

CHAPTER 1

Data Guard Architecture



Human error, hardware failures, software and network failures, and large-scale events such as fires, hurricanes, and earthquakes all jeopardize the availability of databases that are the lifeblood of business applications. The impact to operations when critical databases are unavailable is so obvious that few people need to be convinced of the importance of data protection and availability.

As an Oracle user, you have already done your homework on Oracle Data Guard. You know that Data Guard is purpose-built for protecting Oracle data, offering the highest levels of data protection and availability while still maintaining the best performance for your Oracle database. You know that, as a native capability built into the Oracle kernel, Data Guard's integration with other Oracle High Availability technologies—most notably Oracle Real Application Clusters (RAC), Oracle Recovery Manager (RMAN), and Oracle Flashback Technologies—offers many benefits. You also know that your finance department will be happy that Active Data Guard standby databases will not consume your IT budget on systems, storage, and software that sit idle until a failure occurs. And because there is no such thing as one-size-fits-all, you know that Data Guard offers the flexibility you need to address a wide range of requirements.

On the flip side of things, “comprehensive and flexible” means that you have a number of decisions to make. You might not be sure about the best way to deploy Data Guard for your environment, and while you have read the Oracle documentation, you may find that you still don't completely understand how Data Guard works. You need more insight into the trade-offs inherent in the different configuration options that Data Guard offers and what you need to know to manage a Data Guard configuration. The good news is that you are reading this book. We will provide you with a broader and deeper understanding of Data Guard that will ensure your success.

Data Guard Overview

Data Guard operates on a simple principle: ship redo, and then apply redo. Redo includes all of the information needed by the Oracle Database to recover a database transaction. A production database, referred to as the *primary database*, transmits redo to one or more independent replicas referred to as *standby databases*. Data Guard standby databases are in a continuous state of recovery, validating and applying redo to maintain synchronization with the primary database. Data Guard will also automatically resynchronize a standby database that becomes temporarily disconnected from its primary database because of a network or standby outage. This simple architecture makes it possible to have one or more synchronized replicas immediately available to resume processing in the event of a planned or unplanned outage of the primary database. A high-level overview of the Data Guard transport and apply architecture is provided in Figure 1-1.

What Is Redo?

Redo is at the center of everything Data Guard does. While Chapter 3 provides more details on redo concepts, a basic knowledge of this feature is fundamental to your understanding of how Data Guard works.

Data Guard vs. Remote Mirroring: Advantage Data Guard

Data Guard transmits only *redo data*—the information needed to recover a database transaction—to synchronize a standby database with its primary. Data Guard also prevents the primary from propagating corruption by performing Oracle validation before applying changes to a standby database. Before Data Guard became available, companies would use storage or host-based remote mirroring to maintain a synchronized copy of their Oracle database files. Unfortunately, remote mirroring does not have any knowledge of an Oracle transaction; thus it can't distinguish between redo, undo, data block changes, or control file writes. This requires remote mirroring to transmit every write to every file, generating 7 times the network volume and 27 times more network I/O operations than Data Guard.¹ Remote mirroring is also unable to perform Oracle validation, making it impossible to provide the same level of protection as Data Guard. For these reasons and others discussed later in this chapter, Data Guard has become the preferred data availability and protection solution for the Oracle Database.

Primary database transactions generate redo records. Oracle documentation defines a redo record as follows:²

A redo record, also called a redo entry, is made up of a group of change vectors, each of which is a description of a change made to a single block in the database. For example, if you change a salary value in an employee table, you generate a redo record containing change vectors that describe changes to the data segment block for the table, the undo segment data block, and the transaction table of the undo segments.

Redo records contain all the information needed to reconstruct changes made to the database. During media recovery, the database will read change vectors in the redo records and apply the changes to the relevant blocks.

Redo records are buffered in a circular fashion in the redo log buffer of the System Global Area (SGA). The log writer process (LGWR) is the database background process responsible for redo log buffer management. At specific times, the LGWR writes redo entries to a sequential file—the online redo log file (ORL)—to free space in the redo log buffer for new entries. The LGWR always writes all redo entries that have been copied into the redo log buffer since the last time it wrote. The LGWR writes the following:

- **A commit record** Whenever a transaction is committed, the LGWR writes the transaction redo records from the redo log buffer to an ORL and assigns a system change number (SCN) to identify the redo records for each committed transaction. Only when all redo records associated with a given transaction have been written to the ORL is the user process notified that the transaction has been committed.

¹“Oracle Data Guard and Remote Mirroring Solutions,” Oracle Technology Network: www.oracle.com/technology/deploy/availability/htdocs/DataGuardRemoteMirroring.html

²Oracle Database Administrator's Guide 11g Release 1 (11.1)

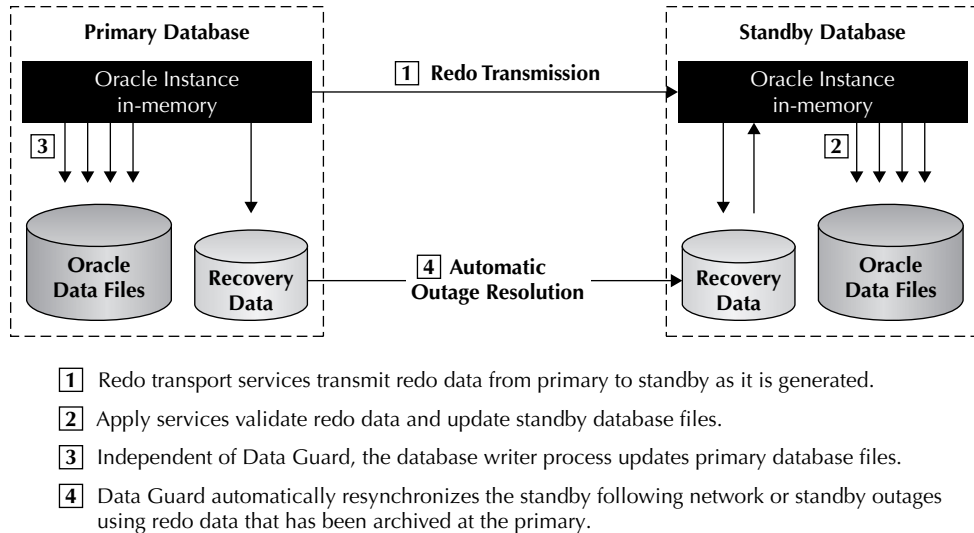


FIGURE 1-1. Overview: Data Guard redo transport and apply

- Redo log buffers** If the redo log buffer becomes a third full or if 3 seconds have passed since the last time the LGWR wrote to the ORL, all redo entries in the log buffer will be written to the ORL. This means that redo records can be written to an ORL before the corresponding transaction has been committed. If necessary, media recovery will roll back these changes using the undo that is also part of the redo entry. The LGWR will also write all redo records to the ORL if the database writer process (DBWn) writes modified buffers to disk and the LGWR had not already completed writing all of the redo records associated with the modified buffers.

It is worth noting that in times of high activity, the LGWR can write to the ORL using “group” commits. For example, assume a user commits a transaction. While the LGWR is writing the commit record to disk, other users may also be issuing COMMIT statements. However, the LGWR cannot write to the redo log file to commit these transactions until it completes the previous write operation. After the first transaction’s entries are written to the redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than if each transaction entry were handled individually. (The LGWR always does sequential writes—the larger the write, the more efficient it is.) If requests to commit continue at a high rate, every LGWR write from the redo log buffer will contain multiple commit records. This impacts what is referred to as *redo-write size*, one of the factors that influence database performance in a Data Guard synchronous configuration, which is discussed later in this chapter and in Chapter 2.

While the LGWR is going about its business making sure that transactions are recoverable, changes to data blocks in the primary database are deferred until it is more efficient for the DBWn to flush changes in the buffer cache to disk. The LGWR’s write of the redo entry containing the transaction’s commit record is the single event that determines that the transaction has been

committed. Oracle Database is able to issue a success code to the committing transaction, even though the DBWn has not yet flushed data buffers to disk. This enables high performance while guaranteeing that transactions are not lost if the primary database crashes before all data blocks have been written to disk.

Everything discussed in this section is normal processing for any Oracle database, whether or not Data Guard is in use. As transactions commit, they generate redo. This is where a detailed discussion of Data Guard can begin.

Redo Transport Services

Data Guard Redo Transport Services coordinate the transmission of redo from a primary database to the standby database. At the same time that the primary database LGWR process is writing redo to its ORL, a separate Data Guard process called the *Log Network Server (LNS)* is reading from the redo buffer in SGA and passes redo to Oracle Net Services for transmission to the standby database.

Data Guard's flexible architecture allows a primary database to transmit redo directly to a maximum of nine standby databases. Data Guard is also well integrated with Oracle RAC. An Oracle RAC database has two or more servers (nodes), each running its own Oracle instance and all having shared access to the same Oracle database. Either the primary, or standby, or both can be an Oracle RAC database. Each primary instance that is active generates its own thread of redo and has its own LNS process to transmit redo to the standby database.

Redo records transmitted by the LNS are received at the standby database by another Data Guard process called the *Remote File Server (RFS)*. The RFS receives the redo at the standby database and writes it to a sequential file called a *standby redo log file (SRL)*. In a multi-standby configuration, the primary database has a separate LNS process that manages redo transmissions for each standby database. In a configuration with three standby databases, for example, three LNS processes are active on each primary database instance.

Data Guard supports two redo transport methods using the LNS process: synchronous or asynchronous. A high-level overview of the redo transport architecture is provided in Figure 1-2.

Synchronous Redo Transport

Synchronous transport (SYNC) is also referred to as a “zero data loss” method because the LGWR is not allowed to acknowledge a commit has succeeded until the LNS can confirm that the redo needed to recover the transaction has been written to disk at the standby site. SYNC is described

Myth Buster: LGWR Transmits Redo to Standby Databases

A common misconception is that the LGWR is the process that transmits data to a standby database. This is *not* the case. The Data Guard LNS process manages all synchronous and asynchronous redo transmissions. Eliminating this perception is the reason why the Data Guard 11g documentation simply refers to the redo transport methods as SYNC or ASYNC, rather than LGWR SYNC or LGWR ASYNC as was done in previous releases.

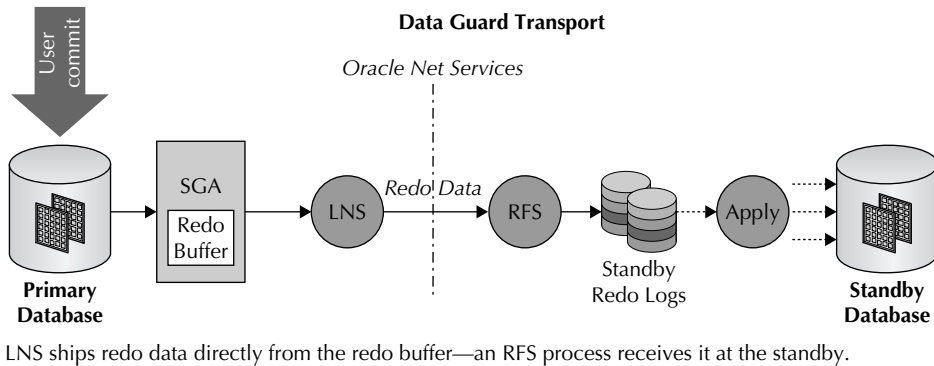


FIGURE 1-2. Data Guard redo transport process architecture

in detail in Figure 1-3. The numbered list that follows outlines each phase of SYNC redo transport and corresponds to the numbers shown in Figure 1-3.

1. The user commits a transaction creating a redo record in SGA. The LGWR reads the redo record from the log buffer, writes it to the online redo log file, and waits for confirmation from the LNS.
2. The LNS reads the same redo record from the log buffer and transmits it to the standby database using Oracle Net Services. The RFS receives the redo at the standby database and writes it to a standby redo log file.

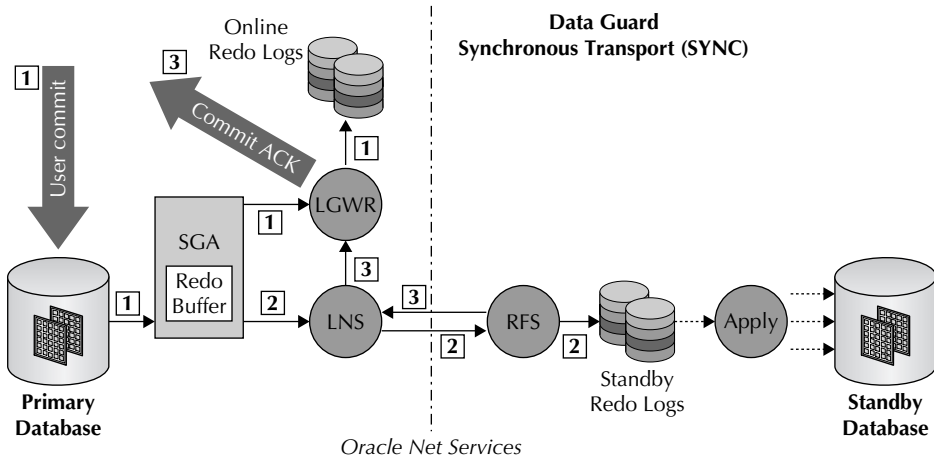


FIGURE 1-3. SYNC redo transport architecture

3. When the RFS receives a write-complete from the disk, it transmits an acknowledgment back to the LNS process on the primary database, which in turn notifies the LGWR that transmission is complete. The LGWR then sends a commit acknowledgment to the user.

While SYNC guarantees protection for every transaction that the database acknowledges as having been committed, this guarantee can also impact primary database performance. The cause of the performance impact is obvious: the LGWR must wait for confirmation that data is protected at the standby before it can proceed with the next transaction. The degree of impact this has on application response time and database throughput is a function of several factors: the redo-write size, available network bandwidth, round-trip network latency (RTT), and standby I/O performance writing to the SRL. Because network RTT increases with distance, so will the performance impact on your primary database, imposing a practical limit on how far apart you will be able to locate your primary and standby databases. The cumulative impact of these factors can be seen in the wait event “LNS wait on SENDREQ,” found in the `V$SYSTEM_EVENT` dynamic performance view (optimizing redo transport is discussed in Chapter 2).

Having read this, you are probably wondering what happens to the primary database if the network or standby database fails while using SYNC? Will the primary database wait forever for an acknowledgment that will never come? Please hold that thought until the “Data Guard Protection Modes” section and the discussion of the `NET_TIMEOUT` attribute, later in this chapter.

Asynchronous Redo Transport

Asynchronous transport (ASYNC) is different from SYNC in that it eliminates the requirement that the LGWR wait for acknowledgment from the LNS, creating near zero performance impact on the primary database regardless of the distance between primary and standby locations.

The LGWR will continue to acknowledge commit success even if limited bandwidth prevents the redo of previous transactions from being sent to the standby database immediately (picture a sink filling with water faster than it can drain). If the LNS is unable to keep pace and the log buffer is recycled before the redo can be transmitted to the standby, the LNS automatically transitions to reading and sending from the ORL (Data Guard 11g onward). Once the LNS is caught up, it automatically transitions back to reading/sending directly from the log buffer.

If ASYNC redo transport falls behind to the degree that the LNS is still in the ORL at log switch time, LNS will continue until it completes sending the contents of the original ORL. Once complete, it seamlessly transitions back to reading/sending from the current online log file.

Data Guard 11g ASYNC Enhancements

ASYNC behavior has varied over previous Data Guard releases. The LNS process in Data Guard 11g ASYNC now reads directly from the redo log buffer, but unlike pre-10.2 releases, there is never a “buffer full” state that can cause transmission to terminate. Instead, the LNS process seamlessly transitions to read and send from the online redo log of the primary database. Data Guard 11g ASYNC is also more efficient in how it utilizes available network bandwidth, increasing the network throughput rate that can be achieved for any given bandwidth. The higher the network latency, the greater the gain in network throughput compared to previous Data Guard releases.

Optimizing ASYNC Redo Transport

The log buffer hit ratio is tracked in the view `X$logbuf_readhist`. A low hit ratio indicates that the LNS is frequently reading from the ORL instead of the log buffer. If there are periods when redo transport is coming close, but is not quite keeping pace with your redo generation rate, consider increasing the log buffer size in Data Guard 11g to achieve a more favorable hit ratio. This will reduce or eliminate I/O overhead of the LNS reading from the ORL. See Chapter 2 for more details.

When the LNS catches up with the LGWR, it seamlessly transitions back to reading/sending from the redo log buffer.

In the rarer case in which there are two or more log switches before the LNS has completed sending the original ORL, the LNS will still transition back to reading the contents of the current online log file. Any ORLs that were archived between the original ORL and the current ORL are transmitted via Data Guard's *gap resolution process* described in the section "Automatic Gap Resolution" a little later in the chapter. Note that if you find that this "rare case" is a frequent occurrence, it is most likely a sign that you have not provisioned enough bandwidth to transport your redo volume.

The behavior of ASYNC transport enables the primary database to buffer a large amount of redo, called a *transport lag*, without terminating transmission or impacting availability. While the I/O overhead related to the ASYNC LNS reading from the ORL can marginally impact primary database performance, this is insignificant compared to the potential performance impact of SYNC on a high latency network. The relative simplicity of ASYNC is evident when comparing Figures 1-4 and 1-3. The only drawback of ASYNC is the increased potential for data loss. If a failure destroys the primary database before any transport lag is reduced to zero, any committed transactions that are a part of the transport lag will be lost. Provisioning enough network bandwidth to handle peak redo generation rates when using ASYNC will minimize this potential for data loss.

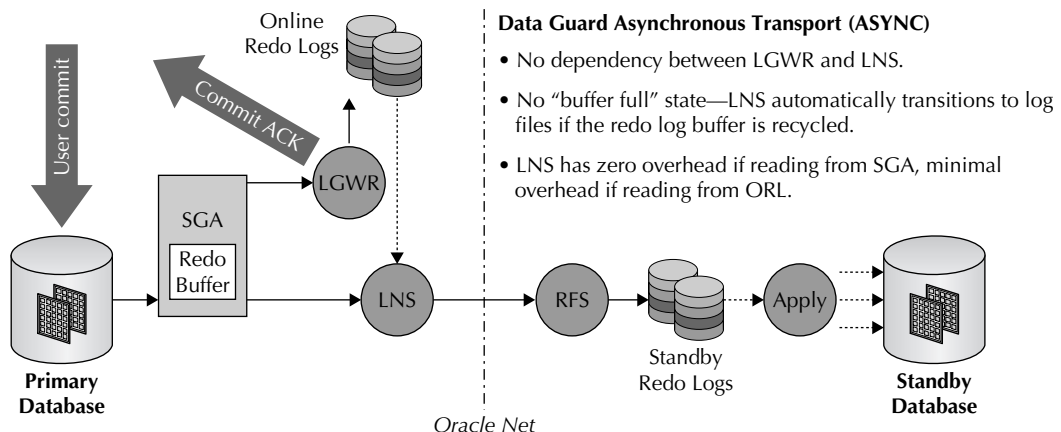


FIGURE 1-4. ASYNC redo transport architecture

Enabling ASYNC Redo Transport Compression

Buried in Oracle MetaLink Note 729551.1 is the information needed to enable redo transport compression for Oracle Database 11g Release 1 and Data Guard ASYNC (Maximum Performance) using the parameter `_REDO_TRANSPORT_COMPRESS_ALL`. A license for Oracle Advanced Compression is required to enable redo transport compression.

Redo Transport Compression

An additional consideration when using ASYNC is determining whether it is advantageous to compress redo to reduce your bandwidth requirements. Oracle released a new product for Oracle Enterprise Edition 11g called the *Advanced Compression option*. This new product contains several compression features, one of which is redo transport compression for Data Guard. Initially this feature could only be enabled when Data Guard was transmitting log files needed to resolve an archive log gap. However, in response to customer request, Oracle has published information about an undocumented parameter that enables compression for ASYNC redo transport as well. (See sidebar, “Enabling ASYNC Redo Transport Compression.”)

ASYNC redo transport compression will increase CPU utilization; however, in bandwidth-constrained environments it can make the difference between success and failure in accomplishing your recovery point (data loss) objectives. For example, Oracle Japan and Hitachi Ltd. tested the impact of using compression in a bandwidth-constrained environment with a test workload that generated 20 MB/sec of redo. While compression ratios will vary from one workload to the next, the compression ratio achieved in the test was 60 percent. The benefit of using compression was significant, making it possible to sustain a transport lag of less than 10 seconds and achieve recovery point objectives.³ This compared very favorably to baseline test runs without compression, in which transmission could not keep pace with primary redo generation, resulting in a transport lag that continued to increase linearly over time for the duration of the test. The testing also showed that as long as sufficient CPU resources were available for compression, minimal impact was experienced on database throughput or response time.

Automatic Gap Resolution

A log file gap occurs whenever a primary database continues to commit transactions while the LNS process has ceased transmitting redo to the standby database. This can occur whenever the network or the standby database is down, depending on how you have chosen to implement your Data Guard configuration (discussed in the section “Data Guard Protection Modes” later in this chapter). While in this state, the primary database LGWR process continues writing to the current ORL, fills it, and then switches to a new ORL while an archive (ARCH) process archives the completed ORL locally. This cycle can repeat itself many times over on a busy system before the connection between the primary and standby is restored, creating a large log file gap.

³“Batch Processing in Disaster Recovery Configurations: Best Practices for Oracle Data Guard,” validation report on Data Guard redo transport compression and proper network configuration by Hitachi Ltd./Oracle Japan GRID Center: www.hitachi.co.jp/Prod/comp/soft1/oracle/pdf/OBtecinfo-08-008.pdf

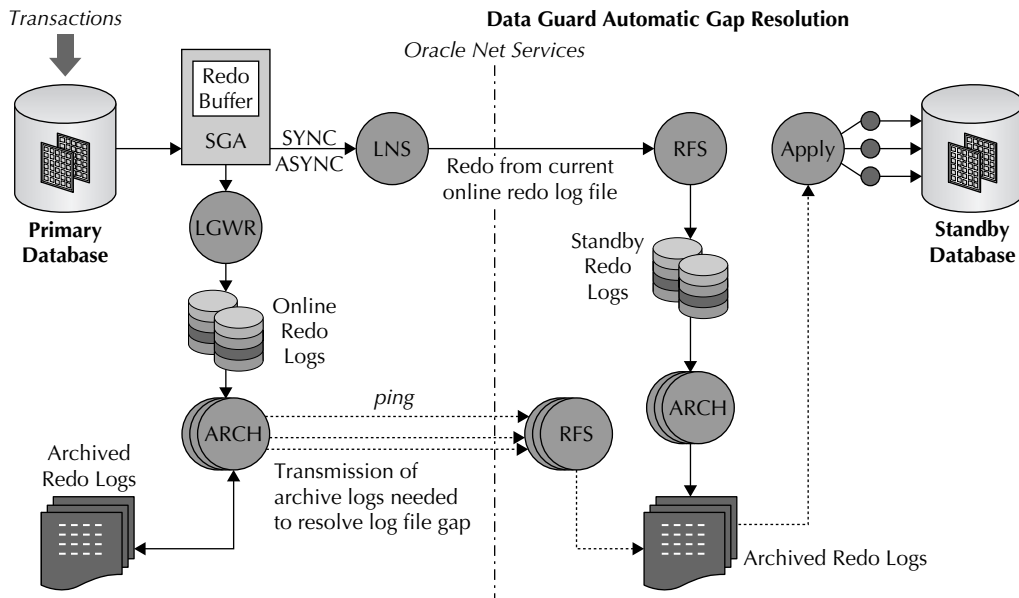


FIGURE 1-5. Automatic gap resolution

Data Guard uses an ARCH process on the primary database to continuously ping the standby database during the outage to determine its status. When communication with the standby is restored, the ARCH ping process queries the standby control file (via its RFS process) to determine the last complete log file that the standby received from the primary database. Data Guard determines which log files are required to resynchronize the standby database and immediately begins transmitting them using additional ARCH processes. At the very next log switch, the LNS will attempt and succeed in making a connection to the standby database and will begin transmitting current redo while the ARCH processes resolve the gap in the background. The dashed lines in Figure 1-5 portray the transmission and apply of redo needed to resolve the log file gap. Once the standby apply process is able to catch up to current redo records, the apply process automatically transitions out of reading from archived redo logs, and into reading from the current SRL (assuming the user has configured Data Guard *real-time apply*). One last side note: beginning with Data Guard 10g, one ARCH process at the primary database is always dedicated to local archival to ensure that remote archival during gap resolution does not impact the ability of the primary to recycle its ORLs.⁴

The performance of automatic gap resolution is critical. The longer the primary and standby databases remain unsynchronized, the greater the risk of data loss should a failure occur. The primary must be able to transmit data at a much faster pace than its normal redo generation rate if the standby is to have any hope of catching up. The Data Guard architecture enables gaps to be resolved quickly using multiple background ARCH processes, while at the same time the LNS process is conducting normal SYNC or ASYNC transmission of the current log stream.

⁴This functionality is available in Oracle9i Data Guard starting at version 9.2.0.5. See MetaLink Note 260040.1.

Why Isn't ARCH Redo Transport in the Data Guard 11g Documentation?

Three redo transport methods were documented prior to Data Guard 11g: SYNC, ASYNC, and ARCH. *ARCH* refers to traditional archive log shipping, in which Data Guard would wait for an ORL to be archived before the contents of the resulting archive log file were shipped by an ARCH process. Data Guard 11g ASYNC performance enhancements have led Oracle to deprecate ARCH as a documented redo transport method. Though deprecated, the functionality still exists to use ARCH for redo transport and provide backward compatibility for previous customer installations. The ARCH transport infrastructure also continues to be used transparently by Data Guard 11g when automatically resolving archive log gaps between primary and standby databases.

Apply Services

Data Guard offers two different methods to apply redo to a standby database: Redo Apply (physical standby) and SQL Apply (logical standby). We will describe the differences in a moment, but first let's discuss key objectives that Redo Apply and SQL Apply have in common.

The primary goal of Data Guard is to protect against data loss; thus its first design objective is that the standby database be a synchronized copy of the primary database. Data Guard is designed from the ground up for simple one-way replication of the entire database. Data Guard also has built-in safeguards that prevent any unauthorized modifications from being made at the standby database to data it has replicated from the primary database. These characteristics explain the fundamental difference between Data Guard and Oracle's full-featured replication product, Oracle Streams. Oracle Streams offers various methods for granular, *n*-way replication and transformation of subsets of an Oracle database. By definition, the additional functionality of Oracle Streams means that it has more moving parts with the usual implications for performance and management complexity. Data Guard has been designed for a simpler mission, and this is reflected in the relative simplicity of implementing and managing a Data Guard configuration.

The second objective for Data Guard is to provide a high degree of isolation between primary and standby databases. This prevents problems that occur at the primary database from impacting the standby database and compromising data protection and availability. This also prevents problems that occur at the standby from impacting the availability or performance of the primary database. For example, Data Guard apply processes validate redo before it is applied to the standby database, preventing physical corruptions that can occur at the primary database from being propagated to the standby database. Also, consider for a moment the earlier discussion of redo transport services. Nowhere is there a dependency between redo transport and standby database apply. Primary database availability, performance, and its ability to transmit redo to the standby database are not impacted by how standby apply is configured, or the performance of the apply process, or even whether apply is on or off.

The third objective for Data Guard is to provide data availability and high availability should the primary database fail. Redo Apply and SQL Apply have the same capabilities to transition a synchronized standby database quickly to the primary role. This protects data and restores availability following planned or unplanned outages of the primary database.

Data Guard Apply and Oracle RAC

Each primary Oracle RAC instance ships its own thread of redo that is merged by the Data Guard apply process at the standby and applied in SCN order to the standby database (see Chapter 8 for a more detailed explanation). If the standby is an Oracle RAC database, only one instance (the apply instance) can merge and apply changes to the standby database. Should the apply instance fail for any reason, the apply process can automatically failover to a surviving instance in the Oracle RAC standby database when using the Data Guard broker, discussed in Chapter 5.

The final objective for Data Guard is to deliver a high return on investment in standby systems, storage, and software, without compromising its core mission of data protection and availability. Both Redo Apply and SQL Apply enable the productive use of standby databases while in a standby role, without impacting data protection or the ability to achieve recovery time objectives (RTO).

Now that you know what Redo Apply and SQL Apply have in common, you need to understand the differences between the two to determine which type of standby database is best suited to your requirements. An overview of the unique characteristics and benefits of Redo Apply and SQL Apply are discussed next. Additional details are provided in Chapters 2, 3, and 4.

Redo Apply (Physical Standby)

Redo Apply maintains a standby database that is an exact, block-by-block, physical replica of the primary database. As the RFS process on the standby receives primary redo and writes it to an SRL, Redo Apply uses Media Recovery to read redo records from the SRL into memory and apply change vectors directly to the standby database. Media Recovery does parallel media recovery (Figure 1-6) for very high performance. It comprises a Media Recovery Coordinator and multiple parallel apply processes. The Media Recovery Coordinator (MRP0) manages the recovery session, merges redo by SCN from multiple instances (if Oracle RAC primary), and then parses redo into

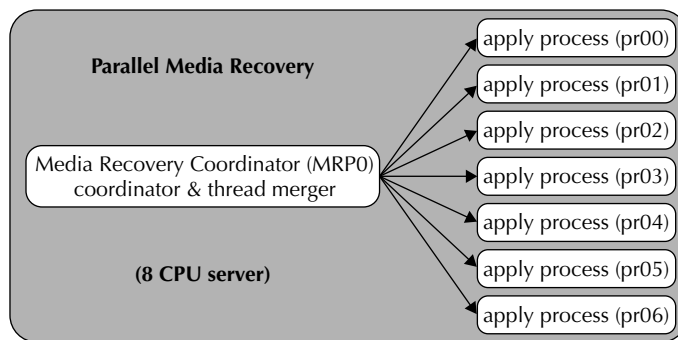


FIGURE 1-6. Parallel media recovery for Redo Apply (physical standby)

change mappings partitioned by apply process. The apply processes (pr00, 01, 02...) read data blocks, assemble redo changes from mappings, and then apply redo changes to data blocks. Redo Apply automatically configures a number of apply processes equal to the number of CPUs in the standby system minus one. This architecture, along with significant Media Recovery enhancements in Oracle Database 11g, achieves very high performance. Oracle has benchmarked Data Guard Redo Apply rates up to 47 MB/sec for an online transaction processing (OLTP) workload and 112 MB/sec for a direct path load.⁵

Oracle Active Data Guard 11g

The usefulness of a physical standby database while in the standby role was significantly enhanced by the Active Data Guard Option for Oracle Database 11g Enterprise Edition. In previous Data Guard releases, the database would have to be in the mount state when media recovery was active. Media recovery has always been optimized for the highest possible performance and was never designed to present queries with a read-consistent view while enabled. Querying a physical standby database has required disabling media recovery and opening the standby database in read-only mode. Since standby data can quickly become stale once media recovery is disabled, the usefulness of a physical standby to offload read-only queries and reporting from a primary database was limited.

Active Data Guard 11g solves the read consistency problem without impacting standby apply performance by use of a “query” SCN. The media recovery process on the standby database advances the query SCN after all dependent changes in a transaction have been fully applied (the new query SCN is also propagated to all instances in an Oracle RAC standby). The query SCN is exposed to the user as the `CURRENT_SCN` column of the `V$DATABASE` view on the standby database. Read-only users will only see data up to the query SCN, guaranteeing the same read consistency as the primary database. This enables a physical standby database to be open read-only while media recovery is active, making it very useful for offloading read-only workloads from the primary database.

Corruption Protection

Data Guard Redo Apply provides superior data protection by preventing physical corruptions that can occur at the primary database from being applied to a standby database. Redo transmitted directly from SGA by SYNC or ASYNC is completely isolated from physical I/O corruptions

Remote Mirroring and Corruption

We frequently hear reports from users of Storage Area Network (SAN) or host-based remote mirroring of cases in which physical corruptions caused by component failure at their primary site were mirrored to remote volumes, making both copies unusable. Since Oracle cannot be mounted on remote volumes while the mirroring session is active, it cannot perform end-to-end validation of changes before they are applied to the standby database. Worse yet, remote mirroring users often do not learn that a problem exists until they need their standby database—and at that point it’s too late. Data Guard does not have these limitations.

⁵Active Data Guard 11g and media recovery best practices: www.oracle.com/technology/deploy/availability/pdf/maa_wp_11gr1_activedataguard.pdf

caused by component failures at the primary site. The software code-path executed by Redo Apply on a standby database is also fundamentally different from that of a primary—providing the standby database an additional level of isolation from software errors that can impact the primary database. Data Guard uses Oracle processes to validate redo before it is applied to the standby database. Corruption-detection checks occur at the following key interfaces:

- **On the primary database during Redo Transport** `LGWR`, `LNS`, `ARCH` On an Oracle Database 11g primary database, corruption detection/protection is best enabled using the parameter `DB_ULTRA_SAFE`.
- **On the standby database during Redo Apply** `RFS`, `ARCH`, `MRP`, `DBWR` On an Oracle Database 11g standby database, corruption detection/prevention is best enabled using the parameters `DB_BLOCK_CHECKSUM=FULL` and `DB_LOST_WRITE_PROTECT=TYPICAL`.

If Redo Apply detects any corrupt redo at the standby database, Data Guard will automatically fetch new copies of the relevant archive logs from the primary database using the gap resolution process in the hope that the originals are free of corruption.

Physical Standby utilizes the new Oracle Database 11g parameter, `DB_LOST_WRITE_PROTECT`, to provide industry-unique protection against corruptions caused by lost writes. A *lost write* occurs when an I/O subsystem acknowledges the completion of a write, while in fact the write did not occur in persistent storage. On a subsequent block read the I/O subsystem returns the stale version of the data block that is used to update other blocks, spreading corruptions across the database. When the `DB_LOST_WRITE_PROTECT` initialization parameter is set, the database records buffer cache block reads in the redo log, and this information is used to detect lost writes. Meaningful protection using lost write detection requires the use of a Data Guard physical standby database. You set `DB_LOST_WRITE_PROTECT` to `TYPICAL` in both primary and standby databases (setting `DB_ULTRA_SAFE` at the primary as noted above will automatically set `DB_LOST_WRITE_PROTECT=TYPICAL` on the primary database). When the standby database applies redo using Redo Apply, it reads the corresponding blocks and compares the SCNs with the SCNs in the redo log. If the comparison shows:

- The block SCN on the primary database is lower than the block SCN on the standby database, then a lost write has occurred on the primary database and an external error (ORA-752) is signaled. The recommended procedure in response to an ORA-752 is to execute a failover to the physical standby and re-create the primary database.
- The block SCN is higher, then a lost write has occurred on the standby database, and an internal error (ORA-600 3020) is signaled. If possible, you can fix the standby using a backup from the primary database of the affected data files. Otherwise, you will have to rebuild the standby completely.

Redo Apply Benefits

Physical standby databases maintained using Redo Apply are generally the best choice for disaster recovery (DR) based upon their simplicity, transparency, high performance, and superior data protection. In summary, the advantages of a physical standby database include the following:

- Complete application and data transparency—no data type or other restrictions.
- Very high performance, least management complexity, and fewest moving parts.

Rolling Database Upgrades Using a Physical Standby

Data Guard 11g enables a physical standby database to be used for rolling database upgrades via the `KEEP IDENTITY` clause and SQL Apply. A physical standby is temporarily converted to a *transient logical standby* and upgraded to the new release. Although the process of upgrading the Oracle Home must be performed on both the primary and standby systems, the execution of the database upgrade script only needs to be performed once on the transient logical standby database. Following a switchover, the original primary database is converted back into a physical standby and is upgraded by applying the redo generated by the execution of the upgrade script previously run on the transient logical standby (see Chapter 11 for details). This eliminates the extra cost and effort of deploying additional storage for a logical standby database solely for the purpose of a rolling database upgrade.

- Oracle end-to-end validation before apply provides the best protection against physical corruptions, including corruptions due to lost writes.
- Able to be utilized for up-to-date read-only queries and reporting while providing DR (Active Data Guard 11g).
- Able to offload backups from the primary database while providing DR.
- Able to support QA testing and other activities requiring read-write access, while continuing to provide DR protection for primary data (Data Guard 11g Snapshot Standby).
- Able to execute rolling database upgrades beginning with Oracle Database 11g (Transient Logical)

SQL Apply (Logical Standby)

SQL Apply uses the Logical Standby Process (LSP) to coordinate the apply of changes to the standby database. SQL Apply requires more processing than Redo Apply, as can be seen in Figure 1-7 and discussed in detail in Chapter 4. The processes that make up SQL Apply read the SRL and “mine” the redo by converting it to logical change records, and then building SQL transactions and applying SQL to the standby database. Because the process of reconstruction and replaying workload has more moving parts, it requires more memory, CPU, and I/O than Redo Apply.

SQL Apply also does not provide the same level of transparency as Redo Apply. SQL Apply performance can vary from one transaction profile to the next. SQL Apply does not support all data types (such as XML in object relational format, and Oracle supplied types such as Oracle Spatial, Oracle Intermedia, and Oracle Text). Collectively, these attributes result in SQL Apply requiring more extensive performance testing, tuning, and management effort than a physical standby database. (Refer to Oracle MetaLink for an excellent note that provides insight into optimizing SQL Apply performance.⁶) While such characteristics are found to varying degrees in any SQL-based replication solution, whether provided by Oracle or by third parties, SQL Apply

⁶MetaLink Note 603361.1: “Developer and DBA Tips for Pro-Actively Optimizing SQL Apply”

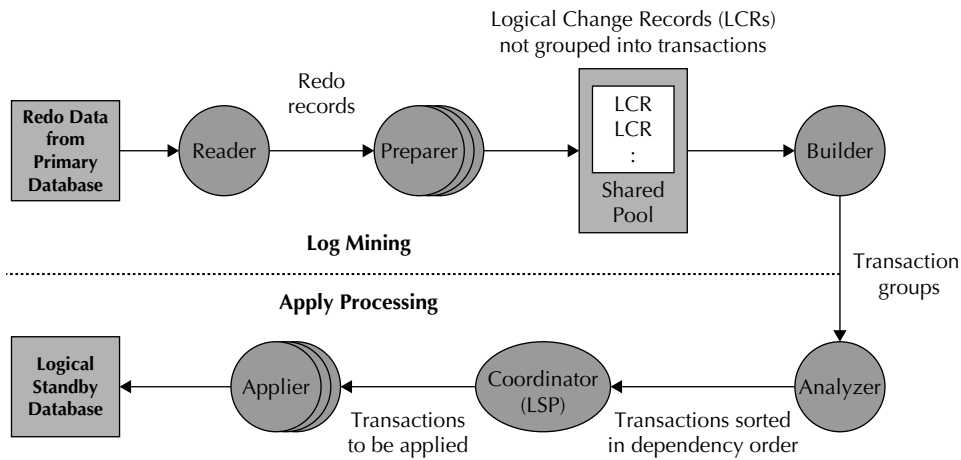


FIGURE 1-7. SQL Apply process architecture

has an inherent advantage over third-party SQL replication products due to its native integration with the Oracle Database kernel.

SQL Apply Benefits

The extra processing performed by SQL Apply is also the source of its advantages when compared to Redo Apply. Because SQL Apply applies SQL, a logical standby database is opened read-write while apply is active. While SQL Apply prevents any modifications from being made to the data it is replicating, a logical standby database has the additional flexibility of allowing inserts, updates, and deletes to local tables and schemas that have been added to the standby database independent of the primary. This is very useful, for example, if you want to use the standby to offload a reporting application from the primary database that must make frequent writes to global temporary tables or other local tables that exist only at the standby database. A logical standby database also allows the creation of local indexes and materialized views that don't exist on the primary database. This enables indexes that can be quite expensive to maintain, in terms of their impact on an OLTP system, to be implemented on a logical standby database where they are valuable for optimizing reporting and browsing activities. SQL Apply benefits include the following:

- A native Oracle capability that is simpler and less intrusive on primary database performance and administration than third-party SQL-based replication products. This is accomplished by having a simpler design objective of one-way replication for the entire primary database. (Redo Transport Services efficiently transmit all primary database redo, and SQL Apply always performs all of its processing at the standby database.)
- A standby database that is opened read-write while SQL Apply is active.
- A “guard” setting that prevents applications from modifying data in the standby database that is being maintained by SQL Apply.

Myth Buster: SQL Apply Is an Immature Feature

SQL Apply WAS an immature feature when first released in Oracle9i, leading early users to believe that SQL Apply could not be used successfully in a production environment. This perception is now a myth as SQL Apply has matured over several major Oracle releases. This statement is substantiated by the growing number of successful production implementations using Data Guard 10g Release 2. Data Guard 11g SQL Apply is a very attractive solution for the requirements it is designed to address.

- SQL Apply can be used for rolling database upgrades to new Oracle releases and patchsets, beginning with Oracle Database 10.1.0.4 for logical standby databases, and beginning with Oracle Database 11.1.0.6 for physical standby databases (using the `KEEP IDENTITY` clause).

We recommend using SQL Apply if you can satisfy its prerequisites and you have the additional requirement for a standby database that is open read-write while it provides DR protection for the primary database.

Can't Decide? Then Use Both!

We know that making a choice between Redo Apply and SQL Apply can create a dilemma. You want the simplicity and performance of Redo Apply for data protection and availability. Redo Apply when using Active Data Guard 11g also offers an excellent solution for offloading read-only queries from your primary databases. However, you may have cases where a reporting application needs read-write access to the standby database, requiring the additional flexibility offered by SQL Apply. Data Guard support for multi-standby configurations having a mix of physical and logical standby databases can provide users with the flexibility to satisfy all requirements in an optimum fashion in a single Data Guard configuration.⁷

Myth Buster: Standby Apply Performance Can Impact the Primary Database

A common misperception is that standby apply performance can impact the primary database. This perception is perpetuated by the fact that competing RDBMS products do not deliver the same level of isolation implemented by Data Guard. Standby database apply performance does not have any impact on primary database availability or performance in a Data Guard configuration.

⁷“Managing Data Guard Configurations Having Multiple Standby Databases—MAA Best Practices”: www.oracle.com/technology/deploy/availability/pdf/maa10gr2multiplestandbybp.pdf

Data Guard Protection Modes

Many DBAs are interested in the superior data protection of Data Guard SYNC redo transport, but they are often concerned that the primary database may hang indefinitely if it does not receive acknowledgment from its standby database, due to the standby database being unavailable or a network down condition. The last thing that most DBAs want to report to their customers is that while the primary database is completely healthy, it is refusing to process any more transactions until it can guarantee that data is protected by a standby database. Then again, perhaps you have a different set of requirements and you must absolutely guarantee that data is protected even at the expense of primary database availability. Both of these use cases can utilize SYNC transport to provide zero data loss protection, but the two cases require a different response to a network or standby failure. Data Guard protection modes implement rules that govern how the configuration will respond to failures, enabling you to achieve your specific objectives for data protection, availability, and performance. Data Guard can support multiple standby databases in a single Data Guard configuration, and they may all have the same, or different, protection mode setting, depending on your requirements. The different Data Guard protection modes are Maximum Performance, Maximum Availability, and Maximum Protection.

Maximum Performance

This mode emphasizes primary database performance over data protection. It requires ASYNC redo transport so that the LGWR process never waits for acknowledgment from the standby database. Primary database performance and availability are not impacted by redo transport, by the status of the network connection between primary and standby, or by the availability of the standby database. As discussed earlier in this chapter, ASYNC enhancements in Data Guard 11g have made it the default redo transport method for Maximum Performance. Oracle no longer recommends the ARCH transport for Maximum Performance in Data Guard 11g given that it provides a lower level of data protection with no performance advantage compared to ASYNC.

Maximum Availability

This mode emphasizes availability as its first priority and zero data loss protection as a very close second priority. It requires SYNC redo transport, thus primary database performance may be impacted by the amount of time required to receive an acknowledgment from the standby that redo has been written to disk. SYNC transport, however, guarantees 100-percent data protection during normal operation in the event that the primary database fails.

However, events that have no impact on the availability of the primary database can impact its ability to transmit redo to the standby. For example, a network or standby database failure will make it impossible to transmit to the standby database, yet the primary database is still capable of accepting new transactions. A primary database configured for Maximum Availability will wait a maximum of `NET_TIMEOUT` seconds (a user configurable parameter which is discussed more completely in Chapter 2) before giving up on the standby destination and allowing primary database processing to proceed even though it can no longer communicate with the standby. This prevents a failure in communication between the primary and standby databases from impacting the availability of the primary database.

Data Guard will automatically resynchronize the standby database once the primary is able to re-establish a connection to the standby (utilizing the gap resolution process described earlier in this chapter). Specifically, once `NET_TIMEOUT` seconds expire, the LGWR process disconnects from the

LNS process, acknowledges the commit, and proceeds without the standby. Processing continues until the current ORL is complete and the LGWR cycles into a new ORL. As the new ORL is opened, the LGWR will terminate the previous LNS process, if necessary, and start a new LNS process that will attempt to make a new connection to the standby database. If it succeeds, the contents of the new ORL will be sent as usual. If the LNS does not succeed within `NET_TIMEOUT` seconds, the LGWR continues as before, acknowledges the current commit, and proceeds without the standby. This process is repeated at each log switch until LNS succeeds in making a connection to the standby database. (How soon the LGWR retries a failed standby can be tuned using the `REOPEN` attribute, which is discussed in Chapter 2.)

Meanwhile, the primary database has archived one or more ORLs that have not been completely transmitted to the standby database. A Data Guard ARCH process continuously pings the standby database until it can again make contact and determine which archive logs are incomplete or missing at the standby. With this knowledge in-hand, Data Guard immediately begins transmitting any log files needed to resynchronize the standby database. Once the ping process makes contact with the standby Data Guard will also force a log switch on the primary database. This closes off the current online log file and initiates a new LNS connection to immediately begin shipping current redo, preventing redo transport from falling any further behind while gap resynchronization is in progress. The potential for data loss during this process exists only if another failure impacts the primary database before the automatic resynchronization process is complete.

Maximum Protection

As its name implies, this mode places utmost priority on data protection. It also requires SYNC redo transport. The primary will not acknowledge a commit to the application unless it receives acknowledgment from at least one standby database in the configuration that the data needed to recover that transaction is safely on disk. It has the same impact on primary database performance as Maximum Availability, except that it does not consider the `NET_TIMEOUT` parameter. If the primary does not receive acknowledgment from a SYNC standby database, it will stall and eventually abort, preventing any unprotected commits from occurring. This behavior guarantees complete data protection even in the case of multiple failure events (for example, first the network drops, and later the primary site fails). Note that most users who implement Maximum Protection configure a minimum of two SYNC standby databases at different locations, so that failure of an individual standby database does not impact the availability of the primary database.

Role Management Services

Let's step back for a moment and review what we have covered thus far. Our review of Data Guard transport and apply services has shown the following:

- Data Guard only needs to transmit redo records to synchronize remote standby databases.
- Transmission can be either synchronous (zero data loss) or asynchronous.
- Synchronous transmission can impact primary database throughput and response time because of the time it takes for the primary to receive acknowledgment from the remote standby that data is safely written to disk. We can control how long a primary database will wait for that acknowledgment so that we do not fall into an indefinite hang if the primary loses its link to the standby.

- Asynchronous transmission will never cause the primary to stall or impact primary database performance or response time in a material way.
- There are two different types of standby databases: Redo Apply (physical) and SQL Apply (logical). We know their relative strengths, and we know that regardless of the method chosen, standby apply performance will never impact the availability or performance of the primary database. We know that all redo is validated by Oracle before it is applied to the standby database, preventing physical corruptions or lost writes that may occur on the primary database from impacting the standby database. We know that all Data Guard standby databases are active, able to be open for read-only queries and reports in order to offload work from a primary database and get more value from investments in standby systems.
- The Data Guard protection modes control how the configuration will respond to failures so that availability, performance, and data protection objectives are achieved. We know that the availability of the standby database or the network connection between primary and standby will never impact primary database availability unless explicitly configured to do so to achieve the highest possible level of data protection.

The next area of Data Guard architecture we will discuss is role management services that enable the rapid transition of a standby database to the primary database role. Data Guard documentation uses the term *switchover* to describe a planned role transition, usually for the purpose of minimizing downtime during planned maintenance activities. The term *failover* is used to describe a role transition in response to unplanned events.

Switchover

Switchover is a planned event in which Data Guard reverses the roles of the primary and a standby database. Switchover is particularly useful for minimizing downtime during planned maintenance. The most obvious case is when migrating to new Oracle Database releases or patchsets using a rolling database upgrade. A Data Guard switchover also minimizes downtime when migrating to new storage (including Exadata storage⁸), migrating volume managers (for example, moving to Oracle Automatic Storage Management), migrating from single instance to Oracle RAC, performing technology refresh, operating system or hardware maintenance, and even relocating data centers. The switchover command executes the following steps:

1. Notifies the primary database that a switchover is about to occur.
2. Disconnects all users from the primary.
3. Generates a special redo record that signals the End Of Redo (EOR).
4. Converts the primary database into a standby database.
5. Once the standby database applies the final EOR record, guaranteeing that no data has been lost, converts the standby to the primary role.

The new primary automatically begins transmitting redo to all other standby databases in the configuration. The transition in a multi-standby configuration is orderly because each standby

⁸MAA “Best Practices for Migrating to Oracle Exadata⁸MAA Server”: www.oracle.com/technology/products/bi/db/exadata/pdf/migration-to-exadata-whitepaper.pdf

received the identical EOR record transmitted the original primary, they know that the next redo received will be from the database that has just become the new primary database.

The basic principle for using switchover to reduce downtime during planned maintenance is usually the same. The primary database runs unaffected while you implement the required changes on your standby database (e.g. patchset upgrades, full Oracle version upgrades, etc). Once complete, production is switched over to the standby site running at the new release. In the case of a data center move, you simply create a standby database in the new data center and move production to that database using a switchover operation.

Alternatively, before performing maintenance that will impact the availability of the primary site, you can first switch production to the standby site so that applications remain available the entire time that site maintenance is being performed. Once the work is complete Data Guard will resynchronize both databases and enable you to switch production back to the original primary site. Regardless of how much time is required to perform planned maintenance, the only production database downtime is the time required to execute a switchover—a task that can be completed in less than 60 seconds as documented by Oracle best practices⁹, and in as fast as 5 seconds as documented in collaborative validation testing performed more recently by Oracle Japan and IBM.¹⁰

Switchover operations become even more valuable given Oracle's increasing support for different primary/standby systems in the same Data Guard configuration. For example, Oracle Database 11g can support a Windows primary and Linux standby, or a 32-bit Oracle primary and a 64-bit Oracle standby, and other select mixed configurations.¹¹ This makes it very easy to migrate between supported platform combinations with very little risk simply by creating a standby database on the new platform and then switching over. In most cases, you are able to minimize your risk even more by continuing to keep the old database on the previous platform synchronized with the new. If an unanticipated problem occurs and you need to fall back to the previous platform, you can simply execute a second switchover and no data is lost.

Failover

Failover is the term used to describe role transitions due to unplanned events. The process is similar to switchover except that the primary database never has the chance to write an EOR record. From the perspective of the standby database, redo transport has suddenly gone dormant. The standby database faithfully applies the redo from the last committed transaction that it has received and waits for redo transport to resume. At this point, whether or not a failover results in data loss depends upon the Data Guard protection mode in effect at the time of failure. There will never be any data loss in Maximum Protection. There will be zero data loss in Maximum Availability, except when a previous failure (e.g. a network failure) had interrupted redo transport and allowed the primary database to diverge from the standby database. Any committed transactions that have not been transmitted to the standby will be lost if a second failure destroys the primary database. Similarly, configurations using Maximum Performance (ASYNC) will lose any committed transactions that were not transmitted to the standby database before the primary database failed.

⁹MAA "Switchover and Failover Best Practices" for Data Guard 10g: www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_SwitchoverFailoverBestPractices.pdf

¹⁰Oracle Japan GRID Center Performance Validation: Data Guard SQL Apply on IBM Power Systems: http://www.oracle.com/technology/deploy/availability/pdf/gridcenter_sqlapply_validation_powersystem.pdf

¹¹MetaLink Note 413484.1

Myth Buster: You Must Re-create the Original Primary Databases after Failover

Beginning with Oracle 10g Release 1, you can often avoid having to restore a failed primary database from a new backup if Flashback Database was enabled on the primary database before the failover occurred (a minimum flashback retention period of 60 minutes is required). If the failed primary can be repaired and the database brought to a mounted state, it can be flashed back to an SCN that precedes the standby becoming the new primary, and converted to a standby database. When using Redo Apply, this SCN is determined by issuing the following query on the new primary database:

```
SQL> SELECT TO_CHAR(STANDBY_BECAME_PRIMARY_SCN)
FROM V$DATABASE;
```

Once the flashback operation is complete, you convert the failed primary to a physical standby database and Data Guard is able to resynchronize it with the new primary to quickly return the configuration to a protected state. This process is a little more involved for a logical standby, but will accomplish the same end result.

DBAs have the choice of configuring either *manual* or *automatic* failover. Manual failover operations give the administrator complete control of role transitions. Manual failover, however, will lengthen the outage by the amount of time required for the administrator to be notified, to respond to the notification, to evaluate what has happened, make the decision to failover, and manually execute the command. In contrast, Data Guard's Fast-Start Failover¹² feature described in Figure 1-8 automatically detects the failure, evaluates the status of the Data Guard configuration, and, if appropriate, executes the failover to a previously chosen standby database. (Fast-Start Failover is discussed in detail in Chapter 8.) In either case, executing a database failover is very fast once the decision has been made to perform a failover. Oracle has benchmarked Data Guard 11g database failover times ranging from 14 to 25 seconds depending on the configuration.¹³

Choosing Between Manual or Automatic Failover

Manual or automatic? How do you decide which approach to executing failover is right for you? Your decision is driven by several factors: RTO objectives, the complexity of application failover in your environment, and your personal comfort level using an automated versus a manual process. All things being equal, manual failover will take longer to complete simply because of the human element involved. Even if the status of the primary database is continuously monitored and alerts are automatically sent to administrators when problems occur, the administrator must respond, evaluate the current status, and decide what to do. Not only does this take time, but also the amount of time required can vary widely from one event to the next, making failover time difficult to predict. If your recovery time objective is lax enough that it can be achieved using manual failover, then there is no benefit to be gained from the additional effort required to

¹²MAA "Fast-Start Failover Best Practices" for Data Guard 10g: www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_FastStartFailoverBestPractices.pdf

¹³MAA "Switchover and Failover Best Practices" for Data Guard 10g: www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_SwitchoverFailoverBestPractices.pdf

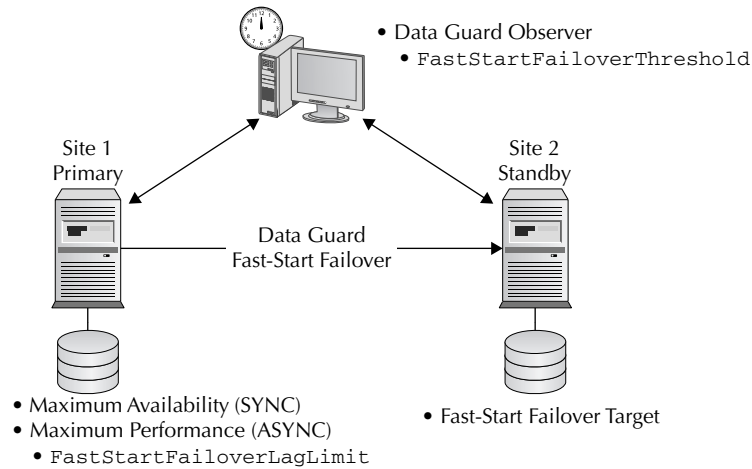


FIGURE 1-8. *Data Guard Fast-Start Failover architecture*

automate failover. However, manual failover can make more aggressive recovery time objectives very difficult, or even impossible to achieve. The more aggressive your recovery time objective, the more there is to be gained from implementing Data Guard Fast-Start Failover.

Application complexity is the second factor to consider in manual versus automatic failover. For example, a U.S. government user of Data Guard since 2003 operates a complex application environment with distributed transactions that execute across multiple databases. A zero data loss failover in Maximum Protection or Maximum Availability mode would be no problem for Fast-Start Failover. The standby database would assume the primary role with no data loss, and there would be no recovery implications for any of the other databases participating in a distributed transaction. An automatic failover in Maximum Performance mode with data loss, however, would be problematic. Manual effort is required because Data Guard is not yet able to coordinate point-in-time recovery across multiple databases participating in a distributed transaction. This user has configured Maximum Performance mode given that primary and standby databases are separated by more than 1000 miles. Even though Data Guard 11g supports automatic failover in Maximum Performance mode, it is not practical for this user to implement because of the additional manual effort required to recover multiple databases to the same point in time to preserve global data consistency following a data loss failover.

How Fast Is Automatic Failover?

Oracle documented Data Guard automatic failover performance for Oracle Database 10g Release 10.2.0.2. Failover timings for this early release of Fast-Start Failover were 17 seconds for physical standby databases and 14 seconds for logical standby databases. Users deploying later releases of Data Guard have anecdotally reported that failover times have dropped to less than 10 seconds depending on configuration.

Myth Buster: Automatic Failover Can Cause Split-Brain

The last thing you ever want to have are two independent databases, each operating as the same primary database. This can happen if, unknown to you, someone restarts the original primary database after you have performed a failover to its standby database. A common misperception is that automatic failover can increase the chance of this occurring. Not so with Data Guard Fast-Start Failover. A failed primary cannot open without first receiving permission from the Data Guard observer process. The observer will know that a failover has occurred and will refuse to allow the original primary to open. The observer will automatically reinstate the failed primary as a standby for the new primary database, making it impossible to have a “split-brain” condition.

In conversations with DBAs, we also frequently observe a reluctance to “trust” software to execute an automatic failover. This apprehension is natural. Administrators are concerned that the lack of manual control may lead to unnecessary failovers (false failovers) and disrupt operations. They fear that automatic failover may result in more data loss than acceptable, or that it may cause a split-brain condition, in which two primary databases each process transactions independent of the other. They worry that applications may not reconnect to the new primary database, impacting availability even though the database failover was successful. They are concerned that they will be forever rebuilding the original primary database after failovers occur.

While these are legitimate concerns for any automatic solution, Data Guard Fast-Start Failover has been carefully designed to avoid these problems. Data Guard has very specific, user-configurable rules to control an automatic failover for SYNC and ASYNC configurations, preventing false failovers and making it impossible for a split-brain condition to occur. It will never allow an automatic failover if the resulting data loss exceeds the previously configured recovery point threshold. It posts system events that can be used with Oracle Fast Application Notification (FAN), Fast Connection Failover (FCF) and Transparent Application Failover (TAF), or other methods external to Oracle that can reliably direct applications to reconnect to the new primary database (also discussed further in Chapter 10).¹⁴ Data Guard Fast-Start Failover automatically reinstates the failed primary database as a standby for the new primary, assuming it is salvageable, and thus creates no extra work for the DBA compared to manual failover procedures. We expect to see more companies deploy Fast-Start Failover as the increasing cost of downtime drives more aggressive RPOs, and as their internal testing validates Data Guard capabilities, eliminating obstacles to its adoption. See Chapter 8 for more details on Role Transitions.

Data Guard Management

Data Guard offers three choices for management interface: SQL*Plus, Data Guard broker, and Enterprise Manager. SQL*Plus is the traditional method for managing a Data Guard configuration. SQL*Plus is the most flexible option, but it’s also the most tedious to use. Any changes made to a Data Guard configuration require attaching directly to each system and making changes locally for that system.

¹⁴MAA “Client Failover Best Practices for Highly Available Oracle Databases”: www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_ClientFailoverBestPractices.pdf

Myth Buster: The Data Guard Broker Is a Single Point of Failure

The Data Guard broker is *not* a single point of failure. Broker processes are background processes that exist on each database in a Data Guard configuration and communicate with each other. Broker configuration files are multiplexed and maintained at all times on each database in the configuration. If the system on which you are attached fails, you simply attach to another database in the Data Guard configuration and resume management from there. More details in Chapter 5.

The Data Guard broker is a distributed management framework that automates and centralizes the creation, maintenance, and monitoring of a Data Guard configuration. It has its own command line (DGMGRL) and syntax. It simplifies and automates many administrative tasks for creation, monitoring, and management of a Data Guard configuration. Centralized management is possible by virtue of the broker maintaining a configuration file that includes profiles for all databases in the Data Guard configuration. You can connect to any database in the configuration and the broker will propagate changes to all other databases in the configuration and their server parameter files. The broker also includes commands to start an observer, the process that monitors the status of a Data Guard configuration and executes an automatic failover (Fast-Start Failover) if the primary database should fail.

Oracle Enterprise Manager provides a GUI to the Data Guard broker, replacing the DGMGRL command line and interfacing directly with the broker's monitor processes. The Enterprise Manager Data Guard management overview page is shown in Figure 1-9.



FIGURE 1-9. The Enterprise Manager Data Guard management page

Enterprise Manager also provides an easy-to-use creation wizard that provides a simple point-and-click interface to create a Data Guard configuration. Enterprise Manager requires that the Data Guard broker be enabled. If the broker is not enabled, Enterprise Manager cannot be used to manage your Data Guard configuration, and Enterprise Manager's monitoring of Data Guard related metrics is limited to redo rate, transport lag, and apply lag.

Active Standby Databases

It used to be acceptable for DR solutions to limit their scope to data protection. High availability was considered a separate topic from DR. Then along came Oracle Database 10g and Data Guard Fast-Start Failover, and all of a sudden a DR solution for Oracle Database also possesses high availability attributes. Now, instead of measuring the recovery point objective (RPO) for a DR solution in hours or days, a Data Guard RPO can be measured in seconds or minutes, depending on configuration.

Similarly, DR solutions have traditionally been characterized by standby systems that are unable to be used for any productive purpose while they maintain synchronization with the primary site. This has made DR solutions expensive because they can be used for no other purpose, and has limited their use only to the most critical databases and to companies that could afford their high cost. Sure, some SQL-based replication strategies can be used to work around this limitation, but such approaches do not work transparently with all applications and data types. SQL-based solutions also have difficulty scaling in high workload environments, and they can add considerable management complexity—increasing cost and business risk. With Oracle Database 11g and using Active Data Guard or Data Guard Snapshot Standby, physical standby databases can be used for productive purposes while they also provide DR protection. Asset utilization and performance are enhanced while complexity and the likelihood of disrupting operations when introducing changes to production environments are reduced. This results in higher return on investment with less business risk. Several examples for using your standby databases are described in the sections that follow.

Offload Read-Only Queries and Reporting

Active Data Guard enables a physical standby database to be open read-only while Redo Apply is active; queries run against the standby database receive results that are up-to-date with the primary database. Read-only queries and reports can be offloaded from the primary to the standby database, reducing I/O and CPU consumption, creating headroom for future growth, and improving quality of service for read-write transactions. The entire time the active standby is servicing queries it is also providing DR. If the primary database should fail, data is protected at the standby and failover is immediate because the standby database is completely up-to-date.

Active Data Guard also makes it very easy to test the readiness of your DR solution. In addition to the usual Data Guard status reporting, you can easily issue the same query against your primary and standby databases and compare results to validate that the standby database is functioning and up-to-date. Active Data Guard is unique in that it offers the simplicity, reliability, and high performance of physical replication, while providing much of the utility of more complex SQL-based replication technologies for read-only queries and reporting.

How Fast Are Fast Incremental Backups?

Oracle benchmarking has shown that fast incremental backups using RMAN block change tracking are up to 20 times faster than traditional incremental backups. Changed blocks are easily identified without the performance impact of full table scans. Before Active Data Guard, fast incremental backups using RMAN block change tracking could not be performed on a physical standby database.

Offload Backups

Active Data Guard also includes the ability to use RMAN block change tracking and perform fast, online, incremental backups of your physical standby database. Because backups taken on a physical standby can be used to restore either the primary or standby databases, it is no longer necessary to perform backups on the primary, freeing system resources to process critical transactions. This functionality should be considered even for companies that have previously used storage-based technologies to offload backup overhead from their production databases. For example, it's not uncommon to use storage technologies to take a full copy of a production database and then run backups from this copy. Instead of this practice, the same storage can be repurposed to deploy a local Data Guard physical standby database with Active Data Guard. RMAN fast incremental backups can be performed on the active standby database, providing the same benefit of offloading the primary. But because the standby database is active, it provides additional benefits of better data protection, higher availability, and the ability to offload read-only queries and reports from the primary database.

Testing

One of the biggest IT challenges is minimizing the risk of introducing changes to systems, databases, and applications in critical production environments. How often have you seen changes implemented over a weekend, when everything looks fine until Monday morning and real users get on the system, performance slows to a crawl, and the CEO wants to know why the problems weren't discovered in test and addressed before they disrupted business operations? Ideally, you could avoid this risk by thoroughly testing any proposed changes on a true replica of your production system and database using actual production workload. Ideally, you would also be able to run multiple tests using the same workload and data. This lets you establish a meaningful baseline against which you can iteratively assess the performance impact of proposed changes, optimizing the strategy chosen without impacting production.

Data Guard Snapshot Standby in Oracle Database 11g, a feature included with the Enterprise Edition license, has been developed to help address this problem. Using a single command, a Data Guard 11g physical standby can be converted to a snapshot standby, independent of the primary database, that is open read-write and able to be used for preproduction testing. Behind the scenes, Data Guard uses Flashback Database and sets a guaranteed restore point (GRP)¹⁵ at the

¹⁵Configuring the RMAN Environment: Guaranteed Restore Points: http://download.oracle.com/docs/cd/B28359_01/backup.111/b28270/rmcconfb.htm#BRADV89447

Myth Buster: A Physical Standby Database Can't Receive Primary Redo While Open Read-Write

A physical standby database does *not* defer shipping of redo from primary to standby when open read-write *if* you use Data Guard 11g snapshot standby. Redo for current primary database transactions continues to be received and archived by a snapshot standby database the entire time it is open read-write for testing or other purposes. Primary data is kept safe at the standby, and DR protection is assured at all times.

SCN before the standby was open read-write. Primary database redo continues to be shipped to a snapshot standby, and while not applied, it is archived for later use.

A second command converts the snapshot back into a synchronized physical standby database when testing is complete. Behind the scenes the standby is flashed back to the GRP, discarding changes made while it was open read-write. Redo Apply is started and all primary database redo archived while a snapshot standby is applied until it is caught up with the primary database. While a snapshot standby does not impact recovery point objectives, it can lengthen recovery time at failover due to the time required to apply the backlog of redo archived while it was open read-write.

Oracle Real Application Testing is a new option for the Oracle Database 11g Enterprise Edition and is an ideal complement to Data Guard snapshot standby. It enables the capture of an actual production workload, the replay of the captured workload on a test system (your Data Guard snapshot standby), and subsequent performance analysis. You no longer have to invest time and money writing tests that ultimately do an inadequate job of simulating actual workload. You don't have to try to re-create your transaction profile, load, timing, and concurrency. Using Data Guard, the process is simple:

1. Convert a physical standby database to a snapshot standby and begin capturing workload on your primary database.
2. Explicitly set a second guaranteed restore point on your snapshot standby database.
3. Replay the workload captured from the primary database on the snapshot standby to obtain a base line performance sample.
4. Flash the snapshot standby back to your explicit guaranteed restore point set in step 2.
5. Implement whatever changes you want to test on the snapshot standby.
6. Replay the workload captured from the primary database on the snapshot standby and analyze the impact of the change by comparing the results to your baseline run.
7. If you aren't satisfied with the results and want to modify the change, simply flash the snapshot standby back to your explicit guaranteed restore point set in step 2, make your modifications, replay the same workload, and reassess the impact of the change.
8. When testing is complete, convert from snapshot standby back to a physical standby. Data Guard will discard any changes made during testing and resynchronize the standby with redo it had received from primary and archived while the snapshot standby was open read-write.

Maximum Availability Architecture

The Oracle Technology network portal for MAA best practices is at <http://otn.oracle.com/goto/maa>.

Not only are you able to quickly run a series of tests using actual production workload, you are also able to run them on an exact copy of the production database, and on servers and storage sized similarly to production (given that standby systems are usually sized to run production should a failover ever be necessary). You have eliminated considerable time, effort, and expense of deploying a test system by using the DR system already in place. Most importantly, you achieve a better test result and significantly reduce the risk of impacting performance or availability when implementing changes to production systems.

Data Guard and the Maximum Availability Architecture

Data Guard is only one of the many Oracle Database capabilities that provide high availability and data protection. This chapter has touched on Oracle Real Application Clusters, Oracle Automatic Storage Management, Oracle Recovery Manager, Oracle Flashback Technologies, and Oracle Streams. Other significant features include a growing set of planned maintenance capabilities—online patching, online redefinition, online addition/subtraction of cluster nodes and storage, online configuration of memory and database parameters, and rolling database upgrades. The collective deployment of these capabilities using Oracle documented best practices is referred to as the Oracle *Maximum Availability Architecture (MAA)*. Unlike any third-party DR solution, Data Guard can leverage numerous Oracle technologies to deliver a high availability architecture that provides better data protection, higher availability, better systems utilization, and better performance and scalability, all under a common management environment. This translates into lower cost, less business risk, and greater agility to respond more quickly to changing business requirements.

Conclusion

The Latin phrase *Prodeo quod victum*, meaning “Go forth and conquer,” is an excellent note on which to end this first chapter. We have shared enough information to help you understand the basic architecture of Data Guard and what is possible to achieve. Now you are prepared for Chapter 2 and ready to begin adding to your knowledge of how to implement, manage, and get the most out of your Data Guard configuration.