy can be based on available bandwidth, and a replication RPO of one hour is generally achievable. An RPO lower than one hour becomes more difficult to achieve and should only be performed after consulting NetApp Professional Services. In some cases, 15 minutes is feasible with very good site-to-site network connectivity. However, overall, when an RPO below one hour is required, the multi-volume log replay architecture yields better results.

The RTO with consistency group replication in a disaster recovery scenario is excellent, typically measured in seconds from a storage point of view. The most straightforward approach is to simply break the mirror, and the database is ready to be started. Database startup time is typically about 10 seconds, but very large databases with a lot of logged transactions could take a few minutes.

The more important factor in determining RTO is not the storage system but rather the application and the host operating system upon which it runs. For example, the replicated database data can be made available in a second or two, but this only represents the data. There must also be a correctly configured operating system with application binaries that is available to use the data.

In some cases, customers have prepared disaster recovery instances ahead of time with the storage prediscovered on operating systems. In these cases, activating the disaster recovery scenario can require nothing more than breaking a mirror and starting the database server. In other cases, the OS and associated applications might be mirrored alongside the database as an ESX virtual machine disk (VMDK). In these cases, the RPO is determined by how much a customer has invested in automation to boot that VMDK so that the database can be started.

The retention time is controlled in part by the snapshot limit. For example, volumes in ONTAP have a limit of 255 Snapshot copies. In some cases, customers have multiplexed replication to increase the limit. For example, if 500 days of backups are required, a source can be replicated to two volumes with updates occurring on alternate days. This requires an increase in the initial space required, but it still represents a much more efficient approach than a traditional backup system, which involves multiple full backups.

Single-Volume Consistency Group

The simplest approach is to place all the files or LUNS in a single volume consistency group, which allows SnapMirror and SnapVault updates to be scheduled directly on the storage system. No external software is required.

Multi-Volume Consistency Group

When a database must span volumes, a consistency group snapshot (cg-snapshot) is required. Once again, the most common software used to replicate consistent snapshots is Snap Creator Framework. Snap Creator also includes scheduling, pre/post operation scripting abilities, and replication management. Products like SnapCenter natively perform cg-snapshots when required by the underlying data set.

There is also one additional consideration on the use of multivolume, consistent snapshots for disaster recovery purposes. When performing an update of multiple volumes, it is possible that a disaster could occur while a transfer is still in progress. The result would be a set of volumes that are not consistent with one another. If this happened, some of the volumes must be restored to an earlier snapshot state to deliver a database image that is crash-consistent and ready for use.

13.1 Log Replication

The log replication approach is the best option for important databases that require the lowest possible RPO with point-in-time recoverability. It is also more bandwidth-efficient because only the log files need to be replicated at a short interval to preserve the low RPO. The process is essentially a backup procedure in which the datafiles are separated from the log files. The datafiles and the log files are then replicated on different schedules.

The basic process is the same as performing a local backup:

1. The database is made ready for a backup procedure. The procedures vary.
2. A snapshot is created for the datafiles. If the datafiles span volumes, a consistent group snapshot might be required.
3. A snapshot of the log files is created.

The following snapshots types are created:

- A snapshot containing a recoverable image of the datafiles.
- A snapshot of the log files required to make the database consistent.

The replication schedule is then set independently and controls the RPO and RTO:

- The RPO is controlled by the frequency of log file updates.
- The RTO is controlled by the frequency of datafile updates.

For example, consider a 100TB database with an RPO of 15 minutes and RTO of one hour. A typical configuration updates the datafile replica once per day and updates the log file replica every 15 minutes. In the event of a disaster, the mirrors are broken and all available logs are replayed. The worst-case scenario is a disaster 23 hours and 59 minutes after the previous datafile update. There would be 23 hours and 45 minutes of logs to be replayed, and 15 minutes of unreplicated log data would be lost.

The RPO of this approach is generally limited by available bandwidth. An RPO of one hour is almost always achievable, even with extremely large databases, and 15 minutes is feasible with a good network infrastructure. Replication at intervals below 15 minutes is possible, but tends to be less reliable because of normal fluctuation of database log generation. It might be possible to replicate every 5 minutes much of the time, but there are times when the volume of log data written in between updates cannot be moved in just 5 minutes.

The RTO is effectively controlled by the frequency of the datafile updates. For example, if datafile snapshots are updated every 24 hours, then then worst-case RTO scenario would be a failure 23 hours and 59 minutes after the previous backup. Nearly 24 hours of log files would have to be applied to the backup to fully recover the database. This could require anything from 5 minutes to 24 hours to complete, depending on the volume of logs generated and the relational database management system used. If the time required to replay data logs is unacceptable, the datafile could be decreased from 24 hours to 12 hours.

There are two aspects to the retention time because there are two independently controlled backups, the full database backup and the log file backups. In general, databases require point-in-time recoverability for a limited time, but point-of-the-backup recovery is broader. As a typical example, a database might be backed up nightly, with those nightly backups being retained for 90 days. In addition, log files might be retained for seven days. The result is a database with 90 days of retention time, but specific point-in-time recovery is only possible within the prior seven-day window.

13.2 Disaster Recovery: Activation

NFS

The process of activating the disaster recovery copy depends on the type of storage. With NFS, the file systems can be premounted on the disaster recovery server. They are in a read-only state and become read-write when the mirror is broken. This delivers an extremely low RPO, and the overall disaster recovery process is more reliable because there are fewer parts to manage.

SAN

Activating SAN configurations in the event of disaster recovery become more complicated. The simplest option is generally to temporarily break the mirrors and mount the SAN resources, including steps such as discovering LVM configuration (including application-specific features such as Oracle Automatic Storage Management [ASM]), and adding entries to /etc/fstab.

The result is that the LUNs device paths, volume groups names, and other device paths are known to the target server. Those resources can then be shut down, and afterward the mirrors can be restored. The result is a server that is in a state that can rapidly bring the database storage online. The steps to activate volumes groups, mount file systems, or ASM instances are easily automated in the same script that starts the database itself.

Care must be taken to make sure that the disaster recovery environment is up to date. For example, new LUNs are likely to be added to the source server, which means the new LUNs must be prediscovered on the destination to make sure that the disaster recovery plan works as expected.

14 Microsoft SQL Server

Two approaches to online database backups exist. Some databases allow files to be backed up while the files are fully online and being updated. Log files can then be used during the restore process to make the database consistent. Other databases require a fully consistent set of datafiles to be created by a backup application. Microsoft SQL Server uses the second approach, which means there are fewer options for local data protection and disaster recovery strategies.

14.1 Consistency Group Backup

Placing an entire database in a single volume, including the system databases, tempdb, and transaction logs, is a valid backup, restore, and replication method. However, the RPO is limited to the point of the backup itself. It is suitable for an RPO of one hour or greater. If a database spans volumes, cg-snapshots are required using one of the tools discussed previously.

As an example, the entire database can be in a single volume with the following snapshot schedule:

- 72 hourly snapshots
- 30 nightly snapshots
- 12 monthly snapshots

This delivers an RPO of one hour over the rolling period of the preceding 72 hours, plus additional nightly and monthly backups. Multiple databases or application files can also be included in the single volume or set of cg-snapshots to deliver consistent backups across a larger environment.

14.2 Backups with Transaction Log Replay

As stated previously, recovery of an SQL Server database requires a set of consistent datafiles. This can be done by using the native SQL Server backup command, but it can also be done with a snapshot. This requires the use of a backup application that integrates with Volume Shadow Copy Service (VSS). This is a framework created by Microsoft to enable third-party snapshot integration with Microsoft products.

SQL Server includes a process called SQL Writer, which coordinates VSS services on behalf of SQL Server. During a backup operation, the backup application makes a call to the SQL writer to prepare the database for backup, which results in flushing buffers to make the database files consistent. Updates are then frozen for a brief time. After the database is quiesced, the backup application issues the commands to create storage-side snapshots. The recovery process is similar. The application calls the SQL writer to prepare the database for restoration. When ready, the application restores the required files to the appropriate location.

Multiple products are available to drive this process. SnapCenter Plug-in for SQL Server is the leading product from NetApp, and there are many other third-party backup applications that integrate with ONTAP.

Consult the documentation of the backup application for best practices on file system layouts.

14.3 Consistency Group Disaster Recovery

Consistency group replication can be as simple as scheduling replication of a single volume SnapMirror. This includes system databases, all datafiles, and log files. Every SnapMirror update results in a new copy of the database at the destination site that is consistent and ready for activation by breaking the mirror.

When a database must span volumes, a consistency group snapshot (cg-snapshot) is required. See the section “Log Replication” for additional information on managing cg-snapshots.

An additional benefit of this strategy when used with SnapMirror in block-level replication mode is complete replication of all snapshots on the source storage system. The full complement of backups is replicated in addition to the disaster recovery copy.

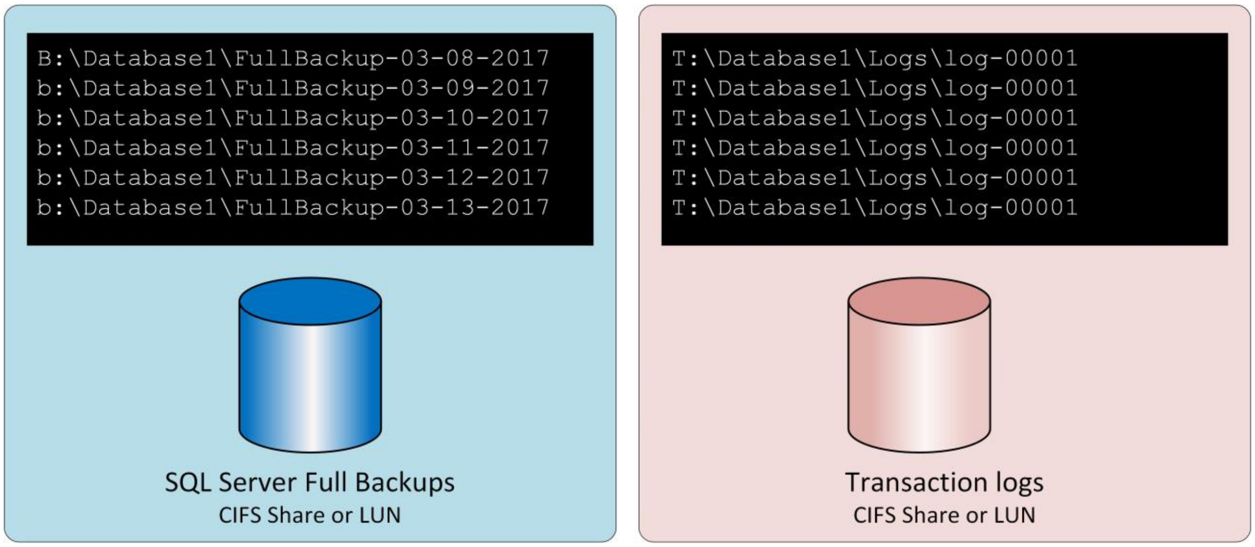
14.4 AlwaysOn

The most common disaster recovery technology for SQL Server databases is using the native capabilities of SQL AlwaysOn availability groups to maintain and manage replicas on a remote site. See the official Microsoft documentation for details and best practices.

14.5 Disaster Recovery with Log Replay

One simple option for disaster recovery with a good RPO is to replicate a standard SQL backup and replicate the transaction logs at a more frequent interval. For example, consider the two-volume layout depicted in Figure 2.

Figure 2) Two-volume layout.



```
B:\Database1\FullBackup-03-08-2017
b:\Database1\FullBackup-03-09-2017
b:\Database1\FullBackup-03-10-2017
b:\Database1\FullBackup-03-11-2017
b:\Database1\FullBackup-03-12-2017
b:\Database1\FullBackup-03-13-2017
```

```
T:\Database1\Logs\log-00001
T:\Database1\Logs\log-00001
T:\Database1\Logs\log-00001
T:\Database1\Logs\log-00001
T:\Database1\Logs\log-00001
T:\Database1\Logs\log-00001
```

Independently scheduling replication on the transaction logs allows the RPO to be adjusted. For example, the full backup volume can be replicated at 24-hour intervals, while the transaction logs are updated every 15 minutes. In the event of disaster, the database is first restored and then all available transaction logs are applied.

15 Oracle

This section reviews covers the specific considerations for Oracle databases.

15.1 Consistency Group Backup

Placing an entire Oracle database in a single volume, including datafiles, archive logs, redo logs, and controlfiles, is a valid backup, restore, and replication method. However, the RPO is limited to the point of the backup itself. It is suitable for an RPO of one hour or greater. If a database spans volumes, cg-snapshots are required using one of the tools discussed previously.

As an example, the entire database can be in a single volume with the following snapshot schedule:

- 72 hourly snapshots

- 30 nightly snapshots
- 12 monthly snapshots

This delivers an RPO of one hour over the rolling period of the preceding 72 hours, plus additional nightly and monthly backups. Multiple databases or application files can also be included in the single volume or set of cg-snapshots to deliver consistent backups across a larger environment.

15.2 Oracle Hot Backup

Two sets of data are required to recover from a hot backup:

- A snapshot of the datafiles in backup mode
- The archive logs created while the datafiles were in hot backup mode

If complete recovery including all committed transactions is required, a third item is required:

- A set of current redo logs

There are a number of ways to drive recovery of a hot backup. Many customers restore snapshots by using the ONTAP CLI and then using Oracle RMAN or sqlplus to complete the recovery. This is especially common with large production environments in which the probability and frequency of database restores is extremely low and any restore procedure is handled by a skilled DBA. For complete automation, solutions such as NetApp SnapCenter include an Oracle plug-in with both command-line and graphical interfaces.

Some large-scale customers have taken a simpler approach by configuring basic scripting on the hosts to place the databases in hot backup mode at a specific time in preparation for a scheduled snapshot. For example, schedule the command `alter database begin backup` at 23:58, `alter database end backup` at 00:02, and then schedule snapshots directly on the storage system at midnight. The result is a simple, highly scalable backup strategy that requires no external software or licenses.

Data Layout

The simplest layout is to isolate datafiles into one or more dedicated volumes. They must be uncontaminated by any other file type. This is to make sure that the datafile volumes can be rapidly restored through a SnapRestore operation without destroying an important redo log, controlfile, or archive log.

SAN has similar requirements for datafile isolation within dedicated volumes. With an operating system such as Microsoft Windows, a single volume might contain multiple datafile LUNs, each with an NTFS file system. With other operating systems, there is generally a logical volume manager. For example, with Oracle ASM, the simplest option would be to confine the LUNs of an ASM disk group to a single volume that can be backed up and restored as a unit. If additional volumes are required for performance or capacity management reasons, creating an additional disk group on the new volume results in simpler management.

If these guidelines are followed, snapshots can be scheduled directly on the storage system with no requirement for performing a consistency group snapshot. The reason is that Oracle hot backups do not require datafiles to be backed up at the same time. The hot backup procedure was designed to allow datafiles to continue to be updated as they are slowly streamed to tape over the course of hours.

A complication arises in situations such as the use of an ASM disk group that is distributed across volumes. In these cases, a cg-snapshot must be performed to make sure that the ASM metadata is consistent across all constituent volumes.

Caution: Verify that the ASM `spfile` and `passwd` files are not in the disk group hosting the datafiles. This interferes with the ability to selectively restore datafiles and only datafiles.

Local Recovery Procedure—NFS

This procedure can be driven manually or through an application such as SnapCenter. The basic procedure is as follows:

1. Shut down the database.
2. Recover the datafile volume(s) to the snapshot immediately prior to the desired restore point.
3. Replay archive logs to the desired point.
4. Replay current redo logs if complete recovery is desired.

This procedure assumes that the desired archive logs are still present in the active file system. If they are not, the archive logs must be restored or rman/sqlplus can be directed to the data in the snapshot directory.

In addition, for smaller databases, datafiles can be recovered by an end user directly from the `.snapshot` directory without assistance from automation tools or storage administrators to execute a `snaprestore` command.

Local Recovery Procedure—SAN

This procedure can be driven manually or through an application such as SnapCenter. The basic procedure is as follows:

1. Shut down the database.
2. Quiesce the disk group(s) hosting the datafiles. The procedure varies depending on the logical volume manager chosen. With ASM, the process requires dismounting the disk group. With Linux, the file systems must be dismounted, and the logical volumes and volume groups must be deactivated. The objective is to stop all updates on the target volume group to be restored.
3. Restore the datafile disk groups to the snapshot immediately prior to the desired restore point.
4. Reactivate the newly restored disk groups.
5. Replay archive logs to the desired point.
6. Replay all redo logs if complete recovery is desired.

This procedure assumes that the desired archive logs are still present in the active file system. If they are not, the archive logs must be restored by taking the archive log LUNs offline and performing a restore. This is also an example in which dividing up archive logs into dedicated volumes is useful. If the archive logs share a volume group with redo logs, then the redo logs must be copied elsewhere before restoration of the overall set of LUNs. This step prevents the loss of those final recorded transactions.

15.3 Oracle Snapshot-Optimized Backup

Snapshot-based backup and recovery becomes even simpler with Oracle 12c because there is no need to place a database in hot backup mode. The result is an ability to schedule snapshot-based backups directly on a storage system and still preserve the ability to perform complete or point-in-time recovery.

Although the hot backup recovery procedure is more familiar to DBAs, it has, for a long time, been possible to use snapshots that were not created while the database was in hot backup mode. Extra manual steps were required with Oracle 10g and 11g during recovery to make the database consistent. With Oracle 12c, `sqlplus` and `rman` contain the extra logic to replay archive logs on datafile backups that were not in hot backup mode.

As discussed previously, recovering a snapshot-based hot backup requires two sets of data:

- A snapshot of the datafiles created while in backup mode
- The archive logs generated while the datafiles were in hot backup mode

During recovery, the database reads metadata from the datafiles to select the required archive logs for recovery.

Snapshot-optimized recovery requires slightly different datasets to accomplish the same results:

- A snapshot of the datafiles, plus a method to identify the time the snapshot was created
- Archive logs from the time of the most recent datafile checkpoint through the exact time of the snapshot

During recovery, the database reads metadata from the datafiles to identify the earliest archive log required. Full or point-in-time recovery can be performed. When performing a point-in-time recovery, it is critical to know the time of the snapshot of the datafiles. The specified recovery point must be after the creation time of the snapshots. NetApp recommends adding at least a few minutes to the snapshot time to account for clock variation.

For complete details, see Oracle's documentation on the topic, "Recovery Using Storage Snapshot Optimization" available in various releases of the Oracle 12c documentation. Also, see Oracle Document ID Doc ID 604683.1 regarding Oracle third-party snapshot support.

Data Layout

The simplest layout is to isolate the datafiles into one or more dedicated volumes. They must be uncontaminated by any other file type. This is to make sure that the datafile volumes can be rapidly restored with a SnapRestore operation without destroying an important redo log, controlfile, or archive log.

SAN has similar requirements for datafile isolation within dedicated volumes. With an operating system such as Microsoft Windows, a single volume might contain multiple datafile LUNs, each with an NTFS file system. With other operating systems, there is generally a logical volume manager as well. For example, with Oracle ASM, the simplest option would be to confine disk groups to a single volume that can be backed up and restored as a unit. If additional volumes are required for performance or capacity management reasons, creating an additional disk group on the new volume results in easier management.

If these guidelines are followed, snapshots can be scheduled directly on ONTAP with no requirement for performing a consistency group snapshot. The reason is that snapshot-optimized backups do not require that datafiles be backed up at the same time.

A complication arises in situations such as an ASM disk group that is distributed across volumes. In these cases, a cg-snapshot must be performed to make sure that the ASM metadata is consistent across all constituent volumes.

Caution: Verify that the ASM spfile and passwd files are not in the disk group hosting the datafiles. This interferes with the ability to selectively restore datafiles and only datafiles.

Local Recovery Procedure—NFS

This procedure can be driven manually or through an application such as SnapCenter. The basic procedure is as follows:

1. Shut down the database.
2. Recover the datafile volume(s) to the snapshot immediately prior to the desired restore point.
3. Replay archive logs to the desired point.

This procedure assumes that the desired archive logs are still present in the active file system. If they are not, the archive logs must be restored, or `rman` or `sqlplus` can be directed to the data in the `.snapshot` directory.

In addition, for smaller databases, datafiles can be recovered by an end user directly from the `.snapshot` directory without assistance from automation tools or a storage administrator to execute a SnapRestore command.

Local Recovery Procedure—SAN

This procedure can be driven manually or through an application such as SnapCenter. The basic procedure is as follows:

1. Shut down the database.
2. Quiesce the disk group(s) hosting the datafiles. The procedure varies depending on the logical volume manager chosen. With ASM, the process requires dismounting the disk group. With Linux, the file systems must be dismounted, and the logical volumes and volume groups are deactivated. The objective is to stop all updates on the target volume group to be restored.
3. Restore the datafile disk groups to the snapshot immediately prior to the desired restore point.
4. Reactivate the newly restored disk groups.
5. Replay archive logs to the desired point.

This procedure assumes that the desired archive logs are still present in the active file system. If they are not, the archive logs must be restored by taking the archive log LUNs offline and performing a restore. This is also an example in which dividing up archive logs into dedicated volumes is useful. If the archive logs share a volume group with redo logs, the redo logs must be copied elsewhere before restoration of the overall set of LUNs to avoid losing the final recorded transactions.

Full Recovery Example

Assume the datafiles have been corrupted or destroyed and full recovery is required. The procedure to do so is as follows:

```
[oracle@jfs2 ~]$ sqlplus / as sysdba

Connected to an idle instance.

SQL> startup mount;

ORACLE instance started.

Total System Global Area 1610612736 bytes
Fixed Size                  2924928 bytes
Variable Size            1040191104 bytes
Database Buffers          553648128 bytes
Redo Buffers               13848576 bytes
Database mounted.

SQL> recover automatic;
Media recovery complete.

SQL> alter database open;

Database altered.

SQL>
```

Point-in-time Recovery Example

The entire recovery procedure is a single command: `recover automatic`.

If point-in-time recovery is required, the timestamp of the snapshot(s) must be known and can be identified as follows:

```
EcoSystems-8060::> snapshot show -vserver svm0 -volume NTAP_oradata -fields create-time

vserver   volume        snapshot   create-time
--------  ------------  ---------  ------------------------
svm0      NTAP_oradata  my-backup  Thu Mar 09 10:10:06 2017
```

The snapshot creation time is listed as March 9th and 10:10:06. To be safe, one minute is added to the snapshot time:

```
[oracle@jfs2 ~]$ sqlplus / as sysdba

Connected to an idle instance.

SQL> startup mount;

ORACLE instance started.

Total System Global Area 1610612736 bytes
Fixed Size                  2924928 bytes
Variable Size            1040191104 bytes
Database Buffers          553648128 bytes
Redo Buffers               13848576 bytes
Database mounted.

SQL> recover database until time '09-MAR-2017 10:44:15' snapshot time '09-MAR-2017 10:11:00';
```

The recovery is now initiated. It specified a snapshot time of 10:11:00, one minute after the recorded time to account for possible clock variance, and a target recovery time of 10:44. Next, sqlplus requests the archive logs required to reach the desired recovery time of 10:44.

```
ORA-00279: change 551760 generated at 03/09/2017 05:06:07 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_31_930813377.dbf
ORA-00280: change 551760 for thread 1 is in sequence #31


Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

ORA-00279: change 552566 generated at 03/09/2017 05:08:09 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_32_930813377.dbf
ORA-00280: change 552566 for thread 1 is in sequence #32


Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

ORA-00279: change 553045 generated at 03/09/2017 05:10:12 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_33_930813377.dbf
ORA-00280: change 553045 for thread 1 is in sequence #33


Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

ORA-00279: change 753229 generated at 03/09/2017 05:15:58 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_34_930813377.dbf
ORA-00280: change 753229 for thread 1 is in sequence #34


Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

Log applied.
Media recovery complete.
```

```
SQL> alter database open resetlogs;

Database altered.

SQL>
```

15.4 Consistency Group Disaster Recovery

Consistency group replication can be can be as simple as scheduling replication of a single volume SnapMirror. This includes datafiles, controlfiles, archive logs, and redo logs. Every SnapMirror update results in a new copy of the database at the destination site that is consistent and ready for activation by breaking the mirror.

Where a database must span volumes, a consistency group snapshot (cg-snapshot) is required. See the section “Log Replication” for additional information on managing cg-snapshots.

An additional benefit of this strategy when used with SnapMirror in block-level replication mode is complete replication of all snapshots on the source storage system. The full complement of backups is replicated in addition to the disaster recovery copy.

15.5 Disaster Recovery with Log Replay

The replication procedures for an Oracle database are essentially the same as the backup procedures. The primary requirement is that the snapshots that constitute a recoverable backup must be replicated to the remote storage system.

As discussed previously in the section on local data protection, a recoverable backup can be created with the hot backup process or by leveraging snapshot-optimized backups.

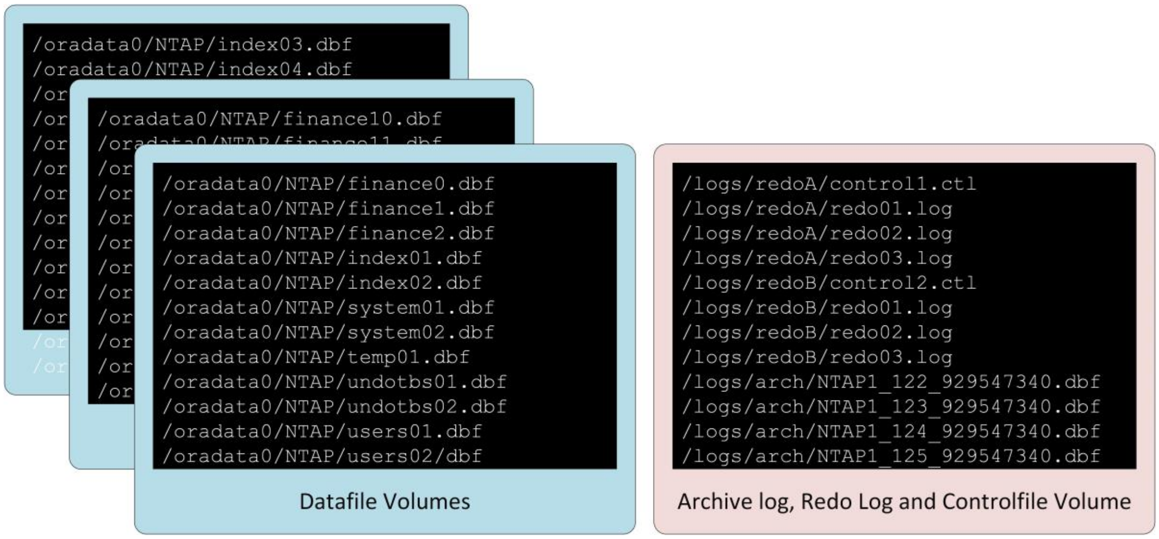
The most important requirement is isolation of the datafiles into one or more dedicated volumes. They must be uncontaminated by any other file type. The reason is to make sure that datafile replication is wholly independent of replication of other data types such as archive logs. For additional information on file layouts and for important details on ensuring the storage layout is snapshot friendly, see section “Data Layout.”

Assuming the datafiles are encapsulated into dedicated volumes, the next question is how to manage the redo logs, archive logs, and controlfiles. The simplest approach is to place all those data types into a single volume. The benefit of this is that replicated redo logs, archive logs, and controlfiles are perfectly synchronized. There is no requirement for incomplete recovery or using a backup controlfile, although it might be desirable to also script creation of backup controlfiles for other potential recovery scenarios.

Two-Volume Layout

The simplest layout is shown in Figure 3.

Figure 3) Two-volume layout.

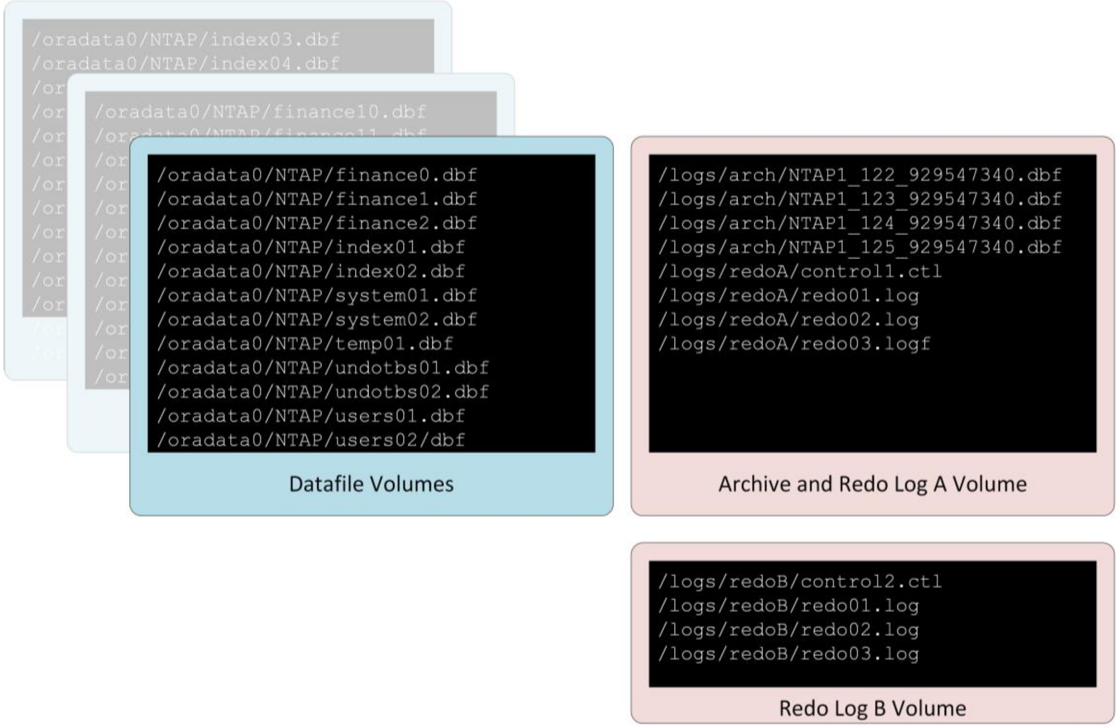

/oradata0/NTAP/index03.dbf
/oradata0/NTAP/index04.dbf
/oradata0/NTAP/finance10.dbf
/oradata0/NTAP/finance0.dbf
/oradata0/NTAP/finance1.dbf
/oradata0/NTAP/finance2.dbf
/oradata0/NTAP/index01.dbf
/oradata0/NTAP/index02.dbf
/oradata0/NTAP/system01.dbf
/oradata0/NTAP/system02.dbf
/oradata0/NTAP/temp01.dbf
/oradata0/NTAP/undotbs01.dbf
/oradata0/NTAP/undotbs02.dbf
/oradata0/NTAP/users01.dbf
/oradata0/NTAP/users02/dbf
Datafile Volumes
/logs/redoA/control1.ctl
/logs/redoA/redo01.log
/logs/redoA/redo02.log
/logs/redoA/redo03.log
/logs/redoB/control2.ctl
/logs/redoB/redo01.log
/logs/redoB/redo02.log
/logs/redoB/redo03.log
/logs/arch/NTAP1_122_929547340.dbf
/logs/arch/NTAP1_123_929547340.dbf
/logs/arch/NTAP1_124_929547340.dbf
/logs/arch/NTAP1_125_929547340.dbf
Archive log, Redo Log and Controlfile Volume

This is the most common approach. From a DBA perspective, it might seem unusual to colocate all copies of the redo logs and archive logs on the same volume. However, separation does not offer much extra protection when the files and LUNs are all still located on the same underlying set of drives.

Three-Volume Layout

Sometimes separation of redo logs is required because of data protection concerns or a need to distribute redo log I/O across controllers. If so, then the three-volume layout depicted in Figure 4 can be used for replication while still avoiding any requirement to perform incomplete recovery or rely on backup controlfiles.

Figure 4) Three-volume layout.


/oradata0/NTAP/index03.dbf
/oradata0/NTAP/index04.dbf
/oradata0/NTAP/finance10.dbf
/oradata0/NTAP/finance0.dbf
/oradata0/NTAP/finance1.dbf
/oradata0/NTAP/finance2.dbf
/oradata0/NTAP/index01.dbf
/oradata0/NTAP/index02.dbf
/oradata0/NTAP/system01.dbf
/oradata0/NTAP/system02.dbf
/oradata0/NTAP/temp01.dbf
/oradata0/NTAP/undotbs01.dbf
/oradata0/NTAP/undotbs02.dbf
/oradata0/NTAP/users01.dbf
/oradata0/NTAP/users02/dbf
Datafile Volumes
/logs/arch/NTAP1_122_929547340.dbf
/logs/arch/NTAP1_123_929547340.dbf
/logs/arch/NTAP1_124_929547340.dbf
/logs/arch/NTAP1_125_929547340.dbf
/logs/redoA/control1.ctl
/logs/redoA/redo01.log
/logs/redoA/redo02.log
/logs/redoA/redo03.logf
Archive and Redo Log A Volume
/logs/redoB/control2.ctl
/logs/redoB/redo01.log
/logs/redoB/redo02.log
/logs/redoB/redo03.log
Redo Log B Volume

This permits striping of the redo logs and controlfiles across independent sets of spindles and controllers on the source. However, the archive logs and one set of controlfiles and redo logs can still be replicated in a synchronized state with the archive logs.

In this model, the Redo Log B volume is not replicated.

Disaster Recovery Procedure—Hot Backups

To perform disaster recovery by using hot backups, use the following basic procedure:

Prerequisites

1. Oracle binaries are installed on the disaster recovery server.
2. Database instances are listed in `/etc/oratab`.
3. The `passwd` and `pfile` or `spfile` for the instance must be in the `$ORACLE_HOME/dbs` directory.

Disaster Recovery

1. Break the mirrors for the datafiles and common log volume.
2. Restore the datafile volume(s) to the most recent hot backup snapshot of the datafiles.
3. If SAN is used, activate volume groups and/or mount file systems.
4. Replay archive logs to the desired point.
5. Replay current redo logs if complete recovery is desired.

Using NFS simplifies the procedure dramatically because the NFS file systems for the datafiles and log files can be mounted on the disaster recovery server at any time. It becomes read/write when the mirrors are broken.

Disaster Recovery Procedure—Snapshot-Optimized Backups

Recovering from snapshot-optimized backups is almost identical to the hot backup recovery procedure with the following changes:

1. Break the mirrors for the datafiles and common log volume.
2. Restore the datafile volume(s) to a snapshot created prior to the current log volume replica.
3. If SAN is used, activate volume groups and/or mount file systems.
4. Replay archive logs to the desired point.
5. Replay current redo logs if complete recovery is desired.

These differences simplify the overall recovery procedure because there is no requirement for making sure that a snapshot was properly created on the source while the database was in hot backup mode. The disaster recovery procedure is based on the timestamps of the snapshots on the disaster recovery site. The state of the database when the snapshots were created is not important.

Disaster Recovery with Hot Backup Snapshots

This is an example of a disaster recovery strategy based on the replication of hot backup snapshots. It also serves as an example of a simple and scalable local backup strategy.

The example database is located on a basic two-volume architecture. `/oradata` contains datafiles and `/oralogs` is used for combined redo logs, archive logs, and controlfiles.

```
[root@jfs2 ~]# ls /ora*

/oradata:
dbf
```

```
/oralogs:
arch  ctrl  redo
```

Two schedules are required, one for the nightly datafile backups and one for the log file backups. These are called midnight and 15minutes, respectively.

```
EcoSystems-8060::> job schedule cron show -name midnight|15minutes
Name                Description
----------------    -----------------------------------------------------
15minutes           @:00,:15,:30,:45
midnight            @0:00
2 entries were displayed.
```

These schedules are then used inside the snapshot policies `NTAP-datafile-backups` and `NTAP-log-backups`, as shown below:

```
EcoSystems-8060::> snapshot policy show -vserver jfsCloud0 -policy NTAP-* -fields
schedules,counts
vserver   policy                schedules                    counts
--------- --------------------- ---------------------------- ------
jfsCloud0 NTAP-datafile-backups midnight                     60
jfsCloud0 NTAP-log-backups      15minutes                    72
2 entries were displayed.
```

Finally, these snapshot policies are applied to the volumes.

```
EcoSystems-8060::> volume show -vserver jfsCloud0 -volume jfs2_oracle* -fields snapshot-policy
vserver   volume                 snapshot-policy
--------- ---------------------- ---------------------
jfsCloud0 jfs2_oracle_datafiles  NTAP-datafile-backups
jfsCloud0 jfs2_oracle_logs       NTAP-log-backups
```

This defines the backup schedule of the volumes. Datafile snapshots are created at midnight and retained for 60 days. The log volume contains 72 snapshots created at 15-minute intervals, which adds up to 18 hours of coverage.

Then, make sure that the database is in hot backup mode when a datafile snapshot is created. This is done with a small script that accepts some basic arguments that start and stop backup mode on the specified SID.

```
58 * * * * /snapomatic/current/smatic.db.ctrl --sid NTAP --startbackup
02 * * * * /snapomatic/current/smatic.db.ctrl --sid NTAP --stopbackup
```

This step makes sure that the database is in hot backup mode during a four-minute window surrounding the midnight snapshot.

The replication to the disaster recovery site is configured as follows:

```
EcoSystems-8060::> snapmirror show -destination-path jfsCloud1:jfsdr2* -fields source-
path,destination-path,schedule
source-path                      destination-path                   schedule
-------------------------------- ---------------------------------- --------
jfsCloud0:jfs2_oracle_datafiles  jfsCloud1:jfsdr2_oracle_datafiles  6hours
jfsCloud0:jfs2_oracle_logs       jfsCloud1:jfsdr2_oracle_logs       15minutes
2 entries were displayed.
```

The log volume destination is updated every 15 minutes. This delivers an RPO of approximately 15 minutes. The precise update interval varies a little depending on the total volume of data that must be transferred during the update.

The datafile volume destination is updated every six hours. This does not affect the RPO or RTO. If disaster recovery is required, one of the first steps is to restore the datafile volume back to a hot backup snapshot. The purpose of the more frequent update interval is to smooth the transfer rate of this volume. If the update is scheduled for once per day, all changes that accumulated during the day must be transferred at once. With more frequent updates, the changes are replicated more gradually across the day.

If a disaster occurs, the first step is to break the mirrors for both volumes:

```
EcoSystems-8060::> snapmirror break -destination-path jfsCloud1:jfsdr2_oracle_datafiles -force
Operation succeeded: snapmirror break for destination "jfsCloud1:jfsdr2_oracle_datafiles".

EcoSystems-8060::> snapmirror break -destination-path jfsCloud1:jfsdr2_oracle_logs -force
Operation succeeded: snapmirror break for destination "jfsCloud1:jfsdr2_oracle_logs".

EcoSystems-8060::>
```

The replicas are now read-write. The next step is to verify the timestamp of the log volume.

```
EcoSystems-8060::> snapmirror show -destination-path jfsCloud1:jfsdr2_oracle_logs -field newest-
snapshot-timestamp
source-path                destination-path             newest-snapshot-timestamp
-------------------------- ---------------------------- -------------------------
jfsCloud0:jfs2_oracle_logs jfsCloud1:jfsdr2_oracle_logs 03/14 13:30:00
```

The most recent copy of the log volume is March 14th at 13:30:00.

Next, identify the hot backup snapshot created immediately prior to the state of the log volume. This is required because the log replay process requires all archive logs created during hot backup mode. The log volume replica therefore must be older than the hot backup images or it would not contain the required logs.

```
EcoSystems-8060::> snapshot list -vserver jfsCloud1 -volume jfsdr2_oracle_datafiles -fields
create-time -snapshot midnight*

vserver   volume                    snapshot                   create-time
--------- ------------------------  -------------------------- ------------------------
jfsCloud1 jfsdr2_oracle_datafiles   midnight.2017-01-14_0000   Sat Jan 14 00:00:00 2017
jfsCloud1 jfsdr2_oracle_datafiles   midnight.2017-01-15_0000   Sun Jan 15 00:00:00 2017

...

jfsCloud1 jfsdr2_oracle_datafiles   midnight.2017-03-12_0000   Sun Mar 12 00:00:00 2017
jfsCloud1 jfsdr2_oracle_datafiles   midnight.2017-03-13_0000   Mon Mar 13 00:00:00 2017
jfsCloud1 jfsdr2_oracle_datafiles   midnight.2017-03-14_0000   Tue Mar 14 00:00:00 2017
60 entries were displayed.

EcoSystems-8060::>
```

The most recently created snapshot is `midnight.2017-03-14_0000`. This is the most recent hot backup image of the datafiles, and it is then restored as follows:

```
EcoSystems-8060::> snapshot restore -vserver jfsCloud1 -volume jfsdr2_oracle_datafiles -snapshot
midnight.2017-03-14_0000

EcoSystems-8060::>
```

At this stage, the database is now ready to be recovered. If this was a SAN environment, the next step would include activating volume groups and mounting file systems, an easily automated process. Because this example uses NFS, the file systems are already mounted and became read-write with no further need for mounting or activation the moment the mirrors were broken.

The database can now be recovered to the desired point in time, or it can be fully recovered with respect to the copy of the redo logs that was replicated. This example illustrates the value of the combined archive log, controlfile, and redo log volume. The recovery process is dramatically simpler because there is no requirement to rely on backup controlfiles or reset log files.

```
[oracle@jfsdr2 ~]$ sqlplus / as sysdba

Connected to an idle instance.

SQL> startup mount;
ORACLE instance started.

Total System Global Area 1610612736 bytes
Fixed Size                  2924928 bytes
Variable Size            1090522752 bytes
Database Buffers          503316480 bytes
Redo Buffers               13848576 bytes
Database mounted.

SQL> recover database until cancel;

ORA-00279: change 1291884 generated at 03/14/2017 12:58:01 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_34_938169986.dbf
ORA-00280: change 1291884 for thread 1 is in sequence #34


Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
auto
ORA-00279: change 1296077 generated at 03/14/2017 15:00:44 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_35_938169986.dbf
ORA-00280: change 1296077 for thread 1 is in sequence #35
ORA-00278: log file '/oralogs_nfs/arch/1_34_938169986.dbf' no longer needed for
this recovery

...

ORA-00279: change 1301407 generated at 03/14/2017 15:01:04 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_40_938169986.dbf
ORA-00280: change 1301407 for thread 1 is in sequence #40
ORA-00278: log file '/oralogs_nfs/arch/1_39_938169986.dbf' no longer needed for
this recovery


ORA-00279: change 1301418 generated at 03/14/2017 15:01:19 needed for thread 1
ORA-00289: suggestion : /oralogs_nfs/arch/1_41_938169986.dbf
ORA-00280: change 1301418 for thread 1 is in sequence #41
ORA-00278: log file '/oralogs_nfs/arch/1_40_938169986.dbf' no longer needed for
this recovery


ORA-00308: cannot open archived log '/oralogs_nfs/arch/1_41_938169986.dbf'
ORA-17503: ksfdopn:4 Failed to open file /oralogs_nfs/arch/1_41_938169986.dbf
ORA-17500: ODM err:File does not exist


SQL> recover database;

Media recovery complete.

SQL> alter database open;

Database altered.

SQL>
```

15.6 Disaster Recovery with Snapshot-Optimized Backups

The disaster recovery procedure using snapshot-optimized backups is nearly identical to the hot backup disaster recovery procedure. As with the hot backup snapshot procedure, it is also essentially an extension of a local backup architecture in which the backups are replicated for use in disaster recovery. The following example shows the detailed configuration and recovery procedure. This example also calls out the key differences between hot backups and snapshot-optimized backups.

The example database is located on a basic two-volume architecture. `/oradata` contains datafiles, and `/oralogs` is used for combined redo logs, archive logs, and controlfiles.

```
 [root@jfs3 ~]# ls /ora*
/oradata:
dbf

/oralogs:
arch  ctrl  redo
```

Two schedules are required: one for the nightly datafile backups and one for the log file backups. These are called midnight and 15minutes, respectively.

```
EcoSystems-8060::> job schedule cron show -name midnight|15minutes
Name                Description
----------------    -----------------------------------------------------
15minutes           @:00,:15,:30,:45
midnight            @0:00
2 entries were displayed.
```

These schedules are then used inside the snapshot policies `NTAP-datafile-backups` and `NTAP-log-backups`, as shown below:

```
EcoSystems-8060::> snapshot policy show -vserver jfsCloud0 -policy NTAP-* -fields
schedules,counts
vserver   policy                schedules                    counts
--------- --------------------- ---------------------------- ------
jfsCloud0 NTAP-datafile-backups midnight                     60
jfsCloud0 NTAP-log-backups      15minutes                    72
2 entries were displayed.
```

Finally, these snapshot policies are applied to the volumes.

```
EcoSystems-8060::> volume show -vserver jfsCloud0 -volume jfs3_oracle* -fields snapshot-policy
vserver   volume                 snapshot-policy
--------- ---------------------- ---------------------
jfsCloud0 jfs2_oracle_datafiles  NTAP-datafile-backups
jfsCloud0 jfs2_oracle_logs       NTAP-log-backups
```

This controls the ultimate backup schedule of the volumes. Snapshots are created at midnight and retained for 60 days. The log volume contains 72 snapshots created at 15-minute intervals which adds up to 18 hours of coverage.

The replication to the disaster recovery site is configured as follows:

```
EcoSystems-8060::> snapmirror show -destination-path jfsCloud1:jfsdr3* -fields source-
path,destination-path,schedule
source-path                      destination-path                   schedule
-------------------------------- ---------------------------------- --------
jfsCloud0:jfs3_oracle_datafiles  jfsCloud1:jfsdr3_oracle_datafiles  6hours
jfsCloud0:jfs3_oracle_logs       jfsCloud1:jfsdr3_oracle_logs       15minutes
2 entries were displayed.
```

The log volume destination is updated every 15 minutes. This delivers an RPO of approximately 15 minutes, with the precise update interval varying a little depending on the total volume of data that must be transferred during the update.

The datafile volume destination is updated every 6 hours. This does not affect the RPO or RTO. If disaster recovery is required, you must first restore the datafile volume back to a hot backup snapshot. The purpose of the more frequent update interval is to smooth the transfer rate of this volume. If the update was scheduled once per day, all changes that accumulated during the day must be transferred at once. With more frequent updates, the changes are replicated more gradually across the day.

If a disaster occurs, first step is to break the mirrors for all the volumes:

```
EcoSystems-8060::> snapmirror break -destination-path jfsCloud1:jfsdr3_oracle_datafiles -force
Operation succeeded: snapmirror break for destination "jfsCloud1:jfsdr3_oracle_datafiles".

EcoSystems-8060::> snapmirror break -destination-path jfsCloud1:jfsdr3_oracle_logs -force
Operation succeeded: snapmirror break for destination "jfsCloud1:jfsdr3_oracle_logs".

EcoSystems-8060::>
```

The replicas are now read-write. The next step is to verify the timestamp of the log volume.

```
EcoSystems-8060::> snapmirror show -destination-path jfsCloud1:jfsdr3_oracle_logs -field newest-
snapshot-timestamp
source-path                destination-path             newest-snapshot-timestamp
-------------------------- ---------------------------- -------------------------
jfsCloud0:jfs3_oracle_logs jfsCloud1:jfsdr3_oracle_logs 03/14 13:30:00
```

The most recent copy of the log volume is March 14th at 13:30. Next, identify the datafile snapshot created immediately prior to the state of the log volume. This is required because the log replay process requires all archive logs from just prior to the snapshot to the desired recovery point.

```
EcoSystems-8060::> snapshot list -vserver jfsCloud1 -volume jfsdr3_oracle_datafiles -fields
create-time -snapshot midnight*

vserver   volume                    snapshot                   create-time
--------- ------------------------  -------------------------- ------------------------
jfsCloud1 jfsdr3_oracle_datafiles   midnight.2017-01-14_0000   Sat Jan 14 00:00:00 2017
jfsCloud1 jfsdr3_oracle_datafiles   midnight.2017-01-15_0000   Sun Jan 15 00:00:00 2017

...

jfsCloud1 jfsdr3_oracle_datafiles   midnight.2017-03-12_0000   Sun Mar 12 00:00:00 2017
jfsCloud1 jfsdr3_oracle_datafiles   midnight.2017-03-13_0000   Mon Mar 13 00:00:00 2017
jfsCloud1 jfsdr3_oracle_datafiles   midnight.2017-03-14_0000   Tue Mar 14 00:00:00 2017
60 entries were displayed.

EcoSystems-8060::>
```

The most recently created snapshot is `midnight.2017-03-14_0000`. Restore this snapshot.

```
EcoSystems-8060::> snapshot restore -vserver jfsCloud1 -volume jfsdr3_oracle_datafiles -snapshot
midnight.2017-03-14_0000

EcoSystems-8060::>
```

The database is now ready to be recovered. If this was a SAN environment, you would then activate volume groups and mount file systems, an easily automated process. However, this example is using NFS, so the file systems are already mounted and became read-write with no further need for mounting or activation the moment the mirrors were broken.

The database can now be recovered to the desired point in time, or it can be fully recovered with respect to the copy of the redo logs that was replicated. This example illustrates the value of the combined archive log, controlfile, and redo log volume. The recover process is dramatically simpler because there is no requirement to rely on backup controlfiles or reset log files.

```
[oracle@jfsdr3 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Wed Mar 15 12:26:51 2017

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup mount;
ORACLE instance started.

Total System Global Area 1610612736 bytes
```

```
Fixed Size                  2924928 bytes
Variable Size            1073745536 bytes
Database Buffers          520093696 bytes
Redo Buffers               13848576 bytes
Database mounted.
SQL> recover automatic;
Media recovery complete.
SQL> alter database open;

Database altered.

SQL>
```

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer’s installation in accordance with published specifications.

Copyright Information

Trademark Information