Oracle database migration

Enterprise applications

NetApp
April 25, 2024

Table of Contents

Oracle database migration

Migration of Oracle databases to ONTAP storage systems

Leveraging the capabilities of a new storage platform has one unavoidable requirement; data must be placed on the new storage system. ONTAP makes the migration process simple, including both ONTAP to ONTAP migrations and upgrades, foreign LUN imports, and procedures for using the host operating system or Oracle database software directly.

This documentation replaces previously published technical report *TR-4534: Migration of Oracle Databases to NetApp Storage Systems*

In the case of a new database project, this is not a concern because the database and application environments are constructed in place. Migration, however, poses special challenges regarding business disruption, the time required for the completion of migration, needed skill sets, and risk minimization.

Scripts

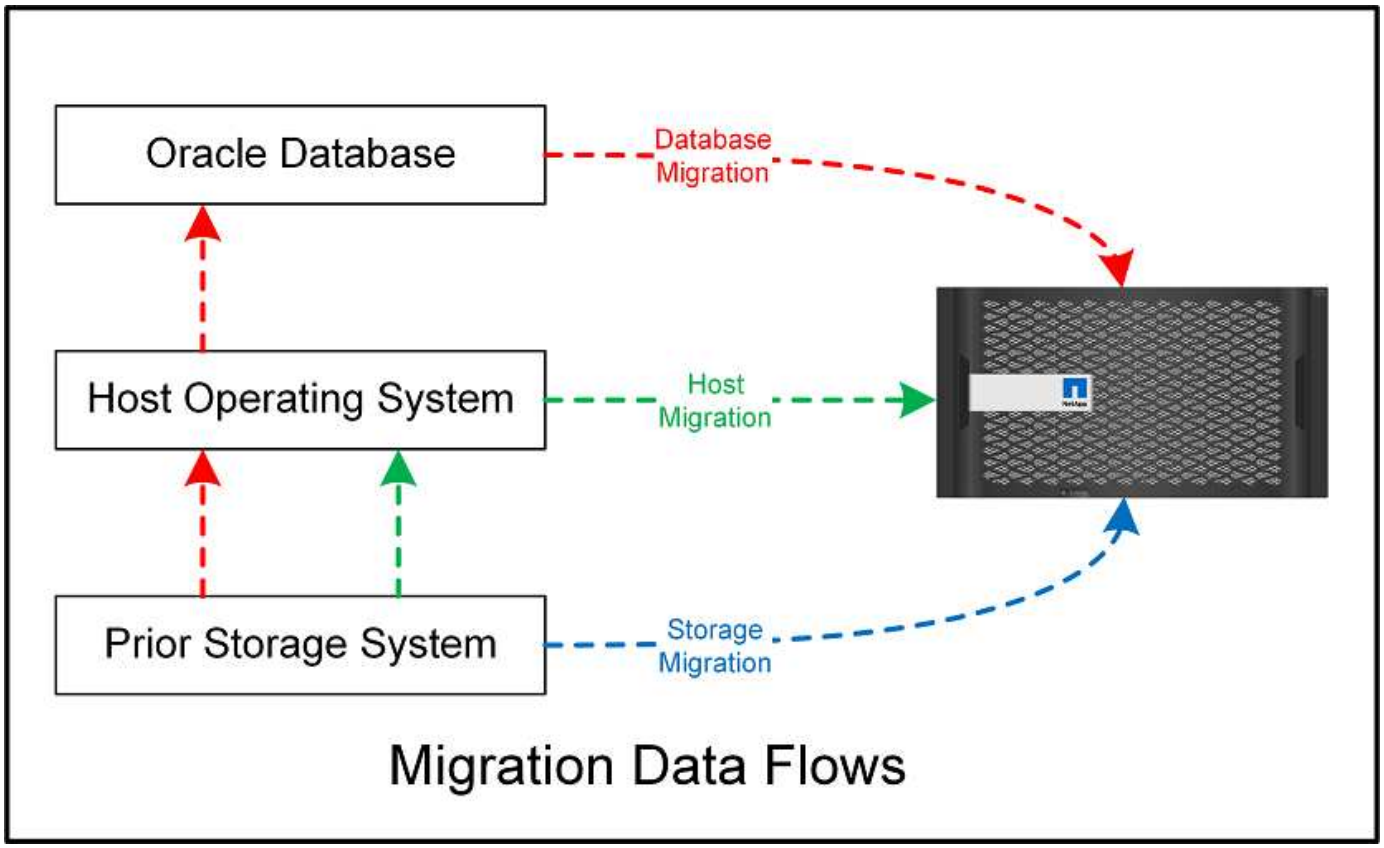
Sample scripts are provided in this documentation. These scripts provide sample methods of automating various aspects of migration to reduce the chance of user errors. The scripts can reduce the overall demands on the IT staff responsible for a migration and they can speed up the overall process. These scripts are all drawn from actual migration projects performed by NetApp Professional Services and NetApp partners. Examples of their use are shown throughout this documentation.

Oracle database migration planning

Oracle data migration can occur at one of three levels: the database, the host, or the storage array.

The differences lie in which component of the overall solution is responsible for moving data: the database, the host operating system, or the storage system.

The figure below shows an example of the migration levels and the flow of data. In the case of database-level migration, the data is moved from the original storage system through the host and database layers into the new environment. Host-level migration is similar, but data does not pass through the application layer and is instead written to the new location by using host processes. Finally, with storage-level migration, an array such as a NetApp FAS system is responsible for data movement.



A database-level migration generally refers to the use of Oracle log shipping through a standby database to complete a migration at the Oracle layer. Host-level migrations are performed by using the native capability of the host operating system configuration. This configuration includes file copy operations using commands such as cp, tar, and Oracle Recovery Manager (RMAN) or using a logical volume manager (LVM) to relocate the underlying bytes of a file system. Oracle Automatic Storage Management (ASM) is categorized as a host- level capability because it runs below the level of the database application. ASM takes the place of the usual logical volume manager on a host. Finally, data can be migrated at the storage- array level, which means beneath the level of the operating system.

Planning considerations

The best option for migration depends on a combination of factors, including the scale of the environment to be migrated, the need to avoid downtime, and the overall effort required to perform the migration. Large databases obviously require more time and effort for migration, but the complexity of such a migration is minimal. Small databases can be migrated quickly, but, if there are thousands to be migrated, the scale of the effort can create complications. Finally, the larger the database, the more likely it is to be business-critical, which gives rise to a need to minimize downtime while preserving a back- out path.

Some of the considerations for planning a migration strategy are discussed here.

Data size

The sizes of the databases to be migrated obviously affect migration planning, although size does not necessarily affect the cutover time. When a large amount of data must be migrated, the primary consideration is bandwidth. Copy operations are usually performed with efficient sequential I/O. As a conservative estimate, assume 50% utilization of the available network bandwidth for copy operations. For example, an 8GB FC port can transfer about 800MBps in theory. Assuming 50% utilization, a database can be copied at a rate of about 400MBps. Therefore, a 10TB database can be copied in about seven hours at this rate.

Migration over longer distances usually requires a more creative approach, such as the log shipping process explained in Online datafile move. Long-distance IP networks rarely have bandwidth anywhere close to LAN or SAN speeds. In one case, NetApp assisted with the long-distance migration of a 220TB database with very high archive- log generation rates. The chosen approach for data transfer was daily shipment of tapes, because this method offered the maximum possible bandwidth.

Database count

In many cases, the problem with moving a large amount of data is not the data size, but rather it is the complexity of the configuration that supports the database. Simply knowing that 50TB of databases must be migrated is not sufficient information. It could be a single 50TB mission-critical database, a collection of 4, 000 legacy databases, or a mix of production and nonproduction data. In some cases, much of the data consists of clones of a source database. These clones do not need to be migrated at all because they can be easily recreated, especially when the new architecture is designed to leverage NetApp FlexClone volumes.

For migration planning, you must understand how many databases are in scope and how they must be prioritized. As the number of databases increases, the preferred migration option tends to be lower and lower in the stack. For example, copying a single database might be easily performed with RMAN and a short outage. This is host-level replication.

If there are 50 databases, it might be easier to avoid setting up a new file system structure to receive an RMAN copy and instead move the data in place. This process can be done by leveraging host-based LVM migration to relocate data from old LUNs to new LUNs. Doing so moves responsibility from the database administrator (DBA) team to the OS team, and, as a result, data is migrated transparently with respect to the database. The file system configuration is unchanged.

Finally, if 500 databases across 200 servers must be migrated, storage-based options such as the ONTAP Foreign LUN Import (FLI) capability can be used to perform a direct migration of the LUNs.

Rearchitecture requirements

Typically, a database file layout must be altered to leverage the features of the new storage array; however, this is not always the case. For example, the features of EF-Series all-flash arrays are directed primarily at SAN performance and SAN reliability. In most cases, databases can be migrated to an EF-Series array with no special considerations for data layout. The only requirements are high IOPS, low latency, and robust reliability. Although there are best practices relating to such factors as RAID configuration or Dynamic Disk Pools, EF-Series projects rarely require any significant changes to the overall storage architecture to leverage such features.

In contrast, migration to ONTAP generally requires more consideration of the database layout to make sure that the final configuration delivers maximum value. By itself, ONTAP offers many features for a database environment, even without any specific architecture effort. Most importantly, it delivers the ability to nondisruptively migrate to new hardware when the current hardware reaches its end of life. Generally speaking, a migration to ONTAP is the last migration that you would need to perform. Subsequent hardware is upgraded in place and data is nondisruptively migrated to new media.

With some planning, even more benefits are available. The most important considerations surround the use of snapshots. Snapshots are the basis for performing near-instantaneous backups, restores, and cloning operations. As an example of the power of snapshots, the largest known use is with a single database of 996TB running on about 250 LUNs on 6 controllers. This database can be backed up in 2 minutes, restored in 2 minutes, and cloned in 15 minutes. Additional benefits include the ability to move data around the cluster in response to changes in workload and the application of quality of service (QoS) controls to provide good, consistent performance in a multidatabase environment.

Technologies such as QoS controls, data relocation, snapshots, and cloning work in nearly any configuration.

However, some thought is generally required to maximize benefits. In some cases, database storage layouts can require design changes to maximize the investment in the new storage array. Such design changes can affect the migration strategy because host-based or storage-based migrations replicate the original data layout. Additional steps might be required to complete the migration and deliver a data layout optimized for ONTAP. The procedures shown in Oracle migration procedures overview and later demonstrate some of the methods to not just migrate a database, but to migrate it into the optimal final layout with minimal effort.

Cutover time

The maximum allowable service outage during cutover should be determined. It is a common mistake to assume that the entire migration process causes disruption. Many tasks can be completed before any service interruption begins, and many options enable the completion of migration without disruption or outage. Even when disruption is unavoidable, you must still define the maximum allowable service outage because the duration of the cutover time varies from procedure to procedure.

For example, copying a 10TB database typically requires approximately seven hours to complete. If business needs allow a seven- hour outage, file copying is an easy and safe option for migration. If five hours is unacceptable, a simple log- shipping process (see Oracle log shipping) can be set up with minimal effort to reduce the cutover time to approximately 15 minutes. During this time, a database administrator can complete the process. If 15 minutes is unacceptable, the final cutover process can be automated through scripting to reduce the cutover time to just a few minutes. You can always speed up a migration, but doing so comes at the cost of time and effort. The cutover time targets should be based on what is acceptable to the business.

Back-out path

No migration is completely risk free. Even if technology operates perfectly, there is always a possibility of user error. The risk associated with a chosen migration path must be considered alongside the consequences of a failed migration. For example, the transparent online storage migration capability of Oracle ASM is one of its key features, and this method is one of the most reliable known. However, data is being irreversibly copied with this method. In the highly unlikely event that a problem occurs with ASM, there is no easy back- out path. The only option is to either restore the original environment or use ASM to reverse the migration back to the original LUNs. The risk can be minimized, but not eliminated, by performing a snapshot-type backup on the original storage system, assuming the system is capable of performing such an operation.

Rehearsal

Some migration procedures must be fully verified before execution. A need for migration and rehearsal of the cutover process is a common request with mission-critical databases for which migration must be successful and downtime must be minimized. In addition, user- acceptance tests are frequently included as part of the postmigration work, and the overall system can be returned to production only after these tests are complete.

If there is a need for rehearsal, several ONTAP capabilities can make the process much easier. In particular, snapshots can reset a test environment and quickly create multiple space-efficient copies of a database environment.

Procedures

Oracle migration procedures overview

Many procedures are available for Oracle migration database. The right one depends on your business needs.

In many cases, system administrators and DBAs have their own preferred methods of relocating physical

volume data, mirroring and demirroring, or leveraging Oracle RMAN to copy data.

These procedures are provided primarily as guidance for IT staff less familiar with some of the available options. In addition, the procedures illustrate the tasks, time requirements, and skillset demands for each migration approach. This allows other parties such as NetApp and partner professional services or IT management to more fully appreciate the requirements for each procedure.

There is no single best practice for creating a migration strategy. Creating a plan requires first understanding the availability options and then selecting the method that best suits the needs of the business. The figure below illustrates the basic considerations and typical conclusions made by customers, but it is not universally applicable to all situations.

For example, one step raises the issue of the total database size. The next step depends on whether the database is more or less than 1TB. The recommended steps are just that—recommendations based on typical customer practices. Most customers would not use DataGuard to copy a small database, but some might. Most customers would not attempt to copy a 50TB database because of the time required, but some might have a sufficiently large maintenance window to permit such an operation.

You can find a flowchart of the types of considerations on which migration path is best here.

Online datafile move

Oracle 12cR1 and higher include the ability to move a datafile while the database remains online. It furthermore works between different filesystem types. For example, a datafile can be relocated from an xfs filesystem to ASM. This method is not generally used at scale because of the number of individual datafile move operations that would be required, but it is an option worth considering with smaller databases with fewer datafiles.

In addition, simply moving a datafile is a good option for migrating parts of existing databases. For example, less-active datafiles could be relocated to more cost-efficient storage, such as a FabricPool volume which can store idle blocks in Object Store.

Database-level migration

Migration at the database level means allowing the database to relocate data. Specifically, this means log shipping. Technologies such as RMAN and ASM are Oracle products, but, for the purposes of migration, they operate at the host level where they copy files and manage volumes.

Log shipping

The foundation for database- level migration is the Oracle archive log, which contains a log of changes to the database. Most of the time, an archive log is part of a backup and recovery strategy. The recovery process begins with the restoration of a database and then the replaying of one or more archive logs to bring the database to the desired state. This same basic technology can be used to perform a migration with little to no interruption of operations. More importantly, this technology enables migration while leaving the original database untouched, preserving a back- out path.

The migration process begins with restoration of a database backup to a secondary server. You can do so in a variety of ways, but most customers use their normal backup application to restore the data files. After the data files are restored, users establish a method for log shipping. The goal is to create a constant feed of archive logs generated by the primary database and replay them on the restored database to keep them both close to the same state. When the cutover time arrives, the source database is completely shut down and the final archive logs, and in some cases the redo logs, are copied over and replayed. It is critical that the redo logs are also considered because they might contain some of the final transactions committed.

After these logs have been transferred and replayed, both databases are consistent with one another. At this point, most customers perform some basic testing. If any errors are made during the migration process, then the log replay should report errors and fail. It is still advisable to perform some quick tests based on known queries or application-driven activities to verify that the configuration is optimal. It is also a common practice to create one final test table before shutting down the original database to verify whether it is present in the migrated database. This step makes sure that no errors were made during the final log synchronization.

A simple log- shipping migration can be configured out of band with respect to the original database, which makes it particularly useful for mission-critical databases. No configuration changes are required for the source database, and the restoration and initial configuration of the migration environment have no effect on production operations. After log shipping is configured, it places some I/O demands on the production servers. However, log shipping consists of simple sequential reads of the archive logs, which is unlikely to have any effect on production database performance.

Log shipping has proven to be particularly useful for long-distance, high- change-rate migration projects. In one instance, a single 220TB database was migrated to a new location approximately 500 miles away. The change rate was extremely high and security restrictions prevented the use of a network connection. Log shipping was performed by using tape and courier. A copy of the source database was initially restored by using procedures outlined below. The logs were then shipped on a weekly basis by courier until the time of cutover when the final set of tapes was delivered and the logs were applied to the replica database.

Oracle DataGuard

In some cases, a complete DataGuard environment is warranted. It is incorrect to use the term DataGuard to refer to any log shipping or standby database configuration. Oracle DataGuard is a comprehensive framework for managing database replication, but it is not a replication technology. The primary benefit of a complete DataGuard environment in a migration effort is the transparent switchover from one database to another. DataGuard also enables a transparent switchover back to the original database if a problem is discovered, such as a performance or network connectivity issue with the new environment. A fully configured DataGuard environment requires configuration of not only the database layer but also applications so that applications are able to detect a change in the primary database location. In general, it is not necessary to use DataGuard to complete a migration, but some customers have extensive DataGuard expertise in-house and already rely on it for migration work.

Rearchitecture

As discussed before, leveraging the advanced features of storage arrays sometimes requires changing the database layout. Furthermore, a change in storage protocol such as moving from ASM to an NFS file system necessarily alters the file system layout.

One of the principal advantages of log shipping methods, including DataGuard, is that the replication destination does not have to match the source. There are no issues with using a log-shipping approach to migrate from ASM to a regular file system or vice versa. The precise layout of data files can be changed at the destination to optimize the use of Pluggable Database (PDB) technology or to set QoS controls selectively on certain files. In other words, a migration process based on log shipping allows you to optimize the database storage layout easily and safely.

Server resources

One limitation to database-level migration is the need for a second server. There are two ways this second server can be used:

1. You can use the second server as a permanent new home for the database.
2. You can use the second server as a temporary staging server. After data migration to the new storage array is complete and tested, the LUN or NFS file systems are disconnected from the staging server and

reconnected to the original server.

The first option is the easiest, but using it might not be feasible in very large environments requiring very powerful servers. The second option requires extra work to relocate the file systems back to the original location. This can be a simple operation in which NFS is used as the storage protocol because the file systems can be unmounted from the staging server and remounted on the original server.

Block-based file systems require extra work to update FC zoning or iSCSI initiators. With most logical volume managers (including ASM), the LUNs are automatically detected and brought online after they are made available on the original server. However, some file system and LVM implementations might require more work to export and import the data. The precise procedure might vary, but it is generally easy to establish a simple, repeatable procedure to complete the migration and rehome the data on the original server.

Although it is possible to set up log shipping and replicate a database within a single server environment, the new instance must have a different process SID to replay the logs. It is possible to temporarily bring up the database under a different set of process IDs with a different SID and change it later. However, doing so can lead to a lot of complicated management activities, and it puts the database environment at risk of user error.

Host-level migration

Migrating data at the host level means using the host operating system and associated utilities to complete the migration. This process includes any utility that copies data, including Oracle RMAN and Oracle ASM.

Data copying

The value of a simple copy operation should not be underestimated. Modern network infrastructures can move data at rates measured in gigabytes per second, and file copy operations are based on efficient sequential read and write I/O. More disruption is unavoidable with a host copy operation when compared to log shipping, but a migration is more than just the data movement. It generally includes changes to networking, the database restart time, and postmigration testing.

The actual time required to copy data might not be significant. Furthermore, a copy operation preserves a guaranteed back- out path because the original data remains untouched. If any problems are encountered during the migration process, the original file systems with the original data can be reactivated.

Replatforming

Replatforming refers to a change in the CPU type. When a database is migrated from a traditional Solaris, AIX, or HP-UX platform to x86 Linux, the data must be reformatted because of changes in the CPU architecture. SPARC, IA64, and POWER CPUs are known as big endian processors, while the x86 and x86_64 architectures are known as little endian. As a result, some data within Oracle data files is ordered differently depending on the processor in use.

Traditionally, customers have used DataPump to replicate data across platforms. DataPump is a utility that creates a special type of logical data export that can be more rapidly imported at the destination database. Because it creates a logical copy of the data, DataPump leaves the dependencies of processor endianness behind. DataPump is still used by some customers for replatforming, but a faster option has become available with Oracle 11g: cross-platform transportable tablespaces. This advance allows a tablespace to be converted to a different endian format in place. This is a physical transformation that offers better performance than a DataPump export, which must convert physical bytes to logical data and then convert back to physical bytes.

A complete discussion of DataPump and transportable tablespaces is beyond the scope NetApp documentation, but NetApp has some recommendations based on our experience assisting customers during migration to a new storage array log with a new CPU architecture:

- If DataPump is being used, the time required to complete the migration should be measured in a test environment. Customers are sometimes surprised at the time required to complete the migration. This unexpected additional downtime can cause disruption.
- Many customers mistakenly believe that cross-platform transportable tablespaces do not require data conversion. When a CPU with a different endian is used, an RMAN `convert` operation must be performed on the data files beforehand. This is not an instantaneous operation. In some cases, the conversion process can be sped up by having multiple threads operating on different data files, but the conversion process cannot be avoided.

Logical volume manager-driven migration

LVMs work by taking a group of one or more LUNs and breaking them into small units generally referred to as extents. The pool of extents is then used as a source to create logical volumes that are essentially virtualized. This virtualization layer delivers value in various ways:

- Logical volumes can use extents drawn from multiple LUNs. When a file system is created on a logical volume, it can use the full performance capabilities of all LUNs. It also promotes the even loading of all LUNs in the volume group, delivering more predictable performance.
- Logical volumes can be resized by adding and, in some cases, removing extents. Resizing a file system on a logical volume is generally nondisruptive.
- Logical volumes can be nondisruptively migrated by moving the underlying extents.

Migration using an LVM works in one of two ways: moving an extent or mirroring/demirroring an extent. LVM migration uses efficient large-block sequential I/O and only rarely creates any performance concerns. If this does become an issue, there are usually options for throttling the I/O rate. Doing so increases the time required to complete the migration and yet reduces the I/O burden on the host and storage systems.

Mirror and demirror

Some volume managers, such as AIX LVM, allow the user to specify the number of copies for each extent and to control which devices host each copy. Migration is accomplished by taking an existing logical volume, mirroring the underlying extents to the new volumes, waiting for the copies to synchronize, and then dropping the old copy. If a back- out path is desired, a snapshot of the original data can be created before the point at which the mirror copy is dropped. Alternatively, the server can be shut down briefly to mask original LUNs before forcibly deleting the contained mirror copies. Doing so preserves a recoverable copy of the data in its original location.

Extent migration

Almost all volume managers allow extents to be migrated, and sometimes multiple options exist. For example, some volume managers allow an administrator to relocate the individual extents for a specific logical volume from old to new storage. Volume managers such as Linux LVM2 offer the `pvmove` command, which relocates all extents on the specified LUN device to a new LUN. After the old LUN is evacuated, it can be removed.

The primary risk to operations is the removal of old, unused LUNs from the configuration. Great care must be taken when changing FC zoning and removing stale LUN devices.

Oracle Automatic Storage Management

Oracle ASM is a combined logical volume manager and file system. At a high level, Oracle ASM takes a collection of LUNs, breaks them into small units of allocation, and presents them as a single volume known as an ASM disk group. ASM also includes the ability to mirror the disk group by setting the redundancy level. A volume can be unmirrored (external redundancy), mirrored (normal redundancy), or three-way mirrored (high

redundancy). Care must be taken when configuring the redundancy level because it cannot be changed after creation.

ASM also provides file system functionality. Although the file system is not visible directly from the host, the Oracle database can create, move, and delete files and directories on an ASM disk group. Also, the structure can be navigated by using the asmcmd utility.

As with other LVM implementations, Oracle ASM optimizes I/O performance by striping and load-balancing the I/O of each file across all available LUNs. Second, the underlying extents can be relocated to enable both resizing of the ASM disk group as well as migration. Oracle ASM automates the process through the rebalancing operation. New LUNs are added to an ASM disk group and old LUNs are dropped, which triggers extent relocation and subsequent drop of the evacuated LUN from the disk group. This process is one of the most proven methods of migration, and the reliability of ASM at delivering transparent migration is possibly its most important feature.

Because the mirroring level of Oracle ASM is fixed, it cannot be used with the mirror and demirror method of migration.

Storage-level migration

Storage-level migration means performing the migration below both the application and operating system level. In the past, this sometimes meant using specialized devices that would copy LUNs at the network level, but these capabilities are now found natively in ONTAP.

SnapMirror

Migration of databases from between NetApp systems is almost universally performed with the NetApp SnapMirror data replication software. The process involves setting up a mirror relationship for the volumes to be migrated, allowing them to synchronize, and then waiting for the cutover window. When it arrives, the source database is shut down, one final mirror update is performed, and the mirror is broken. The replica volumes are then ready for use, either by mounting a contained NFS file system directory or by discovering the contained LUNs and starting the database.

Relocating volumes within a single ONTAP cluster is not considered migration, but rather a routine `volume move` operation. SnapMirror is used as the data replication engine within the cluster. This process is fully automated. There are no additional migration steps to be performed when attributes of the volume, such as LUN mapping or the NFS export permissions, are moved with the volume itself. The relocation is nondisruptive to host operations. In some cases, network access must be updated to make sure that the newly relocated data is accessed in the most efficient way possible, but these tasks are also nondisruptive.

Foreign LUN Import (FLI)

FLI is a feature that allows a Data ONTAP system running 8.3 or higher to migrate an existing LUN from another storage array. The procedure is simple: The ONTAP system is zoned to the existing storage array as if it was any other SAN host. Data ONTAP then takes control of the desired legacy LUNs and migrates the underlying data. In addition, the import process uses the efficiency settings of the new volume as data is migrated, meaning that data can be compressed and deduplicated inline during the migration process.

The first implementation of FLI in Data ONTAP 8.3 permitted only offline migration. This was an extremely fast transfer, but it still meant that the LUN data was unavailable until the migration was complete. Online migration was introduced in Data ONTAP 8.3.1. This kind of migration minimizes disruption by allowing ONTAP to serve LUN data during the transfer process. There is a brief disruption while the host is rezoned to use the LUNs through ONTAP. However, as soon as those changes are made, the data is once again accessible and remains accessible throughout the migration process.

Read I/O is proxied through ONTAP until the copy operation is complete, while write I/O is synchronously written to both the foreign and ONTAP LUN. The two LUN copies are kept in sync in this manner until the administrator executes a complete cutover that releases the foreign LUN and no longer replicates writes.

FLI is designed to work with FC, but if there is a desire to change to iSCSI, then the migrated LUN can easily be remapped as an iSCSI LUN after migration is completed.

Among the features of FLI is automatic alignment detection and adjustment. In this context, the term alignment refers to a partition on a LUN device. Optimum performance requires that I/O be aligned to 4K blocks. If a partition is placed at an offset that is not a multiple of 4K, performance suffers.

There is a second aspect of alignment that cannot be corrected by adjusting a partition offset—the file system block size. For example, a ZFS file system generally defaults to an internal block size of 512 bytes. Other customers using AIX have occasionally created jfs2 file systems with a 512- or 1, 024- byte block size. Although the file system might be aligned to a 4K boundary, the files created within that file system are not and performance suffers.

FLI should not be used in these circumstances. Although the data is accessible after migration, the result is file systems with serious performance limitations. As a general principle, any file system supporting a random overwrite workload on ONTAP should use a 4K block size. This is primarily applicable to workloads such as database data files and VDI deployments. The block size can be identified using the relevant host operating system commands.

For example, on AIX, the block size can be viewed with `lsfs -q`. With Linux, `xfs_info` and `tune2fs` can be used for `xfs` and `ext3/ext4`, respectively. With `zfs`, the command is `zdb -C`.

The parameter that controls the block size is `ashift` and generally defaults to a value of 9, which means 2^9, or 512 bytes. For optimum performance, the `ashift` value must be 12 (2^12=4K). This value is set at the time the zpool is created and cannot be changed, which means that data zpools with an `ashift` other than 12 should be migrated by copying data to a newly created zpool.

Oracle ASM does not have a fundamental block size. The only requirement is that the partition on which the ASM disk is built must be properly aligned.

7-Mode Transition Tool

The 7-Mode Transition Tool (7MTT) is an automation utility used to migrate large 7- Mode configurations to ONTAP. Most database customers find other methods easier, in part because they usually migrate their environments database by database rather than relocating the entire storage footprint. Additionally, databases are frequently only a part of a larger storage environment. Therefore, databases are often migrated individually, and then the remaining environment can be moved with 7MTT.

There is a small but significant number of customers who have storage systems that are dedicated to complicated database environments. These environments might contain many volumes, snapshots, and numerous configuration details such as export permissions, LUN initiator groups, user permissions, and Lightweight Directory Access Protocol configuration. In such cases, the automation abilities of 7MTT can simplify a migration.

7MTT can operate in one of two modes:

- **Copy- based transition (CBT).** 7MTT with CBT sets up SnapMirror volumes from an existing 7- Mode system in the new environment. After the data is in sync, 7MTT orchestrates the cutover process.
- **Copy- free transition (CFT).** 7MTT with CFT is based on the in-place conversion of existing 7- Mode disk shelves. No data is copied, and the existing disk shelves can be reused. The existing data protection and

storage efficiency configuration is preserved.

The primary difference between these two options is that copy-free transition is a big- bang approach in which all disk shelves attached to the original 7- Mode HA pair must be relocated to the new environment. There is no option to move a subset of shelves. The copy-based approach allows selected volumes to be moved. There is also potentially a longer cutover window with copy-free transition because of the tie required to recable disk shelves and convert metadata. Based on field experience, NetApp recommends allowing 1 hour for relocating and recabling disk shelves and between 15 minutes and 2 hours for metadata conversion.

Oracle datafile migration

Individual Oracle datafiles can be moved with a single command.

For example, the following command moves the datafile IOPST.dbf from filesystem `/oradata2` to filesystem `/oradata3`.

```
SQL> alter database move datafile  '/oradata2/NTAP/IOPS002.dbf' to
'/oradata3/NTAP/IOPS002.dbf';
Database altered.
```

Moving a datafile with this method can be slow, but it normally should not produce enough I/O that it interferes with the day-to-day database workloads. In contrast, migration via ASM rebalancing can run much faster but at the expense of slowing down the overall database while the data is being moved.

The time required to move datafiles can easily be measured by creating a test datafile and then moving it. The elapsed time for the operation is recorded in the v$session data:

```
SQL> set linesize 300;
SQL> select elapsed_seconds||':'||message from v$session_longops;
ELAPSED_SECONDS||':'||MESSAGE
------------------------------------------------------------------------
---------------
351:Online data file move: data file 8: 22548578304 out of 22548578304
bytes done
SQL> select bytes / 1024 / 1024 /1024 as GB from dba_data_files where
FILE_ID = 8;
        GB
----------
        21
```

In this example, the file that was moved was datafile 8, which was 21GB in size and required about 6 minutes to migrate. The time required obviously depends on the capabilities of the storage system, the storage network, and the overall database activity occurring at the time of migration.

Oracle database migration via log shipping

The goal of a migration using log shipping is to create a copy of the original data files at a new location and then establish a method of shipping changes into the new environment.

Once established, log shipment and replay can be automated to keep the replica database largely in sync with the source. For example, a cron job can be scheduled to (a) copy the most recent logs to the new location and (b) replay them every 15 minutes. Doing so provides minimal disruption at the time of cutover because no more than 15 minutes of archive logs must be replayed.

The procedure shown below is also essentially a database clone operation. The logic shown is similar to the engine within NetApp SnapManager for Oracle (SMO) and the NetApp SnapCenter Oracle Plug- in. Some customers have used the procedure shown within scripts or WFA workflows for custom cloning operations. Although this procedure is more manual than using either SMO or SnapCenter, it is still readily scripted and the data management APIs within ONTAP further simplify the process.

Log shipping - file system to file system

This example demonstrates the migration of a database called WAFFLE from an ordinary file system to another ordinary file system located on a different server. It also illustrates the use of SnapMirror to make a rapid copy of data files, but this is not an integral part of the overall procedure.

Create database backup

The first step is to create a database backup. Specifically, this procedure requires a set of data files that can be used for archive log replay.

Environment

In this example, the source database is on an ONTAP system. The simplest method to create a backup of a database is by using a snapshot. The database is placed in hot backup mode for a few seconds while a `snapshot create` operation is executed on the volume hosting the data files.

```
SQL> alter database begin backup;
Database altered.
```

```
Cluster01::*> snapshot create -vserver vserver1 -volume jfsc1_oradata
hotbackup
Cluster01::*>
```

```
SQL> alter database end backup;
Database altered.
```

The result is a snapshot on disk called `hotbackup` that contains an image of the data files while in hot backup mode. When combined with the appropriate archive logs to make the data files consistent, the data in this snapshot can be used as the basis of a restore or a clone. In this case, it is replicated to the new server.

Restore to new environment

The backup must now be restored in the new environment. This can be done in a number of ways, including Oracle RMAN, restoration from a backup application like NetBackup, or a simple copy operation of data files that were placed in hot backup mode.

In this example, SnapMirror is used to replicate the snapshot hotbackup to a new location.

1. Create a new volume to receive the snapshot data. Initialize the mirroring from `jfsc1_oradata` to `vol_oradata`.

```
Cluster01::*> volume create -vserver vserver1 -volume vol_oradata
-aggregate data_01 -size 20g -state online -type DP -snapshot-policy
none -policy jfsc3
[Job 833] Job succeeded: Successful
```

```
Cluster01::*> snapmirror initialize -source-path vserver1:jfsc1_oradata
-destination-path vserver1:vol_oradata
Operation is queued: snapmirror initialize of destination
"vserver1:vol_oradata".
Cluster01::*> volume mount -vserver vserver1 -volume vol_oradata
-junction-path /vol_oradata
Cluster01::*>
```

2. After the state is set by SnapMirror, indicating that synchronization is complete, update the mirror based specifically on the desired snapshot.

```
Cluster01::*> snapmirror show -destination-path vserver1:vol_oradata
-fields state
source-path             destination-path        state
----------------------- ----------------------- ------------
vserver1:jfsc1_oradata vserver1:vol_oradata SnapMirrored
```

```
Cluster01::*> snapmirror update -destination-path vserver1:vol_oradata
-source-snapshot hotbackup
Operation is queued: snapmirror update of destination
"vserver1:vol_oradata".
```

3. Successful synchronization can be verified by viewing the `newest-snapshot` field on the mirror volume.

```
Cluster01::*> snapmirror show -destination-path vserver1:vol_oradata
-fields newest-snapshot
source-path             destination-path        newest-snapshot
----------------------- ----------------------- ---------------
vserver1:jfsc1_oradata vserver1:vol_oradata hotbackup
```

4. The mirror can then be broken.

```
Cluster01::> snapmirror break -destination-path vserver1:vol_oradata
Operation succeeded: snapmirror break for destination
"vserver1:vol_oradata".
Cluster01::>
```

5. Mount the new file system.With block-based file systems, the precise procedures vary based on the LVM in use. FC zoning or iSCSI connections must be configured. After connectivity to the LUNs is established, commands such as Linux `pvscan` might be needed to discover which volume groups or LUNs need to be properly configured to be discoverable by ASM.

   In this example, a simple NFS file system is used. This file system can be mounted directly.

```
fas8060-nfs1:/vol_oradata        19922944   1639360   18283584   9%
/oradata
fas8060-nfs1:/vol_logs            9961472       128    9961344   1%
/logs
```

Create controlfile creation template

You must next create a controlfile template. The `backup controlfile to trace` command creates text commands to recreate a controlfile. This function can be useful for restoring a database from backup under some circumstances, and it is often used with scripts that perform tasks such as database cloning.

1. The output of the following command is used to recreate the controlfiles for the migrated database.

```
SQL> alter database backup controlfile to trace as '/tmp/waffle.ctrl';
Database altered.
```

2. After the controlfiles have been created, copy the file to the new server.

```
[oracle@jfsc3 tmp]$ scp oracle@jfsc1:/tmp/waffle.ctrl /tmp/
oracle@jfsc1's password:
waffle.ctrl                                           100% 5199
5.1KB/s   00:00
```

Backup parameter file

A parameter file is also required in the new environment. The simplest method is to create a pfile from the current spfile or pfile. In this example, the source database is using an spfile.

```
SQL> create pfile='/tmp/waffle.tmp.pfile' from spfile;
File created.
```

Create oratab entry

The creation of an oratab entry is required for the proper functioning of utilities such as oraenv. To create an oratab entry, complete the following step.

```
WAFFLE:/orabin/product/12.1.0/dbhome_1:N
```

Prepare directory structure

If the required directories were not already present, you must create them or the database startup procedure fails. To prepare the directory structure, complete the following minimum requirements.

```
[oracle@jfsc3 ~]$ . oraenv
ORACLE_SID = [oracle] ? WAFFLE
The Oracle base has been set to /orabin
[oracle@jfsc3 ~]$ cd $ORACLE_BASE
[oracle@jfsc3 orabin]$ cd admin
[oracle@jfsc3 admin]$ mkdir WAFFLE
[oracle@jfsc3 admin]$ cd WAFFLE
[oracle@jfsc3 WAFFLE]$ mkdir adump dpdump pfile scripts xdb_wallet
```

Parameter file updates

1. To copy the parameter file to the new server, run the following commands. The default location is the `$ORACLE_HOME/dbs` directory. In this case, the pfile can be placed anywhere. It is only being used as an intermediate step in the migration process.

```
[oracle@jfsc3 admin]$ scp oracle@jfsc1:/tmp/waffle.tmp.pfile
$ORACLE_HOME/dbs/waffle.tmp.pfile
oracle@jfsc1's password:
waffle.pfile                                          100%  916
0.9KB/s   00:00
```

1. Edit the file as required. For example, if the archive log location has changed, the pfile must be altered to reflect the new location. In this example, only the controlfiles are being relocated, in part to distribute them between the log and data file systems.

```
[root@jfsc1 tmp]# cat waffle.pfile
WAFFLE.__data_transfer_cache_size=0
WAFFLE.__db_cache_size=507510784
WAFFLE.__java_pool_size=4194304
WAFFLE.__large_pool_size=20971520
WAFFLE.__oracle_base='/orabin'#ORACLE_BASE set from environment
WAFFLE.__pga_aggregate_target=268435456
WAFFLE.__sga_target=805306368
WAFFLE.__shared_io_pool_size=29360128
WAFFLE.__shared_pool_size=234881024
WAFFLE.__streams_pool_size=0
*.audit_file_dest='/orabin/admin/WAFFLE/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='/oradata//WAFFLE/control01.ctl','/oradata//WAFFLE/contr
ol02.ctl'
*.control_files='/oradata/WAFFLE/control01.ctl','/logs/WAFFLE/control02.
ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='WAFFLE'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=WAFFLEXDB)'
*.log_archive_dest_1='LOCATION=/logs/WAFFLE/arch'
*.log_archive_format='%t_%s_%r.dbf'
*.open_cursors=300
*.pga_aggregate_target=256m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=768m
*.undo_tablespace='UNDOTBS1'
```

2. After the edits are complete, create an spfile based on this pfile.

```
SQL> create spfile from pfile='waffle.tmp.pfile';
File created.
```

Recreate controlfiles

In a previous step, the output of `backup controlfile to trace` was copied to the new server. The specific portion of the output required is the `controlfile recreation` command. This information can be found in the file under the section marked `Set #1. NORESETLOGS`. It starts with the line `create controlfile reuse database` and should include the word `noresetlogs`. It ends with the semicolon (;) character.

1. In this example procedure, the file reads as follows.

```
CREATE CONTROLFILE REUSE DATABASE "WAFFLE" NORESETLOGS  ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/logs/WAFFLE/redo/redo01.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 2 '/logs/WAFFLE/redo/redo02.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 3 '/logs/WAFFLE/redo/redo03.log'  SIZE 50M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  '/oradata/WAFFLE/system01.dbf',
  '/oradata/WAFFLE/sysaux01.dbf',
  '/oradata/WAFFLE/undotbs01.dbf',
  '/oradata/WAFFLE/users01.dbf'
CHARACTER SET WE8MSWIN1252
;
```

2. Edit this script as desired to reflect the new location of the various files. For example, certain data files known to support high I/O might be redirected to a file system on a high- performance storage tier. In other cases, the changes might be purely for administrator reasons, such as isolating the data files of a given PDB in dedicated volumes.
3. In this example, the `DATAFILE` stanza is left unchanged, but the redo logs are moved to a new location in `/redo` rather than sharing space with archive logs in `/logs`.

```
CREATE CONTROLFILE REUSE DATABASE "WAFFLE" NORESETLOGS  ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/redo/redo01.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 2 '/redo/redo02.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 3 '/redo/redo03.log'  SIZE 50M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  '/oradata/WAFFLE/system01.dbf',
  '/oradata/WAFFLE/sysaux01.dbf',
  '/oradata/WAFFLE/undotbs01.dbf',
  '/oradata/WAFFLE/users01.dbf'
CHARACTER SET WE8MSWIN1252
;
```

```
SQL> startup nomount;
ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             331353200 bytes
Database Buffers          465567744 bytes
Redo Buffers                5455872 bytes
SQL> CREATE CONTROLFILE REUSE DATABASE "WAFFLE" NORESETLOGS  ARCHIVELOG
  2      MAXLOGFILES 16
  3      MAXLOGMEMBERS 3
  4      MAXDATAFILES 100
  5      MAXINSTANCES 8
  6      MAXLOGHISTORY 292
  7  LOGFILE
  8    GROUP 1 '/redo/redo01.log'  SIZE 50M BLOCKSIZE 512,
  9    GROUP 2 '/redo/redo02.log'  SIZE 50M BLOCKSIZE 512,
 10    GROUP 3 '/redo/redo03.log'  SIZE 50M BLOCKSIZE 512
 11  -- STANDBY LOGFILE
 12  DATAFILE
 13    '/oradata/WAFFLE/system01.dbf',
 14    '/oradata/WAFFLE/sysaux01.dbf',
 15    '/oradata/WAFFLE/undotbs01.dbf',
 16    '/oradata/WAFFLE/users01.dbf'
 17  CHARACTER SET WE8MSWIN1252
 18  ;
Control file created.
SQL>
```

If any files are misplaced or parameters are misconfigured, errors are generated that indicate what must be fixed. The database is mounted, but it is not yet open and cannot be opened because the data files in use are still marked as being in hot backup mode. Archive logs must first be applied to make the database consistent.

Initial log replication

At least one log reply operation is required to make the data files consistent. Many options are available to replay logs. In some cases, the original archive log location on the original server can be shared through NFS, and log reply can be done directly. In other cases, the archive logs must be copied.

For example, a simple `scp` operation can copy all current logs from the source server to the migration server:

```
[oracle@jfsc3 arch]$ scp jfsc1:/logs/WAFFLE/arch/* ./
oracle@jfsc1's password:
1_22_912662036.dbf                                          100%   47MB
47.0MB/s   00:01
1_23_912662036.dbf                                          100%   40MB
40.4MB/s   00:00
1_24_912662036.dbf                                          100%   45MB
45.4MB/s   00:00
1_25_912662036.dbf                                          100%   41MB
40.9MB/s   00:01
1_26_912662036.dbf                                          100%   39MB
39.4MB/s   00:00
1_27_912662036.dbf                                          100%   39MB
38.7MB/s   00:00
1_28_912662036.dbf                                          100%   40MB
40.1MB/s   00:01
1_29_912662036.dbf                                          100%   17MB
16.9MB/s   00:00
1_30_912662036.dbf                                          100%  636KB
636.0KB/s   00:00
```

Initial log replay

After the files are in the archive log location, they can be replayed by issuing the command `recover database until cancel` followed by the response `AUTO` to automatically replay all available logs.

```
SQL> recover database until cancel;
ORA-00279: change 382713 generated at 05/24/2016 09:00:54 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_23_912662036.dbf
ORA-00280: change 382713 for thread 1 is in sequence #23
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
AUTO
ORA-00279: change 405712 generated at 05/24/2016 15:01:05 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_24_912662036.dbf
ORA-00280: change 405712 for thread 1 is in sequence #24
ORA-00278: log file '/logs/WAFFLE/arch/1_23_912662036.dbf' no longer
needed for
this recovery
...
ORA-00279: change 713874 generated at 05/26/2016 04:26:43 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_31_912662036.dbf
ORA-00280: change 713874 for thread 1 is in sequence #31
ORA-00278: log file '/logs/WAFFLE/arch/1_30_912662036.dbf' no longer
needed for
this recovery
ORA-00308: cannot open archived log '/logs/WAFFLE/arch/1_31_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

The final archive log reply reports an error, but this is normal. The log indicates that `sqlplus` was seeking a particular log file and did not find it. The reason is, most likely, that the log file does not exist yet.

If the source database can be shut down before copying archive logs, this step must be performed only once. The archive logs are copied and replayed, and then the process can continue directly to the cutover process that replicates the critical redo logs.

Incremental log replication and replay

In most cases, migration is not performed right away. It could be days or even weeks before the migration process is completed, which means that the logs must be continuously shipped to the replica database and replayed. Therefore, when cutover arrives, minimal data must be transferred and replayed.

Doing so can be scripted in many ways, but one of the more popular methods is using rsync, a common file replication utility. The safest way to use this utility is to configure it as a daemon. For example, the `rsyncd.conf` file that follows shows how to create a resource called `waffle.arch` that is accessed with Oracle user credentials and is mapped to `/logs/WAFFLE/arch`. Most importantly, the resource is set to read-only, which allows the production data to be read but not altered.

```
[root@jfsc1 arch]# cat /etc/rsyncd.conf
[waffle.arch]
   uid=oracle
   gid=dba
   path=/logs/WAFFLE/arch
   read only = true
[root@jfsc1 arch]# rsync --daemon
```

The following command synchronizes the new server's archive log destination against the rsync resource `waffle.arch` on the original server. The `t` argument in `rsync - potg` causes the file list to be compared based on timestamp, and only new files are copied. This process provides an incremental update of the new server. This command can also be scheduled in cron to run on a regular basis.

```
[oracle@jfsc3 arch]$ rsync -potg --stats --progress jfsc1::waffle.arch/*
/logs/WAFFLE/arch/
1_31_912662036.dbf
      650240 100%  124.02MB/s    0:00:00 (xfer#1, to-check=8/18)
1_32_912662036.dbf
     4873728 100%  110.67MB/s    0:00:00 (xfer#2, to-check=7/18)
1_33_912662036.dbf
     4088832 100%   50.64MB/s    0:00:00 (xfer#3, to-check=6/18)
1_34_912662036.dbf
     8196096 100%   54.66MB/s    0:00:00 (xfer#4, to-check=5/18)
1_35_912662036.dbf
    19376128 100%   57.75MB/s    0:00:00 (xfer#5, to-check=4/18)
1_36_912662036.dbf
       71680 100%  201.15kB/s    0:00:00 (xfer#6, to-check=3/18)
1_37_912662036.dbf
     1144320 100%    3.06MB/s    0:00:00 (xfer#7, to-check=2/18)
1_38_912662036.dbf
    35757568 100%   63.74MB/s    0:00:00 (xfer#8, to-check=1/18)
1_39_912662036.dbf
      984576 100%    1.63MB/s    0:00:00 (xfer#9, to-check=0/18)
Number of files: 18
Number of files transferred: 9
Total file size: 399653376 bytes
Total transferred file size: 75143168 bytes
Literal data: 75143168 bytes
Matched data: 0 bytes
File list size: 474
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 204
Total bytes received: 75153219
sent 204 bytes  received 75153219 bytes  150306846.00 bytes/sec
total size is 399653376  speedup is 5.32
```

After the logs have been received, they must be replayed. Previous examples show the use of sqlplus to manually run `recover database until cancel`, a process that can easily be automated. The example shown here uses the script described in Replay Logs on Database. The scripts accept an argument that specifies the database requiring a replay operation. This permits the same script to be used in a multidatabase migration effort.

```
[oracle@jfsc3 logs]$ ./replay.logs.pl WAFFLE
ORACLE_SID = [WAFFLE] ? The Oracle base remains unchanged with value
/orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu May 26 10:47:16 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> ORA-00279: change 713874 generated at 05/26/2016 04:26:43 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_31_912662036.dbf
ORA-00280: change 713874 for thread 1 is in sequence #31
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 814256 generated at 05/26/2016 04:52:30 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_32_912662036.dbf
ORA-00280: change 814256 for thread 1 is in sequence #32
ORA-00278: log file '/logs/WAFFLE/arch/1_31_912662036.dbf' no longer
needed for
this recovery
ORA-00279: change 814780 generated at 05/26/2016 04:53:04 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_33_912662036.dbf
ORA-00280: change 814780 for thread 1 is in sequence #33
ORA-00278: log file '/logs/WAFFLE/arch/1_32_912662036.dbf' no longer
needed for
this recovery
...
ORA-00279: change 1120099 generated at 05/26/2016 09:59:21 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_40_912662036.dbf
ORA-00280: change 1120099 for thread 1 is in sequence #40
ORA-00278: log file '/logs/WAFFLE/arch/1_39_912662036.dbf' no longer
needed for
this recovery
ORA-00308: cannot open archived log '/logs/WAFFLE/arch/1_40_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
```

Cutover

When you are ready to cut over to the new environment, you must perform one final synchronization that includes both archive logs and the redo logs. If the original redo log location is not already known, it can be identified as follows:

```
SQL> select member from v$logfile;
MEMBER
--------------------------------------------------------------------------------
------
/logs/WAFFLE/redo/redo01.log
/logs/WAFFLE/redo/redo02.log
/logs/WAFFLE/redo/redo03.log
```

1. Shut down the source database.
2. Perform one final synchronization of the archive logs on the new server with the desired method.
3. The source redo logs must be copied to the new server. In this example, the redo logs were relocated to a new directory at `/redo`.

   ```
   [oracle@jfsc3 logs]$ scp jfsc1:/logs/WAFFLE/redo/* /redo/
   oracle@jfsc1's password:
   redo01.log
   100%   50MB  50.0MB/s   00:01
   redo02.log
   100%   50MB  50.0MB/s   00:00
   redo03.log
   100%   50MB  50.0MB/s   00:00
   ```

4. At this stage, the new database environment contains all of the files required to bring it to the exact same state as the source. The archive logs must be replayed one final time.

```
SQL> recover database until cancel;
ORA-00279: change 1120099 generated at 05/26/2016 09:59:21 needed for
thread 1
ORA-00289: suggestion : /logs/WAFFLE/arch/1_40_912662036.dbf
ORA-00280: change 1120099 for thread 1 is in sequence #40
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
AUTO
ORA-00308: cannot open archived log
'/logs/WAFFLE/arch/1_40_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
ORA-00308: cannot open archived log
'/logs/WAFFLE/arch/1_40_912662036.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

5. Once complete, the redo logs must be replayed. If the message `Media recovery complete` is returned, the process is successful and the databases are synchronized and can be opened.

```
SQL> recover database;
Media recovery complete.
SQL> alter database open;
Database altered.
```

Log shipping - ASM to file system

This example demonstrates the use of Oracle RMAN to migrate a database. It is very similar to the prior example of file system to file system log shipping, but the files on ASM are not visible to the host. The only options for migrating data located on ASM devices is either by relocating the ASM LUN or by using Oracle RMAN to perform the copy operations.

Although RMAN is a requirement for copying files from Oracle ASM, the use of RMAN is not limited to ASM. RMAN can be used to migrate from any type of storage to any other type.

This example shows the relocation of a database called PANCAKE from ASM storage to a regular file system located on a different server at paths `/oradata` and `/logs`.

Create database backup

The first step is to create a backup of the database to be migrated to an alternate server. Because the source uses Oracle ASM, RMAN must be used. A simple RMAN backup can be performed as follows. This method creates a tagged backup that can be easily identified by RMAN later in the procedure.

The first command defines the type of destination for the backup and the location to be used. The second initiates the backup of the data files only.

```
RMAN> configure channel device type disk format '/rman/pancake/%U';
using target database control file instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/rman/pancake/%U';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/rman/pancake/%U';
new RMAN configuration parameters are successfully stored
RMAN> backup database tag 'ONTAP_MIGRATION';
Starting backup at 24-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=251 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=+ASM0/PANCAKE/system01.dbf
input datafile file number=00002 name=+ASM0/PANCAKE/sysaux01.dbf
input datafile file number=00003 name=+ASM0/PANCAKE/undotbs101.dbf
input datafile file number=00004 name=+ASM0/PANCAKE/users01.dbf
channel ORA_DISK_1: starting piece 1 at 24-MAY-16
channel ORA_DISK_1: finished piece 1 at 24-MAY-16
piece handle=/rman/pancake/1gr6c161_1_1 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 24-MAY-16
channel ORA_DISK_1: finished piece 1 at 24-MAY-16
piece handle=/rman/pancake/1hr6c164_1_1 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 24-MAY-16
```

Backup controlfile

A backup controlfile is required later in the procedure for the `duplicate database` operation.

```
RMAN> backup current controlfile format '/rman/pancake/ctrl.bkp';
Starting backup at 24-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
channel ORA_DISK_1: starting piece 1 at 24-MAY-16
channel ORA_DISK_1: finished piece 1 at 24-MAY-16
piece handle=/rman/pancake/ctrl.bkp tag=TAG20160524T032651 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 24-MAY-16
```

Backup parameter file

A parameter file is also required in the new environment. The simplest method is to create a pfile from the current spfile or pfile. In this example, the source database uses an spfile.

```
RMAN> create pfile='/rman/pancake/pfile' from spfile;
Statement processed
```

ASM file rename script

Several file locations currently defined in the controlfiles change when the database is moved. The following script creates an RMAN script to make the process easier. This example shows a database with a very small number of data files, but typically databases contain hundreds or even thousands of data files.

This script can be found in ASM to File System Name Conversion and it does two things.

First, it creates a parameter to redefine the redo log locations called `log_file_name_convert`. It is essentially a list of alternating fields. The first field is the location of a current redo log, and the second field is the location on the new server. The pattern is then repeated.

The second function is to supply a template for data file renaming. The script loops through the data files, pulls the name and file number information, and formats it as an RMAN script. Then it does the same with the temp files. The result is a simple rman script that can be edited as desired to make sure that the files are restored to the desired location.

```
SQL> @/rman/mk.rename.scripts.sql
Parameters for log file conversion:
*.log_file_name_convert = '+ASM0/PANCAKE/redo01.log',
'/NEW_PATH/redo01.log','+ASM0/PANCAKE/redo02.log',
'/NEW_PATH/redo02.log','+ASM0/PANCAKE/redo03.log', '/NEW_PATH/redo03.log'
rman duplication script:
run
{
set newname for datafile 1 to '+ASM0/PANCAKE/system01.dbf';
set newname for datafile 2 to '+ASM0/PANCAKE/sysaux01.dbf';
set newname for datafile 3 to '+ASM0/PANCAKE/undotbs101.dbf';
set newname for datafile 4 to '+ASM0/PANCAKE/users01.dbf';
set newname for tempfile 1 to '+ASM0/PANCAKE/temp01.dbf';
duplicate target database for standby backup location INSERT_PATH_HERE;
}
PL/SQL procedure successfully completed.
```

Capture the output of this screen. The `log_file_name_convert` parameter is placed in the pfile as described below. The RMAN data file rename and duplicate script must be edited accordingly to place the data files in the desired locations. In this example, they are all placed in `/oradata/pancake`.

```
run
{
set newname for datafile 1 to '/oradata/pancake/pancake.dbf';
set newname for datafile 2 to '/oradata/pancake/sysaux.dbf';
set newname for datafile 3 to '/oradata/pancake/undotbs1.dbf';
set newname for datafile 4 to '/oradata/pancake/users.dbf';
set newname for tempfile 1 to '/oradata/pancake/temp.dbf';
duplicate target database for standby backup location '/rman/pancake';
}
```

Prepare directory structure

The scripts are almost ready to execute, but first the directory structure must be in place. If the required directories are not already present, they must be created or the database startup procedure fails. The example below reflects the minimum requirements.

```
[oracle@jfsc2 ~]$ mkdir /oradata/pancake
[oracle@jfsc2 ~]$ mkdir /logs/pancake
[oracle@jfsc2 ~]$ cd /orabin/admin
[oracle@jfsc2 admin]$ mkdir PANCAKE
[oracle@jfsc2 admin]$ cd PANCAKE
[oracle@jfsc2 PANCAKE]$ mkdir adump dpdump pfile scripts xdb_wallet
```

Create oratab entry

The following command is required for utilities such as oraenv to work properly.

```
PANCAKE:/orabin/product/12.1.0/dbhome_1:N
```

Parameter updates

The saved pfile must be updated to reflect any path changes on the new server. The data file path changes are changed by the RMAN duplication script, and nearly all databases require changes to the `control_files` and `log_archive_dest` parameters. There might also be audit file locations that must be changed, and parameters such as `db_create_file_dest` might not be relevant outside of ASM. An experienced DBA should carefully review the proposed changes before proceeding.

In this example, the key changes are the controlfile locations, the log archive destination, and the addition of the `log_file_name_convert` parameter.

```
PANCAKE.__data_transfer_cache_size=0
PANCAKE.__db_cache_size=545259520
PANCAKE.__java_pool_size=4194304
PANCAKE.__large_pool_size=25165824
PANCAKE.__oracle_base='/orabin'#ORACLE_BASE set from environment
PANCAKE.__pga_aggregate_target=268435456
PANCAKE.__sga_target=805306368
PANCAKE.__shared_io_pool_size=29360128
PANCAKE.__shared_pool_size=192937984
PANCAKE.__streams_pool_size=0
*.audit_file_dest='/orabin/admin/PANCAKE/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='+ASM0/PANCAKE/control01.ctl','+ASM0/PANCAKE/control02.ctl
'
*.control_files='/oradata/pancake/control01.ctl','/logs/pancake/control02.
ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='PANCAKE'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=PANCAKEXDB)'
*.log_archive_dest_1='LOCATION=+ASM1'
*.log_archive_dest_1='LOCATION=/logs/pancake'
*.log_archive_format='%t_%s_%r.dbf'
'/logs/path/redo02.log'
*.log_file_name_convert = '+ASM0/PANCAKE/redo01.log',
'/logs/pancake/redo01.log', '+ASM0/PANCAKE/redo02.log',
'/logs/pancake/redo02.log', '+ASM0/PANCAKE/redo03.log',
'/logs/pancake/redo03.log'
*.open_cursors=300
*.pga_aggregate_target=256m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=768m
*.undo_tablespace='UNDOTBS1'
```

After the new parameters are confirmed, the parameters must be put into effect. Multiple options exist, but most customers create an spfile based on the text pfile.

```
bash-4.1$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0 Production on Fri Jan 8 11:17:40 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> create spfile from pfile='/rman/pancake/pfile';
File created.
```

Startup nomount

The final step before replicating the database is to bring up the database processes but not mount the files. In this step, problems with the spfile might become evident. If the `startup nomount` command fails because of a parameter error, it is simple to shut down, correct the pfile template, reload it as an spfile, and try again.

```
SQL> startup nomount;
ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             373296240 bytes
Database Buffers          423624704 bytes
Redo Buffers                5455872 bytes
```

Duplicate the database

Restoring the prior RMAN backup to the new location consumes more time than other steps in this process. The database must be duplicated without a change to the database ID (DBID) or resetting the logs. This prevents logs from being applied, which is a required step to fully synchronize the copies.

Connect to the database with RMAN as aux and issue the duplicate database command by using the script created in a previous step.

```
[oracle@jfsc2 pancake]$ rman auxiliary /
Recovery Manager: Release 12.1.0.2.0 - Production on Tue May 24 03:04:56
2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.
connected to auxiliary database: PANCAKE (not mounted)
RMAN> run
2> {
3> set newname for datafile 1 to '/oradata/pancake/pancake.dbf';
4> set newname for datafile 2 to '/oradata/pancake/sysaux.dbf';
5> set newname for datafile 3 to '/oradata/pancake/undotbs1.dbf';
6> set newname for datafile 4 to '/oradata/pancake/users.dbf';
7> set newname for tempfile 1 to '/oradata/pancake/temp.dbf';
8> duplicate target database for standby backup location '/rman/pancake';
9> }
```

```
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
Starting Duplicate Db at 24-MAY-16
contents of Memory Script:
{
   restore clone standby controlfile from  '/rman/pancake/ctrl.bkp';
}
executing Memory Script
Starting restore at 24-MAY-16
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=243 device type=DISK
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
output file name=/oradata/pancake/control01.ctl
output file name=/logs/pancake/control02.ctl
Finished restore at 24-MAY-16
contents of Memory Script:
{
   sql clone 'alter database mount standby database';
}
executing Memory Script
sql statement: alter database mount standby database
released channel: ORA_AUX_DISK_1
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=243 device type=DISK
contents of Memory Script:
{
   set newname for tempfile  1 to
 "/oradata/pancake/temp.dbf";
   switch clone tempfile all;
   set newname for datafile  1 to
 "/oradata/pancake/pancake.dbf";
   set newname for datafile  2 to
 "/oradata/pancake/sysaux.dbf";
   set newname for datafile  3 to
 "/oradata/pancake/undotbs1.dbf";
   set newname for datafile  4 to
 "/oradata/pancake/users.dbf";
   restore
   clone database
   ;
}
executing Memory Script
```

```
executing command: SET NEWNAME
renamed tempfile 1 to /oradata/pancake/temp.dbf in control file
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
Starting restore at 24-MAY-16
using channel ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/oradata/pancake/pancake.dbf
channel ORA_AUX_DISK_1: restoring datafile 00002 to
/oradata/pancake/sysaux.dbf
channel ORA_AUX_DISK_1: restoring datafile 00003 to
/oradata/pancake/undotbs1.dbf
channel ORA_AUX_DISK_1: restoring datafile 00004 to
/oradata/pancake/users.dbf
channel ORA_AUX_DISK_1: reading from backup piece
/rman/pancake/1gr6c161_1_1
channel ORA_AUX_DISK_1: piece handle=/rman/pancake/1gr6c161_1_1
tag=ONTAP_MIGRATION
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:07
Finished restore at 24-MAY-16
contents of Memory Script:
{
   switch clone datafile all;
}
executing Memory Script
datafile 1 switched to datafile copy
input datafile copy RECID=5 STAMP=912655725 file
name=/oradata/pancake/pancake.dbf
datafile 2 switched to datafile copy
input datafile copy RECID=6 STAMP=912655725 file
name=/oradata/pancake/sysaux.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=7 STAMP=912655725 file
name=/oradata/pancake/undotbs1.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=8 STAMP=912655725 file
name=/oradata/pancake/users.dbf
Finished Duplicate Db at 24-MAY-16
```

Initial log replication

You must now ship the changes from the source database to a new location. Doing so might require a combination of steps. The simplest method would be to have RMAN on the source database write out archive logs to a shared network connection. If a shared location is not available, an alternative method is using RMAN to write to a local file system and then using rcp or rsync to copy the files.

In this example, the `/rman` directory is an NFS share that is available to both the original and migrated database.

One important issue here is the `disk format` clause. The disk format of the backup is `%h_%e_%a.dbf`, which means that you must use the format of thread number, sequence number, and activation ID for the database. Although the letters are different, this matches the `log_archive_format='%t_%s_%r.dbf` parameter in the pfile. This parameter also specifies archive logs in the format of thread number, sequence number, and activation ID. The end result is that the log file backups on the source use a naming convention that is expected by the database. Doing so makes operations such as `recover database` much simpler because sqlplus correctly anticipates the names of the archive logs to be replayed.

```
RMAN> configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/arch/%h_%e_%a.dbf';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters are successfully stored
released channel: ORA_DISK_1
RMAN> backup as copy archivelog from time 'sysdate-2';
Starting backup at 24-MAY-16
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=373 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=70 STAMP=912658508
output file name=/rman/pancake/logship/1_54_912576125.dbf RECID=123
STAMP=912659482
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=41 RECID=29 STAMP=912654101
output file name=/rman/pancake/logship/1_41_912576125.dbf RECID=124
STAMP=912659483
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
...
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=45 RECID=33 STAMP=912654688
output file name=/rman/pancake/logship/1_45_912576125.dbf RECID=152
STAMP=912659514
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=36 STAMP=912654809
output file name=/rman/pancake/logship/1_47_912576125.dbf RECID=153
STAMP=912659515
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
Finished backup at 24-MAY-16
```

Initial log replay

After the files are in the archive log location, they can be replayed by issuing the command `recover database until cancel` followed by the response `AUTO` to automatically replay all available logs. The parameter file is currently directing archive logs to `/logs/archive`, but this does not match the location where RMAN was used to save logs. The location can be temporarily redirected as follows before recovering the database.

```
SQL> alter system set log_archive_dest_1='LOCATION=/rman/pancake/logship'
scope=memory;
System altered.
SQL> recover standby database until cancel;
ORA-00279: change 560224 generated at 05/24/2016 03:25:53 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_49_912576125.dbf
ORA-00280: change 560224 for thread 1 is in sequence #49
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
AUTO
ORA-00279: change 560353 generated at 05/24/2016 03:29:17 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_50_912576125.dbf
ORA-00280: change 560353 for thread 1 is in sequence #50
ORA-00278: log file '/rman/pancake/logship/1_49_912576125.dbf' no longer
needed
for this recovery
...
ORA-00279: change 560591 generated at 05/24/2016 03:33:56 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_54_912576125.dbf
ORA-00280: change 560591 for thread 1 is in sequence #54
ORA-00278: log file '/rman/pancake/logship/1_53_912576125.dbf' no longer
needed
for this recovery
ORA-00308: cannot open archived log
'/rman/pancake/logship/1_54_912576125.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

The final archive log reply reports an error, but this is normal. The error indicates that sqlplus was seeking a particular log file and did not find it. The reason is most likely that the log file does not yet exist.

If the source database can be shut down before copying archive logs, this step must be performed only once. The archive logs are copied and replayed, and then the process can continue directly to the cutover process that replicates the critical redo logs.

Incremental log replication and replay

In most cases, migration is not performed right away. It could be days or even weeks before the migration process is complete, which means that the logs must be continuously shipped to the replica database and replayed. Doing so makes sure that minimal data must be transferred and replayed when the cutover arrives.

This process can easily be scripted. For example, the following command can be scheduled on the original database to make sure that the location used for log shipping is continuously updated.

```
[oracle@jfsc1 pancake]$ cat copylogs.rman
configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
backup as copy archivelog from time 'sysdate-2';
```

```
[oracle@jfsc1 pancake]$ rman target / cmdfile=copylogs.rman
Recovery Manager: Release 12.1.0.2.0 - Production on Tue May 24 04:36:19
2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.
connected to target database: PANCAKE (DBID=3574534589)
RMAN> configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
2> backup as copy archivelog from time 'sysdate-2';
3>
4>
using target database control file instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters are successfully stored
Starting backup at 24-MAY-16
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=369 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=123 STAMP=912659482
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:22
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_54_912576125.dbf
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=41 RECID=124 STAMP=912659483
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:23
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_41_912576125.dbf
continuing other job steps, job failed will not be re-run
...
channel ORA_DISK_1: starting archived log copy
```

```
input archived log thread=1 sequence=45 RECID=152 STAMP=912659514
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:55
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_45_912576125.dbf
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=153 STAMP=912659515
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 05/24/2016
04:36:57
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_47_912576125.dbf
Recovery Manager complete.
```

After the logs have been received, they must be replayed. Previous examples showed the use of sqlplus to manually run `recover database until cancel`, which can be easily automated. The example shown here uses the script described in Replay Logs on Standby Database. The script accepts an argument that specifies the database requiring a replay operation. This process permits the same script to be used in a multidatabase migration effort.

```
[root@jfsc2 pancake]# ./replaylogs.pl PANCAKE
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Tue May 24 04:47:10 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> ORA-00279: change 560591 generated at 05/24/2016 03:33:56 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_54_912576125.dbf
ORA-00280: change 560591 for thread 1 is in sequence #54
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 562219 generated at 05/24/2016 04:15:08 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_55_912576125.dbf
ORA-00280: change 562219 for thread 1 is in sequence #55
ORA-00278: log file '/rman/pancake/logship/1_54_912576125.dbf' no longer
needed for this recovery
ORA-00279: change 562370 generated at 05/24/2016 04:19:18 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_56_912576125.dbf
ORA-00280: change 562370 for thread 1 is in sequence #56
ORA-00278: log file '/rman/pancake/logship/1_55_912576125.dbf' no longer
needed for this recovery
...
ORA-00279: change 563137 generated at 05/24/2016 04:36:20 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_65_912576125.dbf
ORA-00280: change 563137 for thread 1 is in sequence #65
ORA-00278: log file '/rman/pancake/logship/1_64_912576125.dbf' no longer
needed for this recovery
ORA-00308: cannot open archived log
'/rman/pancake/logship/1_65_912576125.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
```

Cutover

When you are ready to cut over to the new environment, you must perform one final synchronization. When working with regular file systems, it is easy to make sure that the migrated database is 100% synchronized against the original because the original redo logs are copied and replayed. There is no good way to do this with ASM. Only the archive logs can be easily recopied. To make sure that no data is lost, the final shutdown of the original database must be performed carefully.

1. First, the database must be quiesced, ensuring that no changes are being made. This quiescing might include disabling scheduled operations, shutting down listeners, and/or shutting down applications.
2. After this step is taken, most DBAs create a dummy table to serve as a marker of the shutdown.
3. Force a log archiving to make sure that the creation of the dummy table is recorded within the archive logs. To do so, run the following commands:

```
SQL> create table cutovercheck as select * from dba_users;
Table created.
SQL> alter system archive log current;
System altered.
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

4. To copy the last of the archive logs, run the following commands. The database must be available but not open.

```
SQL> startup mount;
ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             331353200 bytes
Database Buffers          465567744 bytes
Redo Buffers                5455872 bytes
Database mounted.
```

5. To copy the archive logs, run the following commands:

```
RMAN> configure channel device type disk format
'/rman/pancake/logship/%h_%e_%a.dbf';
2> backup as copy archivelog from time 'sysdate-2';
3>
4>
using target database control file instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/rman/pancake/logship/%h_%e_%a.dbf';
new RMAN configuration parameters are successfully stored
Starting backup at 24-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=8 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=123 STAMP=912659482
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
05/24/2016 04:58:24
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_54_912576125.dbf
continuing other job steps, job failed will not be re-run
...
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=45 RECID=152 STAMP=912659514
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
05/24/2016 04:58:58
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_45_912576125.dbf
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=153 STAMP=912659515
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
05/24/2016 04:59:00
ORA-19635: input and output file names are identical:
/rman/pancake/logship/1_47_912576125.dbf
```

6. Finally, replay the remaining archive logs on the new server.

```
[root@jfsc2 pancake]# ./replaylogs.pl PANCAKE
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Tue May 24 05:00:53 2016
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> ORA-00279: change 563137 generated at 05/24/2016 04:36:20 needed
for thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_65_912576125.dbf
ORA-00280: change 563137 for thread 1 is in sequence #65
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: change 563629 generated at 05/24/2016 04:55:20 needed for
thread 1
ORA-00289: suggestion : /rman/pancake/logship/1_66_912576125.dbf
ORA-00280: change 563629 for thread 1 is in sequence #66
ORA-00278: log file '/rman/pancake/logship/1_65_912576125.dbf' no longer
needed
for this recovery
ORA-00308: cannot open archived log
'/rman/pancake/logship/1_66_912576125.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
```

7. At this stage, replicate all data. The database is ready to be converted from a standby database to an active operational database and then opened.

```
SQL> alter database activate standby database;
Database altered.
SQL> alter database open;
Database altered.
```

8. Confirm the presence of the dummy table and then drop it.

```
SQL> desc cutovercheck
 Name                                      Null?    Type
 ----------------------------------------- --------
---------------------------
 USERNAME                                  NOT NULL VARCHAR2(128)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(4000)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(128)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(128)
 EXTERNAL_NAME                                      VARCHAR2(4000)
 PASSWORD_VERSIONS                                  VARCHAR2(12)
 EDITIONS_ENABLED                                   VARCHAR2(1)
 AUTHENTICATION_TYPE                                VARCHAR2(8)
 PROXY_ONLY_CONNECT                                 VARCHAR2(1)
 COMMON                                             VARCHAR2(3)
 LAST_LOGIN                                         TIMESTAMP(9) WITH
TIME ZONE
 ORACLE_MAINTAINED                                  VARCHAR2(1)
SQL> drop table cutovercheck;
Table dropped.
```

Nondisruptive redo log migration

There are times when a database is correctly organized overall with the exception of the redo logs. This can happen for many reasons, the most common of which is related to snapshots. Products such as SnapManager for Oracle, SnapCenter, and the NetApp Snap Creator storage management framework enable near-instantaneous recovery of a database, but only if you revert the state of the data file volumes. If redo logs share space with the data files, reversion cannot be performed safely because it would result in destruction of the redo logs, likely meaning data loss. Therefore, the redo logs must be relocated.

This procedure is simple and can be performed nondisruptively.

Current redo log configuration

1. Identify the number of redo log groups and their respective group numbers.

```
SQL> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
--------------------------------------------------------------------------------
--------
1 /redo0/NTAP/redo01a.log
1 /redo1/NTAP/redo01b.log
2 /redo0/NTAP/redo02a.log
2 /redo1/NTAP/redo02b.log
3 /redo0/NTAP/redo03a.log
3 /redo1/NTAP/redo03b.log
rows selected.
```

2. Enter the size of the redo logs.

```
SQL> select group#||' '||bytes from v$log;
GROUP#||''||BYTES
--------------------------------------------------------------------------------
--------
1 524288000
2 524288000
3 524288000
```

Create new logs

1. For each redo log, create a new group with a matching size and number of members.

```
SQL> alter database add logfile ('/newredo0/redo01a.log',
'/newredo1/redo01b.log') size 500M;
Database altered.
SQL> alter database add logfile ('/newredo0/redo02a.log',
'/newredo1/redo02b.log') size 500M;
Database altered.
SQL> alter database add logfile ('/newredo0/redo03a.log',
'/newredo1/redo03b.log') size 500M;
Database altered.
SQL>
```

2. Verify the new configuration.

```
SQL> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
--------------------------------------------------------------------------------
--------
1 /redo0/NTAP/redo01a.log
1 /redo1/NTAP/redo01b.log
2 /redo0/NTAP/redo02a.log
2 /redo1/NTAP/redo02b.log
3 /redo0/NTAP/redo03a.log
3 /redo1/NTAP/redo03b.log
4 /newredo0/redo01a.log
4 /newredo1/redo01b.log
5 /newredo0/redo02a.log
5 /newredo1/redo02b.log
6 /newredo0/redo03a.log
6 /newredo1/redo03b.log
12 rows selected.
```

Drop old logs

1. Drop the old logs (groups 1, 2, and 3).

```
SQL> alter database drop logfile group 1;
Database altered.
SQL> alter database drop logfile group 2;
Database altered.
SQL> alter database drop logfile group 3;
Database altered.
```

2. If you encounter an error that prevents you from dropping an active log, force a switch to the next log to release the lock and force a global checkpoint. See the following example of this process. The attempt to drop logfile group 2, which was located on the old location, was denied because there was still active data in this logfile.

```
SQL> alter database drop logfile group 2;
alter database drop logfile group 2
*
ERROR at line 1:
ORA-01623: log 2 is current log for instance NTAP (thread 1) - cannot
drop
ORA-00312: online log 2 thread 1: '/redo0/NTAP/redo02a.log'
ORA-00312: online log 2 thread 1: '/redo1/NTAP/redo02b.log'
```

3. A log archiving followed by a checkpoint enables you to drop the logfile.

```
SQL> alter system archive log current;
System altered.
SQL> alter system checkpoint;
System altered.
SQL> alter database drop logfile group 2;
Database altered.
```

4. Then delete the logs from the file system. You should perform this process with extreme care.

Oracle database host data copy

As with database-level migration, migration at the host layer provides a storage vendor–independent approach.

In other words, sometime "just copy the files" is the best option.

Although this low-tech approach might seem too basic, it does offer significant benefits because no special software is required and the original data remains safely untouched during the process. The primary limitation is the fact that a file-copy data migration is a disruptive process, because the database must be shut down before the copy operation begins. There is no good way to synchronize changes within a file, so the files must be completely quiesced before copying begins.

If the shutdown required by a copy operation is not desirable, the next best host-based option is leveraging a logical volume manager (LVM). Many LVM options exist, including Oracle ASM, all with similar capabilities, but also with some limitations that must be taken into account. In most cases, the migration can be accomplished without downtime and disruption.

Filesystem to filesystem copying

The usefulness of a simple copy operation should not be underestimated. This operation requires downtime during the copy process, but it is a highly reliable process and requires no special expertise with operating systems, databases, or storage systems. Furthermore, it is very safe because it does not affect the original data. Typically, a system administrator changes the source file systems to be mounted as read-only and then reboots a server to guarantee that nothing can damage the current data. The copy process can be scripted to make sure that it runs as quickly as possible without risk of user error. Because the type of I/O is a simple sequential transfer of data, it is highly bandwidth efficient.

The following example demonstrates one option for a safe and rapid migration.

Environment

The environment to be migrated is as follows:

- Current file systems

```
ontap-nfs1:/host1_oradata       52428800  16196928  36231872  31%
/oradata
ontap-nfs1:/host1_logs          49807360    548032  49259328   2% /logs
```

- New file systems

```
ontap-nfs1:/host1_logs_new      49807360       128  49807232   1%
/new/logs
ontap-nfs1:/host1_oradata_new   49807360       128  49807232   1%
/new/oradata
```

Overview

The database can be migrated by a DBA by simply shutting down the database and copying the files, but the process is easily scripted if many databases must be migrated or minimizing downtime is critical. The use of scripts also reduces the chance for user error.

The example scripts shown automate the following operations:

- Shutting down the database
- Converting the existing file systems to a read-only state
- Copying all data from the source to target file systems, which preserves all file permissions
- Unmounting the old and new file systems
- Remounting the new file systems at the same paths as the prior file systems

Procedure

1. Shut down the database.

```
[root@host1 current]# ./dbshut.pl NTAP
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 15:58:48 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP shut down
```

2. Convert the file systems to read-only. This can be done more quickly by using a script, as shown in Convert File System to Read Only.

```
[root@host1 current]# ./mk.fs.readonly.pl /oradata
/oradata unmounted
/oradata mounted read-only
[root@host1 current]# ./mk.fs.readonly.pl /logs
/logs unmounted
/logs mounted read-only
```

3. Confirm that the file systems are now read-only.

```
ontap-nfs1:/host1_oradata on /oradata type nfs
(ro,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
ontap-nfs1:/host1_logs on /logs type nfs
(ro,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
```

4. Synchronize file system contents with the `rsync` command.

```
[root@host1 current]# rsync -rlpogt --stats --progress
--exclude=.snapshot /oradata/ /new/oradata/
sending incremental file list
./
NTAP/
NTAP/IOPS.dbf
```

```
 10737426432 100%  153.50MB/s    0:01:06 (xfer#1, to-check=10/13)
NTAP/iops.dbf.zip
    22823573 100%   12.09MB/s    0:00:01 (xfer#2, to-check=9/13)
...
NTAP/undotbs02.dbf
  1073750016 100%  131.60MB/s    0:00:07 (xfer#10, to-check=1/13)
NTAP/users01.dbf
     5251072 100%    3.95MB/s    0:00:01 (xfer#11, to-check=0/13)
Number of files: 13
Number of files transferred: 11
Total file size: 18570092218 bytes
Total transferred file size: 18570092218 bytes
Literal data: 18570092218 bytes
Matched data: 0 bytes
File list size: 277
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 18572359828
Total bytes received: 228
sent 18572359828 bytes  received 228 bytes  162204017.96 bytes/sec
total size is 18570092218  speedup is 1.00
[root@host1 current]# rsync -rlpogt --stats --progress
--exclude=.snapshot /logs/ /new/logs/
sending incremental file list
./
NTAP/
NTAP/1_22_897068759.dbf
    45523968 100%   95.98MB/s    0:00:00 (xfer#1, to-check=15/18)
NTAP/1_23_897068759.dbf
    40601088 100%   49.45MB/s    0:00:00 (xfer#2, to-check=14/18)
...
NTAP/redo/redo02.log
    52429312 100%   44.68MB/s    0:00:01 (xfer#12, to-check=1/18)
NTAP/redo/redo03.log
    52429312 100%   68.03MB/s    0:00:00 (xfer#13, to-check=0/18)
Number of files: 18
Number of files transferred: 13
Total file size: 527032832 bytes
Total transferred file size: 527032832 bytes
Literal data: 527032832 bytes
Matched data: 0 bytes
File list size: 413
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 527098156
Total bytes received: 278
```

```
sent 527098156 bytes  received 278 bytes  95836078.91 bytes/sec
total size is 527032832  speedup is 1.00
```

5. Unmount the old file systems and relocate the copied data. This can be done more quickly by using a script, as shown in Replace File System.

```
[root@host1 current]# ./swap.fs.pl /logs,/new/logs
/new/logs unmounted
/logs unmounted
Updated /logs mounted
[root@host1 current]# ./swap.fs.pl /oradata,/new/oradata
/new/oradata unmounted
/oradata unmounted
Updated /oradata mounted
```

6. Confirm that the new file systems are in position.

```
ontap-nfs1:/host1_logs_new on /logs type nfs
(rw,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
ontap-nfs1:/host1_oradata_new on /oradata type nfs
(rw,bg,vers=3,rsize=65536,wsize=65536,addr=172.20.101.10)
```

7. Start the database.

```
[root@host1 current]# ./dbstart.pl NTAP
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 16:10:07 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             390073456 bytes
Database Buffers          406847488 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP started
```

Fully automated cutover

This sample script accepts arguments of the database SID followed by common-delimited pairs of file systems. For the example shown above, the command is issued as follows:

```
[root@host1 current]# ./migrate.oracle.fs.pl NTAP /logs,/new/logs
/oradata,/new/oradata
```

When executed, the example script attempts to perform the following sequence. It terminates if it encounters an error in any step:

1. Shut down the database.
2. Convert the current file systems to read-only status.
3. Use each comma-delimited pair of file system arguments and synchronize the first file system to the second.
4. Dismount the prior file systems.
5. Update the `/etc/fstab` file as follows:
   a. Create a backup at `/etc/fstab.bak`.
   b. Comment out the prior entries for the prior and new file systems.
   c. Create a new entry for the new file system that uses the old mountpoint.
6. Mount the file systems.
7. Start the database.

The following text provides an execution example for this script:

```
[root@host1 current]# ./migrate.oracle.fs.pl NTAP /logs,/new/logs
/oradata,/new/oradata
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 17:05:50 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP shut down
sending incremental file list
```

```
./
NTAP/
NTAP/1_22_897068759.dbf
    45523968 100%  185.40MB/s    0:00:00 (xfer#1, to-check=15/18)
NTAP/1_23_897068759.dbf
    40601088 100%   81.34MB/s    0:00:00 (xfer#2, to-check=14/18)
...
NTAP/redo/redo02.log
    52429312 100%   70.42MB/s    0:00:00 (xfer#12, to-check=1/18)
NTAP/redo/redo03.log
    52429312 100%   47.08MB/s    0:00:01 (xfer#13, to-check=0/18)
Number of files: 18
Number of files transferred: 13
Total file size: 527032832 bytes
Total transferred file size: 527032832 bytes
Literal data: 527032832 bytes
Matched data: 0 bytes
File list size: 413
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 527098156
Total bytes received: 278
sent 527098156 bytes  received 278 bytes  150599552.57 bytes/sec
total size is 527032832  speedup is 1.00
Succesfully replicated filesystem /logs to /new/logs
sending incremental file list
./
NTAP/
NTAP/IOPS.dbf
 10737426432 100%  176.55MB/s    0:00:58 (xfer#1, to-check=10/13)
NTAP/iops.dbf.zip
    22823573 100%    9.48MB/s    0:00:02 (xfer#2, to-check=9/13)
... NTAP/undotbs01.dbf
   309338112 100%   70.76MB/s    0:00:04 (xfer#9, to-check=2/13)
NTAP/undotbs02.dbf
  1073750016 100%  187.65MB/s    0:00:05 (xfer#10, to-check=1/13)
NTAP/users01.dbf
     5251072 100%    5.09MB/s    0:00:00 (xfer#11, to-check=0/13)
Number of files: 13
Number of files transferred: 11
Total file size: 18570092218 bytes
Total transferred file size: 18570092218 bytes
Literal data: 18570092218 bytes
Matched data: 0 bytes
File list size: 277
File list generation time: 0.001 seconds
```

```
File list transfer time: 0.000 seconds
Total bytes sent: 18572359828
Total bytes received: 228
sent 18572359828 bytes  received 228 bytes  177725933.55 bytes/sec
total size is 18570092218  speedup is 1.00
Succesfully replicated filesystem /oradata to /new/oradata
swap 0 /logs /new/logs
/new/logs unmounted
/logs unmounted
Mounted updated /logs
Swapped filesystem /logs for /new/logs
swap 1 /oradata /new/oradata
/new/oradata unmounted
/oradata unmounted
Mounted updated /oradata
Swapped filesystem /oradata for /new/oradata
ORACLE_SID = [oracle] ? The Oracle base has been set to /orabin
SQL*Plus: Release 12.1.0.2.0 Production on Thu Dec 3 17:08:59 2015
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> ORACLE instance started.
Total System Global Area  805306368 bytes
Fixed Size                  2929552 bytes
Variable Size             390073456 bytes
Database Buffers          406847488 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
NTAP started
[root@host1 current]#
```

Oracle ASM spfile and passwd migration

One difficulty in completing migration involving ASM is the ASM-specific spfile and the password file. By default, these critical metadata files are created on the first ASM disk group defined. If a particular ASM disk group must be evacuated and removed, the spfile and password file that govern that ASM instance must be relocated.

Another use case in which these files might need to be relocated is during a deployment of database management software such as SnapManager for Oracle or the SnapCenter Oracle plug- in. One of the features of these products is to rapidly restore a database by reverting the state of the ASM LUNs hosting the data files. Doing so requires taking the ASM disk group offline before performing a restore. This is not a problem as long as a given database's data files are isolated in a dedicated ASM disk group.

When that disk group also contains the ASM spfile/passwd file, the only way the disk group can be brought offline is to shut down the entire ASM instance. This is a disruptive process, which means that the spfile/passwd file would need to be relocated.

Environment

1. Database SID = TOAST
2. Current data files on `+DATA`
3. Current logfiles and controlfiles on `+LOGS`
4. New ASM disk groups established as `+NEWDATA` and `+NEWLOGS`

ASM spfile/passwd file locations

Relocating these files can be done nondisruptively. However, for safety, NetApp recommends shutting down the database environment so that you can be certain that the files have been relocated and the configuration is properly updated. This procedure must be repeated if multiple ASM instances are present on a server.

Identify ASM instances

Identify the ASM instances based on the data recorded in the `oratab` file. The ASM instances are denoted by a + symbol.

```
-bash-4.1$ cat /etc/oratab | grep '^+'
+ASM:/orabin/grid:N             # line added by Agent
```

There is one ASM instance called +ASM on this server.

Make sure all databases are shut down

The only smon process visible should be the smon for the ASM instance in use. The presence of another smon process indicates that a database is still running.

```
-bash-4.1$ ps -ef | grep smon
oracle     857     1  0 18:26 ?        00:00:00 asm_smon_+ASM
```

The only smon process is the ASM instance itself. This means that no other databases are running, and it is safe to proceed without risk of disrupting database operations.

Locate files

Identify the current location of the ASM spfile and password file by using the `spget` and `pwget` commands.

```
bash-4.1$ asmcmd
ASMCMD> spget
+DATA/spfile.ora
```

```
ASMCMD> pwget --asm
+DATA/orapwasm
```

The files are both located at the base of the `+DATA` disk group.

Copy files

Copy the files to the new ASM disk group with the `spcopy` and `pwcopy` commands. If the new disk group was recently created and is currently empty, it might need to be mounted first.

```
ASMCMD> mount NEWDATA
```

```
ASMCMD> spcopy +DATA/spfile.ora +NEWDATA/spfile.ora
copying +DATA/spfile.ora -> +NEWDATA/spfilea.ora
```

```
ASMCMD> pwcopy +DATA/orapwasm +NEWDATA/orapwasm
copying +DATA/orapwasm -> +NEWDATA/orapwasm
```

The files have now been copied from `+DATA` to `+NEWDATA`.

Update ASM instance

The ASM instance must now be updated to reflect the change in location. The `spset` and `pwset` commands update the ASM metadata required for starting the ASM disk group.

```
ASMCMD> spset +NEWDATA/spfile.ora
ASMCMD> pwset --asm +NEWDATA/orapwasm
```

Activate ASM using updated files

At this point, the ASM instance still uses the prior locations of these files. The instance must be restarted to force a reread of the files from their new locations and to release locks on the prior files.

```
-bash-4.1$ sqlplus / as sysasm
SQL> shutdown immediate;
ASM diskgroups volume disabled
ASM diskgroups dismounted
ASM instance shutdown
```

```
SQL> startup
ASM instance started
Total System Global Area 1140850688 bytes
Fixed Size                  2933400 bytes
Variable Size            1112751464 bytes
ASM Cache                  25165824 bytes
ORA-15032: not all alterations performed
ORA-15017: diskgroup "NEWDATA" cannot be mounted
ORA-15013: diskgroup "NEWDATA" is already mounted
```

Remove old spfile and password files

If the procedure has been performed successfully, the prior files are no longer locked and can now be removed.

```
-bash-4.1$ asmcmd
ASMCMD> rm +DATA/spfile.ora
ASMCMD> rm +DATA/orapwasm
```

Oracle ASM to ASM copy

Oracle ASM is essentially a lightweight combined volume manager and file system. Because the file system is not readily visible, RMAN must be used to perform copy operations. Although a copy-based migration process is safe and simple, it results in some disruption. The disruption can be minimized, but not fully eliminated.

If you want nondisruptive migration of an ASM-based database, the best option is to leverage ASM's capability to rebalance ASM extents to new LUNs while dropping the old LUNs. Doing so is generally safe and nondisruptive to operations, but it offers no back- out path. If functional or performance problems are encountered, the only option is to migrate the data back to the source.

This risk can be avoided by copying the database to the new location rather than moving data, so that the original data is untouched. The database can be fully tested in its new location before going live, and the original database is available as a fall- back option if problems are found.

This procedure is one of many options involving RMAN. It is designed to allow a two-step process in which the initial backup is created and then later synchronized through log replay. This process is desirable to minimize downtime because it allows the database to remain operational and serving data during the initial baseline copy.

Copy database

Oracle RMAN creates a level 0 (complete) copy of the source database currently located on the ASM disk group `+DATA` to the new location on `+NEWDATA`.

```
-bash-4.1$ rman target /
Recovery Manager: Release 12.1.0.2.0 - Production on Sun Dec 6 17:40:03
2015
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.
connected to target database: TOAST (DBID=2084313411)
RMAN> backup as copy incremental level 0 database format '+NEWDATA' tag
'ONTAP_MIGRATION';
Starting backup at 06-DEC-15
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=302 device type=DISK
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001
name=+DATA/TOAST/DATAFILE/system.262.897683141
...
input datafile file number=00004
name=+DATA/TOAST/DATAFILE/users.264.897683151
output file name=+NEWDATA/TOAST/DATAFILE/users.258.897759623
tag=ONTAP_MIGRATION RECID=5 STAMP=897759622
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 06-DEC-15
channel ORA_DISK_1: finished piece 1 at 06-DEC-15
piece
handle=+NEWDATA/TOAST/BACKUPSET/2015_12_06/nnsnn0_ontap_migration_0.262.89
7759623 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 06-DEC-15
```

Force archive log switch

You must force an archive log switch to make sure that the archive logs contain all data required to make the copy fully consistent. Without this command, key data might still be present in the redo logs.

```
RMAN> sql 'alter system archive log current';
sql statement: alter system archive log current
```

Shut down source database

Disruption begins in this step because the database is shut down and placed in a limited-access, read-only mode. To shut down the source database, run the following commands:

```
RMAN> shutdown immediate;
using target database control file instead of recovery catalog
database closed
database dismounted
Oracle instance shut down
RMAN> startup mount;
connected to target database (not started)
Oracle instance started
database mounted
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                390073456 bytes
Database Buffers             406847488 bytes
Redo Buffers                   5455872 bytes
```

Controlfile backup

You must back up the controlfile in case you must abort the migration and revert to the original storage location. A copy of the backup controlfile isn't 100% required, but it does make the process of resetting the database file locations back to the original location easier.

```
RMAN> backup as copy current controlfile format '/tmp/TOAST.ctrl';
Starting backup at 06-DEC-15
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=358 device type=DISK
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=/tmp/TOAST.ctrl tag=TAG20151206T174753 RECID=6
STAMP=897760073
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 06-DEC-15
```

Parameter updates

The current spfile contains references to the controlfiles on their current locations within the old ASM disk group. It must be edited, which is easily done by editing an intermediate pfile version.

```
RMAN> create pfile='/tmp/pfile' from spfile;
Statement processed
```

Update pfile

Update any parameters referring to old ASM disk groups to reflect the new ASM disk group names. Then save the updated pfile. Make sure that the `db_create` parameters are present.

In the example below, the references to `+DATA` that were changed to `+NEWDATA` are highlighted in yellow. Two key parameters are the `db_create` parameters that create any new files at the correct location.

```
*.compatible='12.1.0.2.0'
*.control_files='+NEWLOGS/TOAST/CONTROLFILE/current.258.897683139'
*.db_block_size=8192
*. db_create_file_dest='+NEWDATA'
*. db_create_online_log_dest_1='+NEWLOGS'
*.db_domain=''
*.db_name='TOAST'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=TOASTXDB)'
*.log_archive_dest_1='LOCATION=+NEWLOGS'
*.log_archive_format='%t_%s_%r.dbf'
```

Update init.ora file

Most ASM-based databases use an `init.ora` file located in the `$ORACLE_HOME/dbs` directory, which is a point to the spfile on the ASM disk group. This file must be redirected to a location on the new ASM disk group.

```
-bash-4.1$ cd $ORACLE_HOME/dbs
-bash-4.1$ cat initTOAST.ora
SPFILE='+DATA/TOAST/spfileTOAST.ora'
```

Change this file as follows:

```
SPFILE=+NEWLOGS/TOAST/spfileTOAST.ora
```

Parameter file recreation

The spfile is now ready to be populated by the data in the edited pfile.

```
RMAN> create spfile from pfile='/tmp/pfile';
Statement processed
```

Start database to start using new spfile

Start the database to make sure that it now uses the newly created spfile and that any further changes to system parameters are correctly recorded.

```
RMAN> startup nomount;
connected to target database (not started)
Oracle instance started
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                373296240 bytes
Database Buffers             423624704 bytes
Redo Buffers                   5455872 bytes
```

Restore controlfile

The backup controlfile created by RMAN can also be restored by RMAN directly to the location specified in the new spfile.

```
RMAN> restore controlfile from
'+DATA/TOAST/CONTROLFILE/current.258.897683139';
Starting restore at 06-DEC-15
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=417 device type=DISK
channel ORA_DISK_1: copied control file copy
output file name=+NEWLOGS/TOAST/CONTROLFILE/current.273.897761061
Finished restore at 06-DEC-15
```

Mount the database and verify the use of the new controlfile.

```
RMAN> alter database mount;
using target database control file instead of recovery catalog
Statement processed
```

```
SQL> show parameter control_files;
NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
control_files                        string
+NEWLOGS/TOAST/CONTROLFILE/cur
                                                 rent.273.897761061
```

Log replay

The database currently uses the data files in the old location. Before the copy can be used, they must be synchronized. Time has passed during the initial copy process, and the changes have been logged primarily in the archive logs. These changes are replicated as follows:

1. Perform an RMAN incremental backup, which contains the archive logs.

```
RMAN> backup incremental level 1 format '+NEWLOGS' for recover of copy
with tag 'ONTAP_MIGRATION' database;
Starting backup at 06-DEC-15
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=62 device type=DISK
channel ORA_DISK_1: starting incremental level 1 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=+DATA/TOAST/DATAFILE/system.262.897683141
input datafile file number=00002
name=+DATA/TOAST/DATAFILE/sysaux.260.897683143
input datafile file number=00003
name=+DATA/TOAST/DATAFILE/undotbs1.257.897683145
input datafile file number=00004
name=+DATA/TOAST/DATAFILE/users.264.897683151
channel ORA_DISK_1: starting piece 1 at 06-DEC-15
channel ORA_DISK_1: finished piece 1 at 06-DEC-15
piece
handle=+NEWLOGS/TOAST/BACKUPSET/2015_12_06/nnndn1_ontap_migration_0.268.
897762693 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental level 1 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 06-DEC-15
channel ORA_DISK_1: finished piece 1 at 06-DEC-15
piece
handle=+NEWLOGS/TOAST/BACKUPSET/2015_12_06/ncsnn1_ontap_migration_0.267.
897762697 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 06-DEC-15
```

2. Replay the log.

```
RMAN> recover copy of database with tag 'ONTAP_MIGRATION';
Starting recover at 06-DEC-15
using channel ORA_DISK_1
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: specifying datafile copies to recover
recovering datafile copy file number=00001
name=+NEWDATA/TOAST/DATAFILE/system.259.897759609
recovering datafile copy file number=00002
name=+NEWDATA/TOAST/DATAFILE/sysaux.263.897759615
recovering datafile copy file number=00003
name=+NEWDATA/TOAST/DATAFILE/undotbs1.264.897759619
recovering datafile copy file number=00004
name=+NEWDATA/TOAST/DATAFILE/users.258.897759623
channel ORA_DISK_1: reading from backup piece
+NEWLOGS/TOAST/BACKUPSET/2015_12_06/nnndn1_ontap_migration_0.268.8977626
93
channel ORA_DISK_1: piece
handle=+NEWLOGS/TOAST/BACKUPSET/2015_12_06/nnndn1_ontap_migration_0.268.
897762693 tag=ONTAP_MIGRATION
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished recover at 06-DEC-15
```

Activation

The controlfile that was restored still references the data files at the original location, and it also contains the path information for the copied data files.

1. To change the active data files, run the `switch database to copy` command.

```
RMAN> switch database to copy;
datafile 1 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/system.259.897759609"
datafile 2 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/sysaux.263.897759615"
datafile 3 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/undotbs1.264.897759619"
datafile 4 switched to datafile copy
"+NEWDATA/TOAST/DATAFILE/users.258.897759623"
```

   The active data files are now the copied data files, but there still might be changes contained within the final redo logs.

2. To replay all of the remaining logs, run the `recover database` command. If the message `media recovery complete` appears, the process was successful.

```
RMAN> recover database;
Starting recover at 06-DEC-15
using channel ORA_DISK_1
starting media recovery
media recovery complete, elapsed time: 00:00:01
Finished recover at 06-DEC-15
```

This process only changed the location of the normal data files. The temporary data files must be renamed, but they do not need to be copied because they are only temporary. The database is currently down, so there is no active data in the temporary data files.

3. To relocate the temporary data files, first identify their location.

```
RMAN> select file#||' '||name from v$tempfile;
FILE#||''||NAME
-------------------------------------------------------------------------------
--------
1 +DATA/TOAST/TEMPFILE/temp.263.897683145
```

4. Relocate temporary data files by using an RMAN command that sets the new name for each data file. With Oracle Managed Files (OMF), the complete name is not necessary; the ASM disk group is sufficient. When the database is opened, OMF links to the appropriate location on the ASM disk group. To relocate files, run the following commands:

```
run {
set newname for tempfile 1 to '+NEWDATA';
switch tempfile all;
}
```

```
RMAN> run {
2> set newname for tempfile 1 to '+NEWDATA';
3> switch tempfile all;
4> }
executing command: SET NEWNAME
renamed tempfile 1 to +NEWDATA in control file
```

Redo log migration

The migration process is nearly complete, but the redo logs are still located on the original ASM disk group. Redo logs cannot be directly relocated. Instead, a new set of redo logs is created and added to the configuration, followed by a drop of the old logs.

1. Identify the number of redo log groups and their respective group numbers.

```
RMAN> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
------------------------------------------------------------------------------
--------
1 +DATA/TOAST/ONLINELOG/group_1.261.897683139
2 +DATA/TOAST/ONLINELOG/group_2.259.897683139
3 +DATA/TOAST/ONLINELOG/group_3.256.897683139
```

2. Enter the size of the redo logs.

```
RMAN> select group#||' '||bytes from v$log;
GROUP#||''||BYTES
------------------------------------------------------------------------------
--------
1 52428800
2 52428800
3 52428800
```

3. For each redo log, create a new group with a matching configuration. If you are not using OMF, you must specify the full path. This is also an example that uses the `db_create_online_log` parameters. As was shown previously, this parameter was set to +NEWLOGS. This configuration allows you to use the following commands to create new online logs without the need to specify a file location or even a specific ASM disk group.

```
RMAN> alter database add logfile size 52428800;
Statement processed
RMAN> alter database add logfile size 52428800;
Statement processed
RMAN> alter database add logfile size 52428800;
Statement processed
```

4. Open the database.

```
SQL> alter database open;
Database altered.
```

5. Drop the old logs.

```
RMAN> alter database drop logfile group 1;
Statement processed
```

6. If you encounter an error that prevents you from dropping an active log, force a switch to the next log to

release the lock and force a global checkpoint. An example is shown below. The attempt to drop logfile group 3, which was located on the old location, was denied because there was still active data in this logfile. A log archiving following a checkpoint allows you to delete the logfile.

```
RMAN> alter database drop logfile group 3;
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of sql statement command at 12/08/2015 20:23:51
ORA-01623: log 3 is current log for instance TOAST (thread 4) - cannot
drop
ORA-00312: online log 3 thread 1:
'+LOGS/TOAST/ONLINELOG/group_3.259.897563549'
RMAN> alter system switch logfile;
Statement processed
RMAN> alter system checkpoint;
Statement processed
RMAN> alter database drop logfile group 3;
Statement processed
```

7. Review the environment to make sure that all location-based parameters are updated.

```
SQL> select name from v$datafile;
SQL> select member from v$logfile;
SQL> select name from v$tempfile;
SQL> show parameter spfile;
SQL> select name, value from v$parameter where value is not null;
```

8. The following script demonstrates how to simplify this process:

```
[root@host1 current]# ./checkdbdata.pl TOAST
TOAST datafiles:
+NEWDATA/TOAST/DATAFILE/system.259.897759609
+NEWDATA/TOAST/DATAFILE/sysaux.263.897759615
+NEWDATA/TOAST/DATAFILE/undotbs1.264.897759619
+NEWDATA/TOAST/DATAFILE/users.258.897759623
TOAST redo logs:
+NEWLOGS/TOAST/ONLINELOG/group_4.266.897763123
+NEWLOGS/TOAST/ONLINELOG/group_5.265.897763125
+NEWLOGS/TOAST/ONLINELOG/group_6.264.897763125
TOAST temp datafiles:
+NEWDATA/TOAST/TEMPFILE/temp.260.897763165
TOAST spfile
spfile                                string
+NEWDATA/spfiletoast.ora
TOAST key parameters
control_files +NEWLOGS/TOAST/CONTROLFILE/current.273.897761061
log_archive_dest_1 LOCATION=+NEWLOGS
db_create_file_dest +NEWDATA
db_create_online_log_dest_1 +NEWLOGS
```

9. If the ASM disk groups were completely evacuated, they can now be unmounted with `asmcmd`. However, in many cases the files belonging to other databases or the ASM spfile/passwd file might still be present.

```
-bash-4.1$ . oraenv
ORACLE_SID = [TOAST] ? +ASM
The Oracle base remains unchanged with value /orabin
-bash-4.1$ asmcmd
ASMCMD> umount DATA
ASMCMD>
```

Oracle ASM to file system copy

The Oracle ASM to file system copy procedure is very similar to the ASM to ASM copy procedure, with similar benefits and restrictions. The primary difference is the syntax of the various commands and configuration parameters when using a visible file system as opposed to an ASM disk group.

Copy database

Oracle RMAN is used to create a level 0 (complete) copy of the source database currently located on the ASM disk group `+DATA` to the new location on `/oradata`.

```
RMAN> backup as copy incremental level 0 database format
'/oradata/TOAST/%U' tag 'ONTAP_MIGRATION';
Starting backup at 13-MAY-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=377 device type=DISK
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001 name=+ASM0/TOAST/system01.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSTEM_FNO-
1_01r5fhjg tag=ONTAP_MIGRATION RECID=1 STAMP=911722099
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00002 name=+ASM0/TOAST/sysaux01.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSAUX_FNO-
2_02r5fhjo tag=ONTAP_MIGRATION RECID=2 STAMP=911722106
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003 name=+ASM0/TOAST/undotbs101.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-UNDOTBS1_FNO-
3_03r5fhjt tag=ONTAP_MIGRATION RECID=3 STAMP=911722113
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=/oradata/TOAST/cf_D-TOAST_id-2098173325_04r5fhk5
tag=ONTAP_MIGRATION RECID=4 STAMP=911722118
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00004 name=+ASM0/TOAST/users01.dbf
output file name=/oradata/TOAST/data_D-TOAST_I-2098173325_TS-USERS_FNO-
4_05r5fhk6 tag=ONTAP_MIGRATION RECID=5 STAMP=911722118
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 13-MAY-16
channel ORA_DISK_1: finished piece 1 at 13-MAY-16
piece handle=/oradata/TOAST/06r5fhk7_1_1 tag=ONTAP_MIGRATION comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 13-MAY-16
```

Force archive log switch

Forcing the archive log switch is required to make sure that the archive logs contain all of the data required to make the copy fully consistent. Without this command, key data might still be present in the redo logs. To force an archive log switch, run the following command:

```
RMAN> sql 'alter system archive log current';
sql statement: alter system archive log current
```

Shut down source database

Disruption begins in this step because the database is shut down and placed in a limited-access read-only mode. To shut down the source database, run the following commands:

```
RMAN> shutdown immediate;
using target database control file instead of recovery catalog
database closed
database dismounted
Oracle instance shut down
RMAN> startup mount;
connected to target database (not started)
Oracle instance started
database mounted
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                331353200 bytes
Database Buffers             465567744 bytes
Redo Buffers                   5455872 bytes
```

Controlfile backup

Back up controlfiles in case you must abort the migration and revert to the original storage location. A copy of the backup controlfile isn't 100% required, but it does make the process of resetting the database file locations back to the original location easier.

```
RMAN> backup as copy current controlfile format '/tmp/TOAST.ctrl';
Starting backup at 08-DEC-15
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=/tmp/TOAST.ctrl tag=TAG20151208T194540 RECID=30
STAMP=897939940
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 08-DEC-15
```

Parameter updates

```
RMAN> create pfile='/tmp/pfile' from spfile;
Statement processed
```

Update pfile

Any parameters referring to old ASM disk groups should be updated and, in some cases, deleted when they are no longer relevant. Update them to reflect the new file system paths and save the updated pfile. Make sure that the complete target path is listed. To update these parameters, run the following commands:

```
*.audit_file_dest='/orabin/admin/TOAST/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='/logs/TOAST/arch/control01.ctl','/logs/TOAST/redo/control
02.ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='TOAST'
*.diagnostic_dest='/orabin'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=TOASTXDB)'
*.log_archive_dest_1='LOCATION=/logs/TOAST/arch'
*.log_archive_format='%t_%s_%r.dbf'
*.open_cursors=300
*.pga_aggregate_target=256m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=768m
*.undo_tablespace='UNDOTBS1'
```

Disable the original init.ora file

This file is located in the `$ORACLE_HOME/dbs` directory and is usually in a pfile that serves as a pointer to the spfile on the ASM disk group. To make sure that the original spfile is no longer used, rename it. Do not delete it, however, because this file is needed if the migration must be aborted.

```
[oracle@jfsc1 ~]$ cd $ORACLE_HOME/dbs
[oracle@jfsc1 dbs]$ cat initTOAST.ora
SPFILE='+ASM0/TOAST/spfileTOAST.ora'
[oracle@jfsc1 dbs]$ mv initTOAST.ora initTOAST.ora.prev
[oracle@jfsc1 dbs]$
```

Parameter file recreation

This is the final step in spfile relocation. The original spfile is no longer used and the database is currently started (but not mounted) using the intermediate file. The contents of this file can be written out to the new spfile location as follows:

```
RMAN> create spfile from pfile='/tmp/pfile';
Statement processed
```

Start database to start using new spfile

You must start the database to release the locks on the intermediate file and start the database by using only the new spfile file. Starting the database also proves that the new spfile location is correct and its data is valid.

```
RMAN> shutdown immediate;
Oracle instance shut down
RMAN> startup nomount;
connected to target database (not started)
Oracle instance started
Total System Global Area     805306368 bytes
Fixed Size                     2929552 bytes
Variable Size                331353200 bytes
Database Buffers             465567744 bytes
Redo Buffers                   5455872 bytes
```

Restore controlfile

A backup controlfile was created at the path `/tmp/TOAST.ctrl` earlier in the procedure. The new spfile defines the controlfile locations as `/logfs/TOAST/ctrl/ctrlfile1.ctrl` and `/logfs/TOAST/redo/ctrlfile2.ctrl`. However, those files do not yet exist.

1. This command restores the controlfile data to the paths defined in the spfile.

   ```
   RMAN> restore controlfile from '/tmp/TOAST.ctrl';
   Starting restore at 13-MAY-16
   using channel ORA_DISK_1
   channel ORA_DISK_1: copied control file copy
   output file name=/logs/TOAST/arch/control01.ctl
   output file name=/logs/TOAST/redo/control02.ctl
   Finished restore at 13-MAY-16
   ```

2. Issue the mount command so that the controlfiles are discovered correctly and contain valid data.

   ```
   RMAN> alter database mount;
   Statement processed
   released channel: ORA_DISK_1
   ```

   To validate the `control_files` parameter, run the following command:

```
SQL> show parameter control_files;
NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
control_files                        string
/logs/TOAST/arch/control01.ctl
                                                 ,
/logs/TOAST/redo/control02.c
                                                 tl
```

Log replay

The database is currently using the data files in the old location. Before the copy can be used, the data files must be synchronized. Time has passed during the initial copy process, and changes were logged primarily in the archive logs. These changes are replicated in the following two steps.

1. Perform an RMAN incremental backup, which contains the archive logs.

```
RMAN>  backup incremental level 1 format '/logs/TOAST/arch/%U' for
recover of copy with tag 'ONTAP_MIGRATION' database;
Starting backup at 13-MAY-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=124 device type=DISK
channel ORA_DISK_1: starting incremental level 1 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=+ASM0/TOAST/system01.dbf
input datafile file number=00002 name=+ASM0/TOAST/sysaux01.dbf
input datafile file number=00003 name=+ASM0/TOAST/undotbs101.dbf
input datafile file number=00004 name=+ASM0/TOAST/users01.dbf
channel ORA_DISK_1: starting piece 1 at 13-MAY-16
channel ORA_DISK_1: finished piece 1 at 13-MAY-16
piece handle=/logs/TOAST/arch/09r5fj8i_1_1 tag=ONTAP_MIGRATION
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 13-MAY-16
RMAN-06497: WARNING: control file is not current, control file
AUTOBACKUP skipped
```

2. Replay the logs.

```
RMAN> recover copy of database with tag 'ONTAP_MIGRATION';
Starting recover at 13-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: specifying datafile copies to recover
recovering datafile copy file number=00001 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-SYSTEM_FNO-1_01r5fhjg
recovering datafile copy file number=00002 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-SYSAUX_FNO-2_02r5fhjo
recovering datafile copy file number=00003 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-UNDOTBS1_FNO-3_03r5fhjt
recovering datafile copy file number=00004 name=/oradata/TOAST/data_D-
TOAST_I-2098173325_TS-USERS_FNO-4_05r5fhk6
channel ORA_DISK_1: reading from backup piece
/logs/TOAST/arch/09r5fj8i_1_1
channel ORA_DISK_1: piece handle=/logs/TOAST/arch/09r5fj8i_1_1
tag=ONTAP_MIGRATION
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished recover at 13-MAY-16
RMAN-06497: WARNING: control file is not current, control file
AUTOBACKUP skipped
```

Activation

The controlfile that was restored still references the data files at the original location, and it also contains the path information for the copied data files.

1. To change the active data files, run the `switch database to copy` command:

```
RMAN> switch database to copy;
datafile 1 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-SYSTEM_FNO-1_01r5fhjg"
datafile 2 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-SYSAUX_FNO-2_02r5fhjo"
datafile 3 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-UNDOTBS1_FNO-3_03r5fhjt"
datafile 4 switched to datafile copy "/oradata/TOAST/data_D-TOAST_I-
2098173325_TS-USERS_FNO-4_05r5fhk6"
```

2. Although the data files should be fully consistent, one final step is required to replay the remaining changes recorded in the online redo logs. Use the `recover database` command to replay these changes and make the copy 100% identical to the original. The copy is not yet open, however.

```
RMAN> recover database;
Starting recover at 13-MAY-16
using channel ORA_DISK_1
starting media recovery
archived log for thread 1 with sequence 28 is already on disk as file
+ASM0/TOAST/redo01.log
archived log file name=+ASM0/TOAST/redo01.log thread=1 sequence=28
media recovery complete, elapsed time: 00:00:00
Finished recover at 13-MAY-16
```

Relocate Temporary Data Files

1. Identify the location of temporary data files still in use on the original disk group.

```
RMAN> select file#||' '||name from v$tempfile;
FILE#||''||NAME
------------------------------------------------------------------------
--------
1 +ASM0/TOAST/temp01.dbf
```

2. To relocate the data files, run the following commands. If there are many tempfiles, use a text editor to create the RMAN command and then cut and paste it.

```
RMAN> run {
2> set newname for tempfile 1 to '/oradata/TOAST/temp01.dbf';
3> switch tempfile all;
4> }
executing command: SET NEWNAME
renamed tempfile 1 to /oradata/TOAST/temp01.dbf in control file
```

Redo log migration

The migration process is nearly complete, but the redo logs are still located on the original ASM disk group. Redo logs cannot be directly relocated. Instead, a new set of redo logs is created and added to the configuration, following by a drop of the old logs.

1. Identify the number of redo log groups and their respective group numbers.

```
RMAN> select group#||' '||member from v$logfile;
GROUP#||''||MEMBER
--------------------------------------------------------------------------------
--------
1 +ASM0/TOAST/redo01.log
2 +ASM0/TOAST/redo02.log
3 +ASM0/TOAST/redo03.log
```

2. Enter the size of the redo logs.

```
RMAN> select group#||' '||bytes from v$log;
GROUP#||''||BYTES
--------------------------------------------------------------------------------
--------
1 52428800
2 52428800
3 52428800
```

3. For each redo log, create a new group by using the same size as the current redo log group using the new file system location.

```
RMAN> alter database add logfile '/logs/TOAST/redo/log00.rdo' size
52428800;
Statement processed
RMAN> alter database add logfile '/logs/TOAST/redo/log01.rdo' size
52428800;
Statement processed
RMAN> alter database add logfile '/logs/TOAST/redo/log02.rdo' size
52428800;
Statement processed
```

4. Remove the old logfile groups that are still located on the prior storage.

```
RMAN> alter database drop logfile group 4;
Statement processed
RMAN> alter database drop logfile group 5;
Statement processed
RMAN> alter database drop logfile group 6;
Statement processed
```

5. If an error is encountered that blocks dropping an active log, force a switch to the next log to release the lock and force a global checkpoint. An example is shown below. The attempt to drop logfile group 3, which was located on the old location, was denied because there was still active data in this logfile. A log

archiving followed by a checkpoint enables logfile deletion.

```
RMAN> alter database drop logfile group 4;
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of sql statement command at 12/08/2015 20:23:51
ORA-01623: log 4 is current log for instance TOAST (thread 4) - cannot
drop
ORA-00312: online log 4 thread 1:
'+NEWLOGS/TOAST/ONLINELOG/group_4.266.897763123'
RMAN> alter system switch logfile;
Statement processed
RMAN> alter system checkpoint;
Statement processed
RMAN> alter database drop logfile group 4;
Statement processed
```

6. Review the environment to make sure that all location-based parameters are updated.

```
SQL> select name from v$datafile;
SQL> select member from v$logfile;
SQL> select name from v$tempfile;
SQL> show parameter spfile;
SQL> select name, value from v$parameter where value is not null;
```

7. The following script demonstrates how to make this process easier.

```
[root@jfsc1 current]# ./checkdbdata.pl TOAST
TOAST datafiles:
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSTEM_FNO-1_01r5fhjg
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSAUX_FNO-2_02r5fhjo
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-UNDOTBS1_FNO-3_03r5fhjt
/oradata/TOAST/data_D-TOAST_I-2098173325_TS-USERS_FNO-4_05r5fhk6
TOAST redo logs:
/logs/TOAST/redo/log00.rdo
/logs/TOAST/redo/log01.rdo
/logs/TOAST/redo/log02.rdo
TOAST temp datafiles:
/oradata/TOAST/temp01.dbf
TOAST spfile
spfile                                  string
/orabin/product/12.1.0/dbhome_
                                                      1/dbs/spfileTOAST.ora
TOAST key parameters
control_files /logs/TOAST/arch/control01.ctl,
/logs/TOAST/redo/control02.ctl
log_archive_dest_1 LOCATION=/logs/TOAST/arch
```

8. If the ASM disk groups were completely evacuated, they can now be unmounted with `asmcmd`. In many cases, files belonging to other databases or the ASM spfile/passwd file can still be present.

```
-bash-4.1$ . oraenv
ORACLE_SID = [TOAST] ? +ASM
The Oracle base remains unchanged with value /orabin
-bash-4.1$ asmcmd
ASMCMD> umount DATA
ASMCMD>
```

Data file cleanup procedure

The migration process might result in data files with long or cryptic syntax, depending on how Oracle RMAN was used. In the example shown here, the backup was performed with the file format of `/oradata/TOAST/%U`. `%U` indicates that RMAN should create a default unique name for each data file. The result is similar to what is shown in the following text. The traditional names for the data files are embedded within the names. This can be cleaned up by using the scripted approach shown in ASM Migration Cleanup.

```
[root@jfsc1 current]# ./fixuniquenames.pl TOAST
#sqlplus Commands
shutdown immediate;
startup mount;
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSTEM_FNO-1_01r5fhjg
/oradata/TOAST/system.dbf
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-SYSAUX_FNO-2_02r5fhjo
/oradata/TOAST/sysaux.dbf
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-UNDOTBS1_FNO-
3_03r5fhjt /oradata/TOAST/undotbs1.dbf
host mv /oradata/TOAST/data_D-TOAST_I-2098173325_TS-USERS_FNO-4_05r5fhk6
/oradata/TOAST/users.dbf
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
SYSTEM_FNO-1_01r5fhjg' to '/oradata/TOAST/system.dbf';
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
SYSAUX_FNO-2_02r5fhjo' to '/oradata/TOAST/sysaux.dbf';
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
UNDOTBS1_FNO-3_03r5fhjt' to '/oradata/TOAST/undotbs1.dbf';
alter database rename file '/oradata/TOAST/data_D-TOAST_I-2098173325_TS-
USERS_FNO-4_05r5fhk6' to '/oradata/TOAST/users.dbf';
alter database open;
```

Oracle ASM rebalance

As discussed previously, an Oracle ASM disk group can be transparently migrated to a new storage system by using the rebalancing process. In summary, the rebalancing process requires the addition of equal-sized LUNs to the existing group of LUNs followed by a drop operation of the prior LUN. Oracle ASM automatically relocates the underlying data to new storage in an optimal layout and then releases the old LUNs when complete.

The migration process uses efficient sequential I/O and does not generally cause any performance disruption, but the migration rate can be throttled when needed.

Identify data to be migrated

```
SQL> select name||' '||group_number||' '||total_mb||' '||path||'
'||header_status from v$asm_disk;
NEWDATA_0003 1 10240 /dev/mapper/3600a098038303537762b47594c315864 MEMBER
NEWDATA_0002 1 10240 /dev/mapper/3600a098038303537762b47594c315863 MEMBER
NEWDATA_0000 1 10240 /dev/mapper/3600a098038303537762b47594c315861 MEMBER
NEWDATA_0001 1 10240 /dev/mapper/3600a098038303537762b47594c315862 MEMBER
SQL> select group_number||' '||name from v$asm_diskgroup;
1 NEWDATA
```

Create new LUNs

Create new LUNs of the same size, and set user and group membership as required. The LUNs should appear as `CANDIDATE` disks.

```
SQL> select name||' '||group_number||' '||total_mb||' '||path||'
'||header_status from v$asm_disk;
 0 0 /dev/mapper/3600a098038303537762b47594c31586b CANDIDATE
 0 0 /dev/mapper/3600a098038303537762b47594c315869 CANDIDATE
 0 0 /dev/mapper/3600a098038303537762b47594c315858 CANDIDATE
 0 0 /dev/mapper/3600a098038303537762b47594c31586a CANDIDATE
NEWDATA_0003 1 10240 /dev/mapper/3600a098038303537762b47594c315864 MEMBER
NEWDATA_0002 1 10240 /dev/mapper/3600a098038303537762b47594c315863 MEMBER
NEWDATA_0000 1 10240 /dev/mapper/3600a098038303537762b47594c315861 MEMBER
NEWDATA_0001 1 10240 /dev/mapper/3600a098038303537762b47594c315862 MEMBER
```

Add new LUNS

While the add and drop operations can be performed together, it is generally easier to add new LUNs in two steps. First, add the new LUNs to the disk group. This step results in half of the extents being migrated from the current ASM LUNs to the new LUNs.

The rebalance power indicates the rate at which data is being transferred. The higher the number, the higher the parallelism of the data transfer. The migration is performed with efficient sequential I/O operations that are unlikely to cause performance problems. However, if desired, the rebalance power of an ongoing migration can be adjusted with the `alter diskgroup [name] rebalance power [level]` command. Typical migrations use a value of 5.

```
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c31586b' rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c315869' rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c315858' rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA add disk
'/dev/mapper/3600a098038303537762b47594c31586a' rebalance power 5;
Diskgroup altered.
```

Monitor operation

A rebalancing operation can be monitored and managed in multiple ways. We used the following command for this example.

```
SQL> select group_number,operation,state from v$asm_operation;
GROUP_NUMBER OPERA STAT
------------ ----- ----
           1 REBAL RUN
           1 REBAL WAIT
```

When migration is complete, no rebalancing operations are reported.

```
SQL> select group_number,operation,state from v$asm_operation;
no rows selected
```

Drop old LUNs

The migration is now halfway complete. It might be desirable to perform some basic performance tests to make sure that the environment is healthy. After confirmation, the remaining data can be relocated by dropping the old LUNs. Note that this does not result in immediate release of the LUNs. The drop operation signals Oracle ASM to relocate the extents first and then release the LUN.

```
sqlplus / as sysasm
SQL> alter diskgroup NEWDATA drop disk NEWDATA_0000 rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup NEWDATA drop disk NEWDATA_0001 rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup newdata drop disk NEWDATA_0002 rebalance power 5;
Diskgroup altered.
SQL> alter diskgroup newdata drop disk NEWDATA_0003 rebalance power 5;
Diskgroup altered.
```

Monitor operation

The rebalancing operation can be monitored and managed in multiple ways. We used the following command for this example:

```
SQL> select group_number,operation,state from v$asm_operation;
GROUP_NUMBER OPERA STAT
------------ ----- ----
           1 REBAL RUN
           1 REBAL WAIT
```

When migration is complete, no rebalancing operations are reported.

```
SQL> select group_number,operation,state from v$asm_operation;
no rows selected
```

Remove old LUNs

Before you remove the old LUNs from the disk group, you should perform one final check on the header status. After a LUN is released from ASM, it no longer has a name listed and the header status is listed as `FORMER`. This indicates that these LUNs can safely be removed from the system.

```
SQL> select name||' '||group_number||' '||total_mb||' '||path||'
'||header_status from v$asm_disk;
NAME||''||GROUP_NUMBER||''||TOTAL_MB||''||PATH||''||HEADER_STATUS
--------------------------------------------------------------------------
------
 0 0 /dev/mapper/3600a098038303537762b47594c315863 FORMER
 0 0 /dev/mapper/3600a098038303537762b47594c315864 FORMER
 0 0 /dev/mapper/3600a098038303537762b47594c315861 FORMER
 0 0 /dev/mapper/3600a098038303537762b47594c315862 FORMER
NEWDATA_0005 1 10240 /dev/mapper/3600a098038303537762b47594c315869 MEMBER
NEWDATA_0007 1 10240 /dev/mapper/3600a098038303537762b47594c31586a MEMBER
NEWDATA_0004 1 10240 /dev/mapper/3600a098038303537762b47594c31586b MEMBER
NEWDATA_0006 1 10240 /dev/mapper/3600a098038303537762b47594c315858 MEMBER
8 rows selected.
```

LVM migration

The procedure presented here shows the principles of an LVM-based migration of a volume group called `datavg`. The examples are drawn from the Linux LVM, but the principles apply equally to AIX, HP-UX, and VxVM. The precise commands might vary.

1. Identify the LUNs currently in the `datavg` volume group.

```
[root@host1 ~]# pvdisplay -C | grep datavg
  /dev/mapper/3600a098038303537762b47594c31582f datavg lvm2 a--  10.00g
10.00g
  /dev/mapper/3600a098038303537762b47594c31585a datavg lvm2 a--  10.00g
10.00g
  /dev/mapper/3600a098038303537762b47594c315859 datavg lvm2 a--  10.00g
10.00g
  /dev/mapper/3600a098038303537762b47594c31586c datavg lvm2 a--  10.00g
10.00g
```

2. Create new LUNs of the same or slightly larger physical size and define them as physical volumes.

```
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315864
  Physical volume "/dev/mapper/3600a098038303537762b47594c315864"
successfully created
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315863
  Physical volume "/dev/mapper/3600a098038303537762b47594c315863"
successfully created
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315862
  Physical volume "/dev/mapper/3600a098038303537762b47594c315862"
successfully created
[root@host1 ~]# pvcreate /dev/mapper/3600a098038303537762b47594c315861
  Physical volume "/dev/mapper/3600a098038303537762b47594c315861"
successfully created
```

3. Add the new volumes to the volume group.

```
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315864
  Volume group "datavg" successfully extended
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315863
  Volume group "datavg" successfully extended
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315862
  Volume group "datavg" successfully extended
[root@host1 tmp]# vgextend datavg
/dev/mapper/3600a098038303537762b47594c315861
  Volume group "datavg" successfully extended
```

4. Issue the `pvmove` command to relocate the extents of each current LUN to the new LUN. The `-i [seconds]` argument monitors the progress of the operation.

```
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c31582f
/dev/mapper/3600a098038303537762b47594c315864
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 14.2%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 28.4%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 42.5%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 57.1%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 72.3%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 87.3%
  /dev/mapper/3600a098038303537762b47594c31582f: Moved: 100.0%
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c31585a
/dev/mapper/3600a098038303537762b47594c315863
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 14.9%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 29.9%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 44.8%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 60.1%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 75.8%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 90.9%
  /dev/mapper/3600a098038303537762b47594c31585a: Moved: 100.0%
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c315859
/dev/mapper/3600a098038303537762b47594c315862
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 14.8%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 29.8%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 45.5%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 61.1%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 76.6%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 91.7%
  /dev/mapper/3600a098038303537762b47594c315859: Moved: 100.0%
[root@host1 tmp]# pvmove -i 10
/dev/mapper/3600a098038303537762b47594c31586c
/dev/mapper/3600a098038303537762b47594c315861
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 0.0%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 15.0%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 30.4%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 46.0%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 61.4%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 77.2%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 92.3%
  /dev/mapper/3600a098038303537762b47594c31586c: Moved: 100.0%
```

5. When this process is complete, drop the old LUNs from the volume group by using the `vgreduce` command. If successful, the LUN can now be safely removed from the system.

```
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c31582f
Removed "/dev/mapper/3600a098038303537762b47594c31582f" from volume
group "datavg"
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c31585a
  Removed "/dev/mapper/3600a098038303537762b47594c31585a" from volume
group "datavg"
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c315859
  Removed "/dev/mapper/3600a098038303537762b47594c315859" from volume
group "datavg"
[root@host1 tmp]# vgreduce datavg
/dev/mapper/3600a098038303537762b47594c31586c
  Removed "/dev/mapper/3600a098038303537762b47594c31586c" from volume
group "datavg"
```

Foreign LUN import

Oracle migration with FLI - planning

The procedures to migrate SAN resources using FLI are documented in NetApp TR-4380: SAN Migration Using Foreign LUN Import.

From a database and host point of view, no special steps are required. After the FC zones are updated and the LUNs become available on ONTAP, the LVM should be able to read the LVM metadata from the LUNs. Also, the volume groups are ready for use with no further configuration steps. In rare cases, environments might include configuration files that were hard-coded with references to the prior storage array. For example, a Linux system that included `/etc/multipath.conf` rules that referenced a WWN of a given device must be updated to reflect the changes introduced by FLI.

Reference the NetApp Compatibility Matrix for information on supported configurations. If your environment is not included, contact your NetApp representative for assistance.

This example shows the migration of both ASM and LVM LUNs hosted on a Linux server. FLI is supported on other operating systems, and, although the host-side commands might differ, the principles are the same, and the ONTAP procedures are identical.

Identify LVM LUNs

The first step in preparation is to identify the LUNs to be migrated. In the example shown here, two SAN-based file systems are mounted at `/orabin` and `/backups`.

```
[root@host1 ~]# df -k
Filesystem                          1K-blocks      Used Available Use%
Mounted on
/dev/mapper/rhel-root                52403200   8811464  43591736  17% /
devtmpfs                             65882776         0  65882776   0% /dev
...
fas8060-nfs-public:/install         199229440 119368128  79861312  60%
/install
/dev/mapper/sanvg-lvorabin           20961280  12348476   8612804  59%
/orabin
/dev/mapper/sanvg-lvbackups          73364480  62947536  10416944  86%
/backups
```

The name of the volume group can be extracted from the device name, which uses the format (volume group name)-(logical volume name). In this case, the volume group is called `sanvg`.

The `pvdisplay` command can be used as follows to identify the LUNs that support this volume group. In this case, there are 10 LUNs that make up the `sanvg` volume group.

```
[root@host1 ~]# pvdisplay -C -o pv_name,pv_size,pv_fmt,vg_name
  PV                                                PSize   VG
  /dev/mapper/3600a0980383030445424487556574266  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574267  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574268  10.00g sanvg
  /dev/mapper/3600a0980383030445424487556574269  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426a  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426b  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426c  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426d  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426e  10.00g sanvg
  /dev/mapper/3600a098038303044542448755657426f  10.00g sanvg
  /dev/sda2                                        278.38g rhel
```

Identify ASM LUNs

ASM LUNs must also be migrated. To obtain the number of LUNs and LUN paths from sqlplus as the sysasm user, run the following command:

```
SQL> select path||' '||os_mb from v$asm_disk;
PATH||''||OS_MB
--------------------------------------------------------------------------------
------
/dev/oracleasm/disks/ASM0 10240
/dev/oracleasm/disks/ASM9 10240
/dev/oracleasm/disks/ASM8 10240
/dev/oracleasm/disks/ASM7 10240
/dev/oracleasm/disks/ASM6 10240
/dev/oracleasm/disks/ASM5 10240
/dev/oracleasm/disks/ASM4 10240
/dev/oracleasm/disks/ASM1 10240
/dev/oracleasm/disks/ASM3 10240
/dev/oracleasm/disks/ASM2 10240
10 rows selected.
SQL>
```

FC network changes

The current environment contains 20 LUNs to be migrated. Update the current SAN so that ONTAP can access the current LUNs. Data is not migrated yet, but ONTAP must read configuration information from the current LUNs to create the new home for that data.

At a minimum, at least one HBA port on the AFF/FAS system must be configured as an initiator port. In addition, the FC zones must be updated so that ONTAP can access the LUNs on the foreign storage array. Some storage arrays have LUN masking configured, which limits which WWNs can access a given LUN. In such cases, LUN masking must also be updated to grant access to the ONTAP WWNs.

After this step is completed, ONTAP should be able to view the foreign storage array with the `storage array show` command. The key field it returns is the prefix that is used to identify the foreign LUN on the system. In the example below, the LUNs on the foreign array `FOREIGN_1` appear within ONTAP using the prefix of `FOR-1`.

Identify foreign array

```
Cluster01::> storage array show -fields name,prefix
name          prefix
------------- ------
FOREIGN_1     FOR-1
Cluster01::>
```

Identify foreign LUNs

The LUNs can be listed by passing the `array-name` to the `storage disk show` command. The data returned is referenced multiple times during the migration procedure.

```
Cluster01::> storage disk show -array-name FOREIGN_1 -fields disk,serial
disk     serial-number
-------- -------------
FOR-1.1  800DT$HuVWBX
FOR-1.2  800DT$HuVWBZ
FOR-1.3  800DT$HuVWBW
FOR-1.4  800DT$HuVWBY
FOR-1.5  800DT$HuVWB/
FOR-1.6  800DT$HuVWBa
FOR-1.7  800DT$HuVWBd
FOR-1.8  800DT$HuVWBb
FOR-1.9  800DT$HuVWBc
FOR-1.10 800DT$HuVWBe
FOR-1.11 800DT$HuVWBf
FOR-1.12 800DT$HuVWBg
FOR-1.13 800DT$HuVWBi
FOR-1.14 800DT$HuVWBh
FOR-1.15 800DT$HuVWBj
FOR-1.16 800DT$HuVWBk
FOR-1.17 800DT$HuVWBm
FOR-1.18 800DT$HuVWBl
FOR-1.19 800DT$HuVWBo
FOR-1.20 800DT$HuVWBn
20 entries were displayed.
Cluster01::>
```

Register foreign array LUNs as import candidates

The foreign LUNs are initially classified as any particular LUN type. Before data can be imported, the LUNs must be tagged as foreign and therefore a candidate for the import process. This step is completed by passing the serial number to the `storage disk modify` command, as shown in the following example. Note that this process tags only the LUN as foreign within ONTAP. No data is written to the foreign LUN itself.

```
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBW} -is
-foreign true
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBX} -is
-foreign true
...
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBn} -is
-foreign true
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBo} -is
-foreign true
Cluster01::*>
```

Create volumes to host migrated LUNs

A volume is needed to host the migrated LUNs. The exact volume configuration depends on the overall plan to leverage ONTAP features. In this example, the ASM LUNs are placed into one volume and the LVM LUNs are placed in a second volume. Doing so allows you to manage the LUNs as independent groups for purposes such as tiering, creation of snapshots, or setting QoS controls.

Set the `snapshot-policy `to `none`. The migration process can include a great deal of data turnover. Therefore, there might be a large increase in space consumption if snapshots are created by accident because unwanted data is captured in the snapshots.

```
Cluster01::> volume create -volume new_asm -aggregate data_02 -size 120G
-snapshot-policy none
[Job 1152] Job succeeded: Successful
Cluster01::> volume create -volume new_lvm -aggregate data_02 -size 120G
-snapshot-policy none
[Job 1153] Job succeeded: Successful
Cluster01::>
```

Create ONTAP LUNs

After the volumes are created, the new LUNs must be created. Normally, the creation of a LUN requires the user to specify such information as the LUN size, but in this case the foreign-disk argument is passed to the command. As a result, ONTAP replicates the current LUN configuration data from the specified serial number. It also uses the LUN geometry and partition table data to adjust LUN alignment and establish optimum performance.

In this step, serial numbers must be cross-referenced against the foreign array to make sure that the correct foreign LUN is matched to the correct new LUN.

```
Cluster01::*> lun create -vserver vserver1 -path /vol/new_asm/LUN0 -ostype
linux -foreign-disk 800DT$HuVWBW
Created a LUN of size 10g (10737418240)
Cluster01::*> lun create -vserver vserver1 -path /vol/new_asm/LUN1 -ostype
linux -foreign-disk 800DT$HuVWBX
Created a LUN of size 10g (10737418240)
...
Created a LUN of size 10g (10737418240)
Cluster01::*> lun create -vserver vserver1 -path /vol/new_lvm/LUN8 -ostype
linux -foreign-disk 800DT$HuVWBn
Created a LUN of size 10g (10737418240)
Cluster01::*> lun create -vserver vserver1 -path /vol/new_lvm/LUN9 -ostype
linux -foreign-disk 800DT$HuVWBo
Created a LUN of size 10g (10737418240)
```

Create import relationships

The LUNs have now been created but are not configured as a replication destination. Before this step can be

taken, the LUNs must first be placed offline. This extra step is designed to protect data from user errors. If ONTAP allowed a migration to be performed on an online LUN, it would create a risk that a typographical error could result in overwriting active data. The extra step of forcing the user to first take a LUN offline helps verify that the correct target LUN is used as a migration destination.

```
Cluster01::*> lun offline -vserver vserver1 -path /vol/new_asm/LUN0
Warning: This command will take LUN "/vol/new_asm/LUN0" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
Cluster01::*> lun offline -vserver vserver1 -path /vol/new_asm/LUN1
Warning: This command will take LUN "/vol/new_asm/LUN1" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
...
Warning: This command will take LUN "/vol/new_lvm/LUN8" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
Cluster01::*> lun offline -vserver vserver1 -path /vol/new_lvm/LUN9
Warning: This command will take LUN "/vol/new_lvm/LUN9" in Vserver
         "vserver1" offline.
Do you want to continue? {y|n}: y
```

After the LUNs are offline, you can establish the import relationship by passing the foreign LUN serial number to the `lun import create` command.

```
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_asm/LUN0
-foreign-disk 800DT$HuVWBW
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_asm/LUN1
-foreign-disk 800DT$HuVWBX
...
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_lvm/LUN8
-foreign-disk 800DT$HuVWBn
Cluster01::*> lun import create -vserver vserver1 -path /vol/new_lvm/LUN9
-foreign-disk 800DT$HuVWBo
Cluster01::*>
```

After all import relationships are established, the LUNs can be placed back online.

```
Cluster01::*> lun online -vserver vserver1 -path /vol/new_asm/LUN0
Cluster01::*> lun online -vserver vserver1 -path /vol/new_asm/LUN1
...
Cluster01::*> lun online -vserver vserver1 -path /vol/new_lvm/LUN8
Cluster01::*> lun online -vserver vserver1 -path /vol/new_lvm/LUN9
Cluster01::*>
```

Create initiator group

An initiator group (igroup) is part of the ONTAP LUN masking architecture. A newly created LUN is not accessible unless a host is first granted access. This is done by creating an igroup that lists either the FC WWNs or iSCSI initiator names that should be granted access. At the time this report was written, FLI was supported only for FC LUNs. However, converting to iSCSI postmigration is a simple task, as shown in Protocol Conversion.

In this example, an igroup is created that contains two WWNs that correspond to the two ports available on the host's HBA.

```
Cluster01::*> igroup create linuxhost -protocol fcp -ostype linux
-initiator 21:00:00:0e:1e:16:63:50 21:00:00:0e:1e:16:63:51
```

Map new LUNs to host

Following igroup creation, the LUNs are then mapped to the defined igroup. These LUNs are available only to the WWNs included in this igroup. NetApp assumes at this stage in the migration process that the host has not been zoned to ONTAP. This is important because if the host is simultaneously zoned to the foreign array and the new ONTAP system, then there is a risk that LUNs bearing the same serial number could be discovered on each array. This situation could lead to multipath malfunctions or damage to data.

```
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN0 -igroup
linuxhost
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN1 -igroup
linuxhost
...
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN8 -igroup
linuxhost
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN9 -igroup
linuxhost
Cluster01::*>
```

Oracle migration with FLI - cutover

Some disruption during a foreign LUN import is unavoidable because of the need to change the FC network configuration. However, the disruption does not have to last much longer than the time required to restart the database environment and update FC zoning to switch the host FC connectivity from the foreign LUN to ONTAP.

This process can be summarized as follows:

1. Quiesce all LUN activity on the foreign LUNs.
2. Redirect host FC connections to the new ONTAP system.
3. Trigger the import process.
4. Rediscover the LUNs.
5. Restart the database.

You do not need to wait for the migration process to complete. As soon as the migration for a given LUN begins, it is available on ONTAP and can serve data while the data copy process continues. All reads are passed through to the foreign LUN, and all writes are synchronously written to both arrays. The copy operation is very fast and the overhead of redirecting FC traffic is minimal, so any impact on performance should be transient and minimal. If there is concern, you can delay restarting the environment until after the migration process is complete and the import relationships have been deleted.

Shut down database

The first step in quiescing the environment in this example is to shut down the database.

```
[oracle@host1 bin]$ . oraenv
ORACLE_SID = [oracle] ? FLIDB
The Oracle base remains unchanged with value /orabin
[oracle@host1 bin]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced
Analytics
and Real Application Testing options
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

Shut down grid services

One of the SAN-based file systems being migrated also includes the Oracle ASM services. Quiescing the underlying LUNs requires dismounting the file systems, which in turn means stopping any processes with open files on this file system.

```
[oracle@host1 bin]$ ./crsctl stop has -f
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'host1'
CRS-2673: Attempting to stop 'ora.evmd' on 'host1'
CRS-2673: Attempting to stop 'ora.DATA.dg' on 'host1'
CRS-2673: Attempting to stop 'ora.LISTENER.lsnr' on 'host1'
CRS-2677: Stop of 'ora.DATA.dg' on 'host1' succeeded
CRS-2673: Attempting to stop 'ora.asm' on 'host1'
CRS-2677: Stop of 'ora.LISTENER.lsnr' on 'host1' succeeded
CRS-2677: Stop of 'ora.evmd' on 'host1' succeeded
CRS-2677: Stop of 'ora.asm' on 'host1' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'host1'
CRS-2677: Stop of 'ora.cssd' on 'host1' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources
on 'host1' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[oracle@host1 bin]$
```

Dismount file systems

If all the processes are shut down, the umount operation succeeds. If permission is denied, there must be a process with a lock on the file system. The `fuser` command can help identify these processes.

```
[root@host1 ~]# umount /orabin
[root@host1 ~]# umount /backups
```

Deactivate volume groups

After all file systems in a given volume group are dismounted, the volume group can be deactivated.

```
[root@host1 ~]# vgchange --activate n sanvg
  0 logical volume(s) in volume group "sanvg" now active
[root@host1 ~]#
```

FC network changes

The FC zones can now be updated to remove all access from the host to the foreign array and establish access to ONTAP.

Start import process

To start the LUN import processes, run the `lun import start` command.

```
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_asm/LUN0
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_asm/LUN1
...
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_lvm/LUN8
Cluster01::lun import*> lun import start -vserver vserver1 -path
/vol/new_lvm/LUN9
Cluster01::lun import*>
```

Monitor import progress

The import operation can be monitored with the `lun import show` command. As shown below, the import of all 20 LUNs is underway, which means that data is now accessible through ONTAP even though the data copy operation still progresses.

```
Cluster01::lun import*> lun import show -fields path,percent-complete
vserver   foreign-disk path              percent-complete
--------- ------------ ----------------- ----------------
vserver1  800DT$HuVWB/ /vol/new_asm/LUN4 5
vserver1  800DT$HuVWBW /vol/new_asm/LUN0 5
vserver1  800DT$HuVWBX /vol/new_asm/LUN1 6
vserver1  800DT$HuVWBY /vol/new_asm/LUN2 6
vserver1  800DT$HuVWBZ /vol/new_asm/LUN3 5
vserver1  800DT$HuVWBa /vol/new_asm/LUN5 4
vserver1  800DT$HuVWBb /vol/new_asm/LUN6 4
vserver1  800DT$HuVWBc /vol/new_asm/LUN7 4
vserver1  800DT$HuVWBd /vol/new_asm/LUN8 4
vserver1  800DT$HuVWBe /vol/new_asm/LUN9 4
vserver1  800DT$HuVWBf /vol/new_lvm/LUN0 5
vserver1  800DT$HuVWBg /vol/new_lvm/LUN1 4
vserver1  800DT$HuVWBh /vol/new_lvm/LUN2 4
vserver1  800DT$HuVWBi /vol/new_lvm/LUN3 3
vserver1  800DT$HuVWBj /vol/new_lvm/LUN4 3
vserver1  800DT$HuVWBk /vol/new_lvm/LUN5 3
vserver1  800DT$HuVWBl /vol/new_lvm/LUN6 4
vserver1  800DT$HuVWBm /vol/new_lvm/LUN7 3
vserver1  800DT$HuVWBn /vol/new_lvm/LUN8 2
vserver1  800DT$HuVWBo /vol/new_lvm/LUN9 2
20 entries were displayed.
```

If you require an offline process, delay rediscovering or restarting services until the `lun import show` command indicates that all migration is successful and complete. You can then complete the migration process as described in Foreign LUN Import—Completion.

If you require an online migration, proceed to rediscover the LUNs in their new home and bring up the services.

Scan for SCSI device changes

In most cases, the simplest option to rediscover new LUNs is to restart the host. Doing so automatically removes old stale devices, properly discovers all new LUNs, and builds associated devices such as multipathing devices. The example here shows a wholly online process for demonstration purposes.

Caution: Before restarting a host, make sure that all entries in `/etc/fstab` that reference migrated SAN resources are commented out. If this is not done and there are problems with LUN access, the OS might not boot. This situation does not damage data. However, it can be very inconvenient to boot into rescue mode or a similar mode and correct the `/etc/fstab` so that the OS can be booted to enable troubleshooting.

The LUNs on the version of Linux used in this example can be rescanned with the `rescan-scsi-bus.sh` command. If the command is successful, each LUN path should appear in the output. The output can be difficult to interpret, but, if the zoning and igroup configuration was correct, many LUNs should appear that include a `NETAPP` vendor string.

```
[root@host1 /]# rescan-scsi-bus.sh
Scanning SCSI subsystem for new devices
Scanning host 0 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
 Scanning for device 0 2 0 0 ...
OLD: Host: scsi0 Channel: 02 Id: 00 Lun: 00
      Vendor: LSI      Model: RAID SAS 6G 0/1  Rev: 2.13
      Type:   Direct-Access                    ANSI SCSI revision: 05
Scanning host 1 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
 Scanning for device 1 0 0 0 ...
OLD: Host: scsi1 Channel: 00 Id: 00 Lun: 00
      Vendor: Optiarc  Model: DVD RW AD-7760H  Rev: 1.41
      Type:   CD-ROM                           ANSI SCSI revision: 05
Scanning host 2 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 3 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 4 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 5 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 6 for  SCSI target IDs  0 1 2 3 4 5 6 7, all LUNs
Scanning host 7 for  all SCSI target IDs, all LUNs
 Scanning for device 7 0 0 10 ...
OLD: Host: scsi7 Channel: 00 Id: 00 Lun: 10
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
 Scanning for device 7 0 0 11 ...
OLD: Host: scsi7 Channel: 00 Id: 00 Lun: 11
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
 Scanning for device 7 0 0 12 ...
...
OLD: Host: scsi9 Channel: 00 Id: 01 Lun: 18
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
 Scanning for device 9 0 1 19 ...
OLD: Host: scsi9 Channel: 00 Id: 01 Lun: 19
      Vendor: NETAPP   Model: LUN C-Mode       Rev: 8300
      Type:   Direct-Access                    ANSI SCSI revision: 05
0 new or changed device(s) found.
0 remapped or resized device(s) found.
0 device(s) removed.
```

Check for multipath devices

The LUN discovery process also triggers the recreation of multipath devices, but the Linux multipathing driver is known to have occasional problems. The output of `multipath -ll` should be checked to verify that the output looks as expected. For example, the output below shows multipath devices associated with a `NETAPP` vendor string. Each device has four paths, with two at a priority of 50 and two at a priority of 10. Although the exact output can vary with different versions of Linux, this output looks as expected.

Reference the host utilities documentation for the version of Linux you use to verify that the `/etc/multipath.conf` settings are correct.

```
[root@host1 /]# multipath -ll
3600a098038303558735d493762504b36 dm-5 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:4  sdat 66:208 active ready running
| `- 9:0:1:4  sdbn 68:16  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:4  sdf  8:80   active ready running
  `- 9:0:0:4  sdz  65:144 active ready running
3600a098038303558735d493762504b2d dm-10 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:8  sdax 67:16  active ready running
| `- 9:0:1:8  sdbr 68:80  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:8  sdj  8:144  active ready running
  `- 9:0:0:8  sdad 65:208 active ready running
...
3600a098038303558735d493762504b37 dm-8 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:5  sdau 66:224 active ready running
| `- 9:0:1:5  sdbo 68:32  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:5  sdg  8:96   active ready running
  `- 9:0:0:5  sdaa 65:160 active ready running
3600a098038303558735d493762504b4b dm-22 NETAPP  ,LUN C-Mode
size=10G features='4 queue_if_no_path pg_init_retries 50
retain_attached_hw_handle' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 7:0:1:19 sdbi 67:192 active ready running
| `- 9:0:1:19 sdcc 69:0   active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 7:0:0:19 sdu  65:64  active ready running
  `- 9:0:0:19 sdao 66:128 active ready running
```

Reactivate LVM volume group

If the LVM LUNs have been properly discovered, the `vgchange --activate y` command should succeed.

This is a good example of the value of a logical volume manager. A change in the WWN of a LUN or even a serial number is unimportant because the volume group metadata is written on the LUN itself.

The OS scanned the LUNs and discovered a small amount of data written on the LUN that identifies it as a physical volume belonging to the `sanvg volumegroup`. It then built all of the required devices. All that is required is to reactivate the volume group.

```
[root@host1 /]# vgchange --activate y sanvg
  Found duplicate PV fpCzdLTuKfy2xDZjai1NliJh3TjLUBiT: using
/dev/mapper/3600a098038303558735d493762504b46 not /dev/sdp
  Using duplicate PV /dev/mapper/3600a098038303558735d493762504b46 from
subsystem DM, ignoring /dev/sdp
  2 logical volume(s) in volume group "sanvg" now active
```

Remount file systems

After the volume group is reactivated, the file systems can be mounted with all of the original data intact. As discussed previously, the file systems are fully operational even if data replication is still active in the back group.

```
[root@host1 /]# mount /orabin
[root@host1 /]# mount /backups
[root@host1 /]# df -k
Filesystem                      1K-blocks      Used Available Use%
Mounted on
/dev/mapper/rhel-root            52403200   8837100  43566100  17% /
devtmpfs                         65882776         0  65882776   0% /dev
tmpfs                             6291456        84   6291372   1%
/dev/shm
tmpfs                            65898668      9884  65888784   1% /run
tmpfs                            65898668         0  65898668   0%
/sys/fs/cgroup
/dev/sda1                          505580    224828    280752  45% /boot
fas8060-nfs-public:/install      199229440 119368256  79861184  60%
/install
fas8040-nfs-routable:/snapomatic   9961472     30528   9930944   1%
/snapomatic
tmpfs                            13179736        16  13179720   1%
/run/user/42
tmpfs                            13179736         0  13179736   0%
/run/user/0
/dev/mapper/sanvg-lvorabin       20961280  12357456   8603824  59%
/orabin
/dev/mapper/sanvg-lvbackups      73364480  62947536  10416944  86%
/backups
```

Rescan for ASM devices

The ASMlib devices should have been rediscovered when the SCSI devices were rescanned. Rediscovery can be verified online by restarting ASMlib and then scanning the disks.

This step is only relevant to ASM configurations where ASMlib is used.

Caution: Where ASMlib is not used, the `/dev/mapper` devices should have been automatically recreated. However, the permissions might not be correct. You must set special permissions on the underlying devices for ASM in the absence of ASMlib. Doing so is usually accomplished through special entries in either the `/etc/multipath.conf` or `udev` rules, or possibly in both rule sets. These files might need to be updated to reflect changes in the environment in terms of WWNs or serial numbers to make sure that the ASM devices still have the correct permissions.

In this example, restarting ASMlib and scanning for disks show the same 10 ASM LUNs as the original environment.

```
[root@host1 /]# oracleasm exit
Unmounting ASMlib driver filesystem: /dev/oracleasm
Unloading module "oracleasm": oracleasm
[root@host1 /]# oracleasm init
Loading module "oracleasm": oracleasm
Configuring "oracleasm" to use device physical block size
Mounting ASMlib driver filesystem: /dev/oracleasm
[root@host1 /]# oracleasm scandisks
Reloading disk partitions: done
Cleaning any stale ASM disks...
Scanning system for ASM disks...
Instantiating disk "ASM0"
Instantiating disk "ASM1"
Instantiating disk "ASM2"
Instantiating disk "ASM3"
Instantiating disk "ASM4"
Instantiating disk "ASM5"
Instantiating disk "ASM6"
Instantiating disk "ASM7"
Instantiating disk "ASM8"
Instantiating disk "ASM9"
```

Restart grid services

Now that the LVM and ASM devices are online and available, the grid services can be restarted.

```
[root@host1 /]# cd /orabin/product/12.1.0/grid/bin
[root@host1 bin]# ./crsctl start has
```

Restart database

After the grid services have been restarted, the database can be brought up. It might be necessary to wait a few minutes for the ASM services to become fully available before trying to start the database.

```
[root@host1 bin]# su - oracle
[oracle@host1 ~]$ . oraenv
ORACLE_SID = [oracle] ? FLIDB
The Oracle base has been set to /orabin
[oracle@host1 ~]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0
Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> startup
ORACLE instance started.
Total System Global Area 3221225472 bytes
Fixed Size                  4502416 bytes
Variable Size            1207962736 bytes
Database Buffers         1996488704 bytes
Redo Buffers               12271616 bytes
Database mounted.
Database opened.
SQL>
```

Oracle migration with FLI - completion

From a host point of view, the migration is complete, but I/O is still served from the foreign array until the import relationships are deleted.

Before deleting the relationships, you must confirm that the migration process is complete for all LUNs.

```
Cluster01::*> lun import show -vserver vserver1 -fields foreign-
disk,path,operational-state
vserver   foreign-disk path              operational-state
--------- ------------ ----------------- -----------------
vserver1 800DT$HuVWB/ /vol/new_asm/LUN4 completed
vserver1 800DT$HuVWBW /vol/new_asm/LUN0 completed
vserver1 800DT$HuVWBX /vol/new_asm/LUN1 completed
vserver1 800DT$HuVWBY /vol/new_asm/LUN2 completed
vserver1 800DT$HuVWBZ /vol/new_asm/LUN3 completed
vserver1 800DT$HuVWBa /vol/new_asm/LUN5 completed
vserver1 800DT$HuVWBb /vol/new_asm/LUN6 completed
vserver1 800DT$HuVWBc /vol/new_asm/LUN7 completed
vserver1 800DT$HuVWBd /vol/new_asm/LUN8 completed
vserver1 800DT$HuVWBe /vol/new_asm/LUN9 completed
vserver1 800DT$HuVWBf /vol/new_lvm/LUN0 completed
vserver1 800DT$HuVWBg /vol/new_lvm/LUN1 completed
vserver1 800DT$HuVWBh /vol/new_lvm/LUN2 completed
vserver1 800DT$HuVWBi /vol/new_lvm/LUN3 completed
vserver1 800DT$HuVWBj /vol/new_lvm/LUN4 completed
vserver1 800DT$HuVWBk /vol/new_lvm/LUN5 completed
vserver1 800DT$HuVWBl /vol/new_lvm/LUN6 completed
vserver1 800DT$HuVWBm /vol/new_lvm/LUN7 completed
vserver1 800DT$HuVWBn /vol/new_lvm/LUN8 completed
vserver1 800DT$HuVWBo /vol/new_lvm/LUN9 completed
20 entries were displayed.
```

Delete import relationships

When the migration process is complete, delete the migration relationship. After you have done so, I/O is served exclusively from the drives on ONTAP.

```
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_asm/LUN0
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_asm/LUN1
...
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_lvm/LUN8
Cluster01::*> lun import delete -vserver vserver1 -path /vol/new_lvm/LUN9
```

Deregister foreign LUNs

Finally, modify the disk to remove the `is-foreign` designation.

```
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBW} -is
-foreign false
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBX} -is
-foreign false
...
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBn} -is
-foreign false
Cluster01::*> storage disk modify {-serial-number 800DT$HuVWBo} -is
-foreign false
Cluster01::*>
```

Oracle migration with FLI - protocol conversion

Changing the protocol used to access a LUN is a common requirement.

In some cases, it is part of an overall strategy to migrate data to the cloud. TCP/IP is the protocol of the cloud, and changing from FC to iSCSI allows easier migration into various cloud environments. In other cases, iSCSI might be desirable to leverage the decreased costs of an IP SAN. On occasion, a migration might use a different protocol as a temporary measure. For example, if a foreign array and ONTAP based LUNs cannot coexist on the same HBAs, you can use iSCSI LUNs long enough to copy data from the old array. You can then convert back to FC after the old LUNs are removed from the system.

The following procedure demonstrates conversion from FC to iSCSI, but the overall principles apply to a reverse iSCSI to FC conversion.

Install iSCSI initiator

Most operating systems include a software iSCSI initiator by default, but if one is not included, it can be easily installed.

```
[root@host1 /]# yum install -y iscsi-initiator-utils
Loaded plugins: langpacks, product-id, search-disabled-repos,
subscription-
              : manager
Resolving Dependencies
--> Running transaction check
---> Package iscsi-initiator-utils.x86_64 0:6.2.0.873-32.el7 will be
updated
--> Processing Dependency: iscsi-initiator-utils = 6.2.0.873-32.el7 for
package: iscsi-initiator-utils-iscsiuio-6.2.0.873-32.el7.x86_64
---> Package iscsi-initiator-utils.x86_64 0:6.2.0.873-32.0.2.el7 will be
an update
--> Running transaction check
---> Package iscsi-initiator-utils-iscsiuio.x86_64 0:6.2.0.873-32.el7 will
be updated
---> Package iscsi-initiator-utils-iscsiuio.x86_64 0:6.2.0.873-32.0.2.el7
will be an update
```

```
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
===
 Package                         Arch   Version              Repository
Size
================================================================================
===
Updating:
 iscsi-initiator-utils           x86_64 6.2.0.873-32.0.2.el7 ol7_latest 416
k
Updating for dependencies:
 iscsi-initiator-utils-iscsiuio x86_64 6.2.0.873-32.0.2.el7 ol7_latest  84
k
Transaction Summary
================================================================================
===
Upgrade  1 Package (+1 Dependent package)
Total download size: 501 k
Downloading packages:
No Presto metadata available for ol7_latest
(1/2): iscsi-initiator-utils-6.2.0.873-32.0.2.el7.x86_6 | 416 kB   00:00
(2/2): iscsi-initiator-utils-iscsiuio-6.2.0.873-32.0.2. |  84 kB   00:00
--------------------------------------------------------------------------------
---
Total                                           2.8 MB/s | 501 kB
00:00Cluster01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.0.2.el7.x86
1/4
  Updating   : iscsi-initiator-utils-6.2.0.873-32.0.2.el7.x86_64
2/4
  Cleanup    : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.el7.x86_64
3/4
  Cleanup    : iscsi-initiator-utils-6.2.0.873-32.el7.x86_64
4/4
rhel-7-server-eus-rpms/7Server/x86_64/productid       | 1.7 kB     00:00
rhel-7-server-rpms/7Server/x86_64/productid           | 1.7 kB     00:00
  Verifying  : iscsi-initiator-utils-6.2.0.873-32.0.2.el7.x86_64
1/4
  Verifying  : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.0.2.el7.x86
2/4
  Verifying  : iscsi-initiator-utils-iscsiuio-6.2.0.873-32.el7.x86_64
```

```
3/4
  Verifying  : iscsi-initiator-utils-6.2.0.873-32.el7.x86_64
4/4
Updated:
  iscsi-initiator-utils.x86_64 0:6.2.0.873-32.0.2.el7
Dependency Updated:
  iscsi-initiator-utils-iscsiuio.x86_64 0:6.2.0.873-32.0.2.el7
Complete!
[root@host1 /]#
```

Identify iSCSI initiator name

A unique iSCSI initiator name is generated during the installation process. On Linux, it is located in the `/etc/iscsi/initiatorname.iscsi` file. This name is used to identify the host on the IP SAN.

```
[root@host1 /]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1992-05.com.redhat:497bd66ca0
```

Create new initiator group

An initiator group (igroup) is part of the ONTAP LUN masking architecture. A newly created LUN is not accessible unless a host is first granted access. This step is accomplished by creating an igroup that lists either the FC WWNs or iSCSI initiator names that require access.

In this example, an igroup is created that contains the iSCSI initiator of the Linux host.

```
Cluster01::*> igroup create -igroup linuxiscsi -protocol iscsi -ostype
linux -initiator iqn.1994-05.com.redhat:497bd66ca0
```

Shut down environment

Before changing the LUN protocol, the LUNs must be fully quiesced. Any database on one of the LUNs being converted must be shut down, file systems must be dismounted, and volume groups must be deactivated. Where ASM is used, make sure that the ASM disk group is dismounted and shut down all grid services.

Unmap LUNs from FC network

After the LUNs are fully quiesced, remove the mappings from the original FC igroup.

```
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_asm/LUN0 -igroup
linuxhost
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_asm/LUN1 -igroup
linuxhost
...
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_lvm/LUN8 -igroup
linuxhost
Cluster01::*> lun unmap -vserver vserver1 -path /vol/new_lvm/LUN9 -igroup
linuxhost
```

Remap LUNs to IP network

Grant access to each LUN to the new iSCSI-based initiator group.

```
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN0 -igroup
linuxiscsi
Cluster01::*> lun map -vserver vserver1 -path /vol/new_asm/LUN1 -igroup
linuxiscsi
...
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN8 -igroup
linuxiscsi
Cluster01::*> lun map -vserver vserver1 -path /vol/new_lvm/LUN9 -igroup
linuxiscsi
Cluster01::*>
```

Discover iSCSI targets

There are two phases to iSCSI discovery. The first is to discover the targets, which is not the same as discovering a LUN. The `iscsiadm` command shown below probes the portal group specified by the `-p argument` and stores a list of all IP addresses and ports that offer iSCSI services. In this case, there are four IP addresses that have iSCSI services on the default port 3260.

> This command can take several minutes to complete if any of the target IP addresses cannot be reached.

```
[root@host1 ~]# iscsiadm -m discovery -t st -p fas8060-iscsi-public1
10.63.147.197:3260,1033 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
10.63.147.198:3260,1034 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
172.20.108.203:3260,1030 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
172.20.108.202:3260,1029 iqn.1992-
08.com.netapp:sn.807615e9ef6111e5a5ae90e2ba5b9464:vs.3
```

Discover iSCSI LUNs

After the iSCSI targets are discovered, restart the iSCSI service to discover the available iSCSI LUNs and build associated devices such as multipath or ASMlib devices.

```
[root@host1 ~]# service iscsi restart
Redirecting to /bin/systemctl restart  iscsi.service
```

Restart environment

Restart the environment by reactivating volume groups, remounting file systems, restarting RAC services, and so on. As a precaution, NetApp recommends that you reboot the server after the conversion process is complete to be certain that all configuration files are correct and all stale devices are removed.

Caution: Before restarting a host, make sure that all entries in `/etc/fstab` that reference migrated SAN resources are commented out. If this step is not taken and there are problems with LUN access, the result can be an OS that does not boot. This issue does not damage data. However, it can be very inconvenient to boot into rescue mode or a similar mode and correct `/etc/fstab` so that the OS can be booted to allow troubleshooting efforts to begin.

Oracle migration procedure sample scripts

The scripts presented are provided as examples of how to script various OS and database tasks. They are supplied as is. If support is required for a particular procedure, contact NetApp or a NetApp reseller.

Database shutdown

The following Perl script takes a single argument of the Oracle SID and shuts down a database. It can be run as the Oracle user or as root.

```
#! /usr/bin/perl
use strict;
use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
my $uid=$<;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
77 Migration of Oracle Databases to NetApp Storage Systems © 2021 NetApp,
Inc. All rights reserved
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
shutdown immediate;
EOF2
'
`;}
else {
@out=`. oraenv << EOF1
$oraclesid
EOF4
sqlplus / as sysdba << EOF2
shutdown immediate;
EOF2
`;};
print @out;
if ("@out" =~ /ORACLE instance shut down/) {
print "$oraclesid shut down\n";
exit 0;}
elsif ("@out" =~ /Connected to an idle instance/) {
print "$oraclesid already shut down\n";
exit 0;}
else {
print "$oraclesid failed to shut down\n";
exit 1;}
```

Database startup

The following Perl script takes a single argument of the Oracle SID and shuts down a database. It can be run as the Oracle user or as root.

```
#! /usr/bin/perl
use strict;
use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
my $uid=$<;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
startup;
EOF2
'
`;}
else {
@out=`. oraenv << EOF3
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
startup;
EOF2
`;};
print @out;
if ("@out" =~ /Database opened/) {
print "$oraclesid started\n";
exit 0;}
elsif ("@out" =~ /cannot start already-running ORACLE/) {
print "$oraclesid already started\n";
exit 1;}
else {
78 Migration of Oracle Databases to NetApp Storage Systems © 2021 NetApp,
Inc. All rights reserved
print "$oraclesid failed to start\n";
exit 1;}
```

Convert file system to read-only

The following script takes a file- system argument and attempts to dismount and remount it as read-only. Doing so is useful during migration processes in which a file system must be kept available to replicate data and yet must be protected against accidental damage.

```
#! /usr/bin/perl
use strict;
#use warnings;
my $filesystem=$ARGV[0];
my @out=`umount '$filesystem'`;
if ($? == 0) {
 print "$filesystem unmounted\n";
 @out = `mount -o ro '$filesystem'`;
 if ($? == 0) {
 print "$filesystem mounted read-only\n";
 exit 0;}}
else {
 print "Unable to unmount $filesystem\n";
 exit 1;}
print @out;
```

Replace file system

The following script example is used to replace one file system with another. Because it edits the `/etc/fstab `file, it must run as root. It accepts a single comma-delimited argument of the old and new file systems.

1. To replace the file system, run the following script:

```
#! /usr/bin/perl
use strict;
#use warnings;
my $oldfs;
my $newfs;
my @oldfstab;
my @newfstab;
my $source;
my $mountpoint;
my $leftover;
my $oldfstabentry='';
my $newfstabentry='';
my $migratedfstabentry='';
($oldfs, $newfs) = split (',',$ARGV[0]);
open(my $filehandle, '<', '/etc/fstab') or die "Could not open
/etc/fstab\n";
while (my $line = <$filehandle>) {
 chomp $line;
 ($source, $mountpoint, $leftover) = split(/[ , ]/,$line, 3);
 if ($mountpoint eq $oldfs) {
 $oldfstabentry = "#Removed by swap script $source $oldfs $leftover";}
 elsif ($mountpoint eq $newfs) {
```

```
 $newfstabentry = "#Removed by swap script $source $newfs $leftover";
 $migratedfstabentry = "$source $oldfs $leftover";}
 else {
 push (@newfstab, "$line\n")}}
79 Migration of Oracle Databases to NetApp Storage Systems © 2021
NetApp, Inc. All rights reserved
push (@newfstab, "$oldfstabentry\n");
push (@newfstab, "$newfstabentry\n");
push (@newfstab, "$migratedfstabentry\n");
close($filehandle);
if ($oldfstabentry eq ''){
 die "Could not find $oldfs in /etc/fstab\n";}
if ($newfstabentry eq ''){
 die "Could not find $newfs in /etc/fstab\n";}
my @out=`umount '$newfs'`;
if ($? == 0) {
 print "$newfs unmounted\n";}
else {
 print "Unable to unmount $newfs\n";
 exit 1;}
@out=`umount '$oldfs'`;
if ($? == 0) {
 print "$oldfs unmounted\n";}
else {
 print "Unable to unmount $oldfs\n";
 exit 1;}
system("cp /etc/fstab /etc/fstab.bak");
open ($filehandle, ">", '/etc/fstab') or die "Could not open /etc/fstab
for writing\n";
for my $line (@newfstab) {
 print $filehandle $line;}
close($filehandle);
@out=`mount '$oldfs'`;
if ($? == 0) {
 print "Mounted updated $oldfs\n";
 exit 0;}
else{
 print "Unable to mount updated $oldfs\n";
 exit 1;}
exit 0;
```

As an example of this script's use, assume that data in `/oradata` is migrated to `/neworadata` and `/logs` is migrated to `/newlogs`. One of the simplest methods to perform this task is by using a simple file copy operation to relocate the new device back to the original mountpoint.

2. Assume that the old and new file systems are present in the `/etc/fstab` file as follows:

```
cluster01:/vol_oradata /oradata nfs rw,bg,vers=3,rsize=65536,wsize=65536
0 0
cluster01:/vol_logs /logs nfs rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_neworadata /neworadata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_newlogs /newlogs nfs rw,bg,vers=3,rsize=65536,wsize=65536
0 0
```

3. When run, this script unmounts the current file system and replaces it with the new:

```
[root@jfsc3 scripts]# ./swap.fs.pl /oradata,/neworadata
/neworadata unmounted
/oradata unmounted
Mounted updated /oradata
[root@jfsc3 scripts]# ./swap.fs.pl /logs,/newlogs
/newlogs unmounted
/logs unmounted
Mounted updated /logs
```

4. The script also updates the `/etc/fstab` file accordingly. In the example shown here, it includes the following changes:

```
#Removed by swap script cluster01:/vol_oradata /oradata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
#Removed by swap script cluster01:/vol_neworadata /neworadata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_neworadata /oradata nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
#Removed by swap script cluster01:/vol_logs /logs nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
#Removed by swap script cluster01:/vol_newlogs /newlogs nfs
rw,bg,vers=3,rsize=65536,wsize=65536 0 0
cluster01:/vol_newlogs /logs nfs rw,bg,vers=3,rsize=65536,wsize=65536 0
0
```

Automated database migration

This example demonstrates the use of shutdown, startup, and file system replacement scripts to fully automate a migration.

```
#! /usr/bin/perl
use strict;
#use warnings;
```

```
my $oraclesid=$ARGV[0];
my @oldfs;
my @newfs;
my $x=1;
while ($x < scalar(@ARGV)) {
   ($oldfs[$x-1], $newfs[$x-1]) = split (',',$ARGV[$x]);
   $x+=1;}
my @out=`./dbshut.pl '$oraclesid'`;
print @out;
if ($? ne 0) {
   print "Failed to shut down database\n";
   exit 0;}
$x=0;
while ($x < scalar(@oldfs)) {
   my @out=`./mk.fs.readonly.pl '$oldfs[$x]'`;
   if ($? ne 0) {
      print "Failed to make filesystem $oldfs[$x] readonly\n";
      exit 0;}
   $x+=1;}
$x=0;
while ($x < scalar(@oldfs)) {
   my @out=`rsync -rlpogt --stats --progress --exclude='.snapshot'
'$oldfs[$x]/' '/$newfs[$x]/'`;
   print @out;
   if ($? ne 0) {
      print "Failed to copy filesystem $oldfs[$x] to $newfs[$x]\n";
      exit 0;}
   else {
      print "Succesfully replicated filesystem $oldfs[$x] to
$newfs[$x]\n";}
   $x+=1;}
$x=0;
while ($x < scalar(@oldfs)) {
   print "swap $x $oldfs[$x] $newfs[$x]\n";
   my @out=`./swap.fs.pl '$oldfs[$x],$newfs[$x]'`;
   print @out;
   if ($? ne 0) {
      print "Failed to swap filesystem $oldfs[$x] for $newfs[$x]\n";
      exit 1;}
   else {
      print "Swapped filesystem $oldfs[$x] for $newfs[$x]\n";}
   $x+=1;}
my @out=`./dbstart.pl '$oraclesid'`;
print @out;
```

Display file locations

This script collects a number of critical database parameters and prints them in an easy-to-read format. This script can be useful when reviewing data layouts. In addition, the script can be modified for other uses.

```
#! /usr/bin/perl
#use strict;
#use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
sub dosql{
        my $command = @_[0];
        my @lines;
        my $uid=$<;
        if ($uid == 0) {
        @lines=`su - $oracleuser -c "export ORAENV_ASK=NO;export
ORACLE_SID=$oraclesid;. oraenv -s << EOF1
EOF1
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
"
        `;}
        else {
        $command=~s/\\\\\\\/\\/g;
        @lines=`export ORAENV_ASK=NO;export ORACLE_SID=$oraclesid;. oraenv
-s << EOF1
EOF1
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
        `;};
return @lines}
print "\n";
@out=dosql('select name from v\\\\\$datafile;');
print "$oraclesid datafiles:\n";
for $line (@out) {
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('select member from v\\\\\$logfile;');
print "$oraclesid redo logs:\n";
for $line (@out) {
```

```
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('select name from v\\\\\$tempfile;');
print "$oraclesid temp datafiles:\n";
for $line (@out) {
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('show parameter spfile;');
print "$oraclesid spfile\n";
for $line (@out) {
        chomp($line);
        if (length($line)>0) {print "$line\n";}}
print "\n";
@out=dosql('select name||\' \'||value from v\\\\\$parameter where
isdefault=\'FALSE\';');
print "$oraclesid key parameters\n";
for $line (@out) {
        chomp($line);
        if ($line =~ /control_files/) {print "$line\n";}
        if ($line =~ /db_create/) {print "$line\n";}
        if ($line =~ /db_file_name_convert/) {print "$line\n";}
        if ($line =~ /log_archive_dest/) {print "$line\n";}}
        if ($line =~ /log_file_name_convert/) {print "$line\n";}
        if ($line =~ /pdb_file_name_convert/) {print "$line\n";}
        if ($line =~ /spfile/) {print "$line\n";}
print "\n";
```

ASM migration cleanup

```
#! /usr/bin/perl
#use strict;
#use warnings;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my @out;
sub dosql{
        my $command = @_[0];
        my @lines;
        my $uid=$<;
        if ($uid == 0) {
        @lines=`su - $oracleuser -c "export ORAENV_ASK=NO;export
ORACLE_SID=$oraclesid;. oraenv -s << EOF1
EOF1
```

```
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
"
        `;}
        else {
        $command=~s/\\\\\\\/\\/g;
        @lines=`export ORAENV_ASK=NO;export ORACLE_SID=$oraclesid;. oraenv
-s << EOF1
EOF1
sqlplus -S / as sysdba << EOF2
set heading off
$command
EOF2
        `;}
return @lines}
print "\n";
@out=dosql('select name from v\\\\\$datafile;');
print @out;
print "shutdown immediate;\n";
print "startup mount;\n";
print "\n";
for $line (@out) {
        if (length($line) > 1) {
                chomp($line);
                ($first, $second,$third,$fourth)=split('_',$line);
                $fourth =~ s/^TS-//;
                $newname=lc("$fourth.dbf");
                $path2file=$line;
                $path2file=~ /(^.*.\/)/;
                print "host mv $line $1$newname\n";}}
print "\n";
for $line (@out) {
        if (length($line) > 1) {
                chomp($line);
                ($first, $second,$third,$fourth)=split('_',$line);
                $fourth =~ s/^TS-//;
                $newname=lc("$fourth.dbf");
                $path2file=$line;
                $path2file=~ /(^.*.\/)/;
                print "alter database rename file '$line' to
'$1$newname';\n";}}
print "alter database open;\n";
print "\n";
```

ASM to file system name conversion

```
set serveroutput on;
set wrap off;
declare
    cursor df is select file#, name from v$datafile;
    cursor tf is select file#, name from v$tempfile;
    cursor lf is select member from v$logfile;
    firstline boolean := true;
begin
    dbms_output.put_line(CHR(13));
    dbms_output.put_line('Parameters for log file conversion:');
    dbms_output.put_line(CHR(13));
    dbms_output.put('*.log_file_name_convert = ');
    for lfrec in lf loop
        if (firstline = true) then
            dbms_output.put('''' || lfrec.member || ''', ');
            dbms_output.put('''/NEW_PATH/' ||
regexp_replace(lfrec.member,'^.*/','') || '''');
        else
            dbms_output.put(',''' || lfrec.member || ''', ');
            dbms_output.put('''/NEW_PATH/' ||
regexp_replace(lfrec.member,'^.*/','') || '''');
        end if;
        firstline:=false;
    end loop;
    dbms_output.put_line(CHR(13));
    dbms_output.put_line(CHR(13));
    dbms_output.put_line('rman duplication script:');
    dbms_output.put_line(CHR(13));
    dbms_output.put_line('run');
    dbms_output.put_line('{');
    for dfrec in df loop
        dbms_output.put_line('set newname for datafile ' ||
            dfrec.file# || ' to ''' || dfrec.name ||''';');
    end loop;
    for tfrec in tf loop
        dbms_output.put_line('set newname for tempfile ' ||
            tfrec.file# || ' to ''' || tfrec.name ||''';');
    end loop;
    dbms_output.put_line('duplicate target database for standby backup
location INSERT_PATH_HERE;');
    dbms_output.put_line('}');
end;
/
```

Replay logs on database

This script accepts a single argument of an Oracle SID for a database that is in mount mode and attempts to replay all currently available archive logs.

```
#! /usr/bin/perl
use strict;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
84 Migration of Oracle Databases to NetApp Storage Systems © 2021 NetApp,
Inc. All rights reserved
my $uid = $<;
my @out;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover database until cancel;
auto
EOF2
'
`;}
else {
@out=`. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover database until cancel;
auto
EOF2
`;
}
print @out;
```

Replay logs on standby database

This script is identical to the preceding script, except that it is designed for a standby database.

```
#! /usr/bin/perl
use strict;
my $oraclesid=$ARGV[0];
my $oracleuser='oracle';
my $uid = $<;
my @out;
if ($uid == 0) {
@out=`su - $oracleuser -c '. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover standby database until cancel;
auto
EOF2
'
`;}
else {
@out=`. oraenv << EOF1
$oraclesid
EOF1
sqlplus / as sysdba << EOF2
recover standby database until cancel;
auto
EOF2
`;
}
print @out;
```

Copyright information

Trademark information