SQL Apply Best Practices: Oracle Data Guard 11*g* Release 1

An Oracle White Paper
September 2008

Maximum Availability Architecture

Oracle Best Practices For High Availability

ORACLE

SQL Apply Best Practices: Oracle Data Guard 11*g* Release 1

SQL Apply Best Practices: Oracle Data Guard 11*g* Release 1

INTRODUCTION

Oracle Data Guard, an included feature of Oracle Database Enterprise Edition, provides the best data protection and availability solution for mission-critical databases. When deployed in conjunction with Oracle Real Application Clusters (Oracle RAC), Data Guard provides a unique level of scalability and high availability referred to as the Oracle Maximum Availability Architecture (MAA) [1].

Data Guard provides several deployment options so that you can implement a configuration optimized for your business and technical requirements. Its redo transport services automatically maintain standby databases by transmitting redo data from the primary database and then applying the redo to the standby database. Although the process of transmitting redo data to a remote standby database is the same for all types of Data Guard standby databases, once the changes have been received by the standby database, you have a choice of methods for applying the data to the standby database: Redo Apply (physical standby database), or SQL Apply (logical standby database). This white paper discusses the MAA best practices for SQL Apply in Oracle Database 11*g* Release 1 (11.1), and it assumes you have already reviewed the Oracle Data Guard documentation [2] [3].

BENEFITS OF SQL APPLY

Data Protection. A logical standby database contains the same logical information as the primary database, although the physical organization and structure of the data can be different between the two databases. Data Guard redo transport services keep the logical standby database synchronized with the primary database through SQL Apply, which transforms the data in the redo received from the primary database into SQL statements and then executes the SQL statements on the standby database. This protects mission-critical databases against data loss by maintaining a synchronized copy of the primary database at a geographically remote location.

Data Availability. Data Guard role management services enable the very fast transition of a standby database to the primary role. There are two types of role transitions: *failover* for handling unplanned events, and *switchover* for handling

planned outages necessary during maintenance and other purposes. As of Oracle Database Release 11g, role transitions involving a logical standby database require zero outage for read-only operations, and a very short (seconds) outage for `INSERT`s, `UPDATE`s, and `DELETE`s, achieving the most aggressive recovery time objectives (RTO). Moreover, you can enable Data Guard fast-start failover to automatically recognize when failure has occurred and rapidly execute a failover to a standby database with no manual intervention.

> **Note:** Fast-start failover supports both the `SYNC` and `ASYNC` redo transport services as of Oracle Database 11*g*. Also, you can use Data Guard in conjunction with Fast Application Notification (FAN) to automatically fail over client applications to the new primary database (FAN support for the current release of SQL Apply is limited to JDBC clients – Redo Apply does not have this restriction).

Minimizing Planned Downtime. Any Data Guard standby database can be used to minimize planned downtime during operating system and hardware maintenance. You can also use SQL Apply to perform rolling database upgrades to upgrade full patch sets or major Oracle Database releases. Beginning with Oracle Database 11g, the transient logical standby database feature enables you to perform rolling upgrades using an existing physical standby database. A physical standby database is converted temporarily to a logical standby database for the duration of the upgrade, and when the upgrade is complete the database reverts back to its original function as a physical standby database. With transient logical standby databases, you can use a physical standby database to perform a database rolling upgrade without requiring a redundant set of storage for an additional logical standby database.

Offload Applications from the Primary Database: Similar to a physical standby database (such as when using the Oracle Active Data Guard option), a logical standby database allows you to move queries, real-time reporting, and other applications to a synchronized standby database as long as such applications do not modify any of the data replicated from the primary database. The key difference from a physical standby database is that a logical standby database is open for read/write I/O, which is an important requirement for many reporting applications.

For example, consider a telephone company (Telco) that has two separate applications, with one application measuring a customer's usage of the network, and the other application billing the customer for that usage. Although the billing application does not change any of the usage information, it requires read/write access to the database to maintain its own metadata. For example, the billing application should track whether a bill has already been sent to a customer, whether or not the customer has an outstanding balance on which finance charges need to be applied, and so on. In this scenario, the primary database supports the call-measurement application, and the billing application is easily offloaded to the

logical standby database because a logical standby database is open for read/write access.

SQL Apply also allows additional indexes and materialized views on the logical standby database that do not exist at the primary database. This allows indexes (that are quite expensive to maintain in terms of the effect on an OLTP system) to be implemented on a logical standby database where they are very valuable for optimizing reporting and browsing activities. For example, a Soundex[1] index allows users to search on a name even if they do not know the exact spelling of the name but they know what the name sounds like. You can add an index such as this to a music catalog so that it is maintained only on the logical standby database, allowing users to misspell the name of the artist and still get a reasonable match. In this example, applications that perform `INSERT`s, `UPDATE`s, and `DELETE`s are run on the primary database unencumbered by the additional Soundex indexes. Thus, you can easily offload all catalog browsing to the logical standby database, where functionality and performance are enhanced by the presence of the additional indexes.

Extended Datatype Support (EDS): Beginning with Oracle Database 11*g*, SQL Apply added native data type support for XML in CLOB. Oracle Database release 11.1.0.7 (and release 10.2.0.4) provides EDS for: Object columns with simple or nested objects, VARRAYS, and Partial Spatial types (`SDO_GEOMETRY`). EDS expands the opportunities for using Data Guard logical standby databases and achieving the SQL Apply benefits described in this paper. See the MAA white paper "Extended Datatype Support: SQL Apply and Streams" and MetaLink note 559353.1 for complete information on EDS.

WHEN TO USE A LOGICAL STANDBY DATABASE

In addition to the Oracle Data Guard logical and physical standby databases, Oracle Streams is a full-featured replication solution that maintains a synchronized remote replica of a production database. Use the following guidelines to decide which of these choices is best for you:

- Use a logical standby database instead of a physical standby when you require the added flexibility of a standby database that is open for read/write access, when the performance characteristics of a logical standby can satisfy your requirements, and when you meet the prerequisites for logical standby databases. For more information, see Section 4.1: "Prerequisites" in *Oracle Data Guard Concepts and Administration* [2].
- Use a logical standby database instead of Oracle Streams if you prefer taking the simplest approach to one-way logical replication of an entire database because more sophisticated replication scenarios are not required.

[1] Soundex is a phonetic algorithm for indexing names by sound, using the English pronunciation.

CUSTOMER EXAMPLES

SQL Apply has matured considerably since its initial introduction in Oracle9*i*, and addresses a wide range of data protection and availability requirements.

The following table describes several representative customer examples:

Industry	Application	Peak Volumes
Market Research	Primary database is an OLTP system for online market research. A logical standby functions as a data warehouse and is used to offload the OLTP system of decision support, analysis, and other internal business processing. SQL Apply updates the data warehouse in near real time.	300 TPS. Redo volume averages 1.2 MB/sec over a 24-hour period. Batch processing can generate peak redo volume of 18.1 MB/sec.
Insurance	Primary database is a claims processing system running a mixed workload consisting of OLTP and batch transactions. This is a multiple-standby configuration having both physical and logical standby databases. The logical standby database supports a reporting system and provides an additional level of disaster protection.	Early rollout, workload peaks at 1,000 logical change records (LCRs) per second. OLTP transactions range from 10 to 400 LCRs each, batch range from 1,000 to 10,000+ LCRs each.
Web Publishing	Primary database is a content processing system servicing content providers. The logical standby provides high availability for online subscribers and provides disaster-recovery protection.	Bulk load LOB content. ▪ Release 10.1: 8 MB/sec. ▪ Release 10.2: 15 MB/sec.

ENHANCEMENTS AND IMPROVEMENTS IN SQL APPLY

SQL Apply New Features

Beginning in Oracle Database 11*g* release 1 (11.1), SQL Apply supports the following database features:

- Oracle Scheduler
- Fine-Grained Auditing (FGA)
- Transparent Data Encryption (TDE)
- XML Type Data Type (CLOB Only)
- Virtual Private Database (VPD)

See the Oracle Database 11*g* documentation set for more information about these features.

For a list of data types and DDL supported by SQL Apply, see Appendix C "Data Type and DDL Support on a Logical Standby Database" in *Oracle Data Guard Concepts and Administration* [2].

SQL Apply Performance Improvements

SQL Apply in Oracle Database 11*g* continues to improve on the performance gains seen in previous releases, as described in the following list.

- In release 10.1, SQL Apply was approximately 25% faster than the Oracle9*i* Database for a transactional workload.
- In release 10.2, SQL Apply improved an additional 19% over release 10.1.
- In release 11.1, SQL Apply continued to improve compared to release 10.2 in the following areas:
  - Transactions involving LOB data types
  - Very large transactions (those involving more than a million rows)

The following table shows different workloads and the performance improvement in throughput observed during testing of Oracle Database 11*g* Release 1 when compared to Oracle Database 10*g* Release 2.

Workload	Improvement in Oracle Database 11*g*
LOB `INSERT` workload (TPS)	85%
SwingBench[2] OLTP workload	15%
Single transaction mass `INSERT` of 8 million rows (redo size)	30%

- The LOB workload performed array inserts with varying sizes of LOB columns into multiple tables concurrently, and achieved the following performance improvements:

Oracle Database 11*g* Release 1	Oracle Database 10*g* Release 2
12 TPS	6.5 TPS

- The SwingBench application, which represents an order entry and fulfillment OLTP application, achieved the following throughput:

Oracle Database 11*g* Release 1	Oracle Database 10*g* Release 2
770 TPS	670 TPS

[2] Swingbench is a free load generator (and benchmark) designed to stress test an Oracle database (Oracle9*i*, Oracle Database 10*g*, and Oracle Database 11*g*). Additional information is available at http://www.dominicgiles.com/swingbench.html.

- The single transaction mass `INSERT` of an 8-million row transaction applied the transaction at the following rates:

Oracle Database 11*g* Release 1	Oracle Database 10*g* Release 2
2.1 Mbs of redo	1.6 Mbs of redo

Faster SQL Apply Performance with LOBs

With Oracle Database 11*g*, SQL Apply achieves higher throughput with LOB workload by optimizing the LOB update processing by caching the LOB locator in-memory whenever possible and by improving the utilization of LCR cache. MAA testing shows a performance improvement of approximately 85% with SQL Apply running Oracle Database release 11.1.[3]

Faster SQL Apply Performance with Multimillion Row Inserts

MAA testing shows a 10% to 20% performance improvement when applying multimillion row inserts with SQL Apply running Oracle Database release 11.1.

SQL APPLY PROCESS ARCHITECTURE

For complete information, see Chapter 10: "Managing a Logical Standby Database" in *Oracle Data Guard Concepts and Administration* [2].

SQL APPLY CONFIGURATION BEST PRACTICES FOR HIGH PERFORMANCE

After tuning and troubleshooting hundreds of logical standby databases, the MAA and SQL Apply development team has compiled these core configuration best practices to attain the best out-of-the-box SQL Apply performance.

- Follow the **general prerequisite conditions and step-by-step instructions to create a logical standby database** as described in Section 4.1: "Prerequisites" in *Oracle Data Guard Concepts and Administration* [2]**.**
- **Follow the advice in Oracle*Metalink* Note 603361.1 "Developer and DBA Tips for Pro-Actively Optimizing SQL Apply."**
- **Determine if you can use Extend Datatype Support (EDS).** See the MAA white paper "Extended Datatype Support: SQL Apply and Streams" and MetaLink note 559353.1 for complete information.
- **Set the `PRESERVE_COMMIT_ORDER` parameter to `FALSE`, if appropriate.**

---

[3] The performance comparison is between Oracle Database 10*g* Release 2 (10.2.0.3) and Oracle Database 11*g* Release 1. A number of the performance improvements available in Oracle Database 11*g* Release 1 are also available in Oracle Database 10*g* Release 2 (10.2.0.4)

By default, SQL Apply applies dependent transactions on the logical standby database in the same order the transactions were applied to the primary database. Setting the PRESERVE_COMMIT_ORDER parameter to FALSE tolerates independent transactions on the primary database to be applied in a different order on the logical standby database. Most third-party replication solutions tolerate this relaxed COMMIT processing. For OLTP applications, setting the parameter to FALSE can potentially double the SQL Apply rates.

Set PRESERVE_COMMIT_ORDER to FALSE when you want the logical standby database to quickly catch up to the primary database. For example:

```
SQL> EXECUTE
DBMS_LOGSTDBY.APPLY_SET('PRESERVE_COMMIT_ORDER', 'FALSE');
```

See Appendix A: "SQL Apply and the PRESERVE_COMMIT_ORDER Parameter" for performance comparison numbers.

- **Set the MAX_SGA=200 MB (minimum value) and increment the SHARED_POOL_SIZE parameter by the size of the MAX_SGA.**

  To avoid expensive pageout operations, SQL Apply leverages a logical change record (LCR) cache to store its metadata. If the cache fills up during heavy activity, then expensive pageouts occur and SQL Apply writes to a SPILL table that resides in SYSAUX tablespace. The default 30 MB setting for the MAX_SGA parameter is much too low for most applications. For example:

  ```
  SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SGA', 200);
  ```

- **Set the MAX_SERVERS parameter to eight times the core.**

  SQL Apply automatically distributes the server processes. There will always be one process for the READER, BUILDER, and ANALYZER roles but in most cases you need a varied number of PREPARER and APPLIER processes. Having a reasonable default setting avoids a lot of initial tuning. The current default setting of 9 is much too low for an application with large transactions or a lot of concurrent transactions. With a higher default setting, there can be more PGA memory utilization. For example:

  ```
  SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SERVERS', <8 x
  CORE>);
  ```

  > **Note:** If you are running a massively parallel system (such as a system with more than 32 Cores), set MAX_SERVERS to a maximum of 100.

  However, if you have a system generating hundreds of transactions (as seen in the AWR report on the primary database), then set MAX_SERVERS to the number of concurrent transactions.

- **Set `_HASH_TABLE_SIZE=10000000` (10 million)**

  An in-memory hash table stored in the `ANALYZER PGA` space maintains a dependency tree for every LCR and transaction. Setting a larger size for the `HASH_TABLE_SIZE` parameter increases the PGA space by approximately 120 MB for a ten-million size hash table, but avoids costly dependency wait events on replicated tables with many constraints. For example:

  ```
  SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('_hash_table_size',10000000);
  ```

- **Set `LOB` objects in a tablespace with larger data block size if possible on the primary and standby databases.**

  Setting a larger data block size reduces the number of LCRs created on the logical standby database because the minimum number of LCRs equal to the size of the `LOB` divided by data block size. `LOB` columns that are stored out-of-line in the database are stored in `CHUNKS` that are defined when the table is created. Each `CHUNK` appears as an LCR in the redo stream, and preceding the first `CHUNK`, there is a record locator that contains the non-`LOB` part of the row.

  See the *Oracle Database SecureFiles and Large Objects Developer's Guide* [5] for complete information about setting data block sizes.

- **Defer data definition language (DDL) statements until noncritical periods**

  When SQL Apply encounters a DDL transaction that needs to be applied, mining of subsequent transactions is suspended and a DDL barrier is established. The DDL transaction is assigned to an applier process that must wait until all preceding transactions have committed. Once committed, the DDL barrier is removed, allowing the miner process to mine additional transactions. For highly concurrent transaction processing applications, mining and applying these DDL transactions can throttle SQL Apply performance significantly.

MONITORING, TUNING, AND TROUBLESHOOTING SQL APPLY

When monitoring SQL Apply performance, the goal is to leverage a minimum set of statistics to identify problems and to quickly tune or adjust SQL Apply for optimal performance.

1. Run AWR, ADDM, or Statspack every 30 minutes.
2. Run the Alert Log Parser for key alerts and messages for tuning the logical standby database.

Symptom: If all Applier processes are busy, then:

1. Issue the following query:

```
select min(pct_applied) pct_applied_min , max(pct_applied)
pct_applied_max, avg(pct_applied) pct_applied_avg ,
count(server_id) number_of_appliers from ( select server_id,
(greatest(nvl(s.total_assigned,0),0.00000001)
/greatest(nvl(c.total_assigned,1),1)) * 100 pct_applied from
v$streams_apply_server s , v$streams_apply_coordinator  c )
```

2. If all Appliers are equally busy (as shown in the following example), then adjust the `MAX_SERVERS` parameter, as appropriate:

```
PCT_APPLIED_MIN  PCT_APPLIED_MAX  PCT_APPLIED_AVG  NUMBER_OF_APPLIERS
---------------  ---------------  ---------------  ------------------
6.24741649       6.25306751       6.25             16
```

Symptom: If the pageout value increases over time, then:

1. Issue the following query:

```
select name, to_number(value) value from
v$logstdby_stats where name = 'bytes paged out'
```

2. If the number of bytes paged increases over time, then consider increasing the `MAX_SGA` parameter.

Symptom: If there are large transactions, then:

1. Issue the following query:

```
select state,commitscn,message_sequence from
v$streams_apply_server where commitscn=0 or ( message_sequence
> 200 and state != 'IDLE');
```

2. Consider an application change or adjust the `_eager_size` parameter.

Symptom: If DDL then:

1. Issue the following query

```
select name,value from v$logstdby_stats where name in 'DDL txns
mined';
```

2. DDL statements cause a "`DDL Barrier`" which causes SQL Apply to force a serialization point within the application of transactions. Defer DDL statements to nonpeak hours when possible.

Symptom: If you receive the `ORA-26786 or ORA-26787` error message, then follow the guidelines in Section 10.4.5 "Adding or Re-Creating Tables on a Logical Standby Database" in *Oracle Data Guard Concepts and Administration*.

ADDITIONAL DATABASE TUNING

If the SQL Apply specific tuning does not result in the expected throughput then you may be incurring a generic database tuning issue. See the *Oracle Performance and Tuning Guide 11g Release 1 (11.1)* [4] for the latest performance tuning overview, tools, and methodologies. Follow the iterative process to determine if system or database resources are constraining your SQL Apply throughput.

CONCLUSION

By following the best practices described in this paper, you should get insight into the significant performance gains for SQL Apply running Oracle Database 11*g* Release 1 (11.1). The configuration best practices also help you to avoid the most common and simple SQL Apply performance issues. The queries and tools help you to detect and troubleshoot application-specific issues that may slow SQL Apply performance. The recommended practices provide in-depth insights into how to optimize your specific SQL Apply environment.

APPENDIX A: SQL APPLY AND PRESERVE_COMMIT_ORDER

The following example uses the SwingBench application running the Order Entry workload on HP ProLiant DL385 Server hardware, Red Hat Linux operating system (64-Bit), and Oracle Database 11*g* release 11.1 (64-Bit). The number of applier processes remained constant for the two runs, and the only thing changed was the setting of the PRESERVE_COMMIT_ORDER parameter.

Setting the PRESERVE_COMMIT_ORDER Parameter

The PRESERVE_COMMIT_ORDER parameter controls the order in which unrelated transactions are applied to the standby database. Normally, SQL Apply commits all transactions in the same sequence as the primary database. This is known as *transactionally consistent ordering* that you explicitly specify by setting the PRESERVE_COMMIT_ORDER parameter to TRUE.

If you are using the SQL Apply database only for disaster-recovery purposes or if the reporting application using the standby database can accommodate transactions being applied out of sequence, then set the PRESERVE_COMMIT_ORDER parameter to FALSE. When SQL Apply is running in this model, transactions that are unrelated to each other may be applied out of sequence.

The following examples illustrate the effect of setting the PRESERVE_COMMIT_ORDER initialization parameter for different situations. The examples are based on a scenario in which a long-running transaction on the primary database starts at 9:30:01 a.m. and commits at 9:35:00 a.m., and a second unrelated transaction starts at 9:35:01 a.m. and commits at 9:35:02 a.m.

Example 1: For this example, assume that on the logical standby database, SQL Apply starts to apply the first transaction immediately and commits the transaction at 9:35:00 a.m., thus taking 5 minutes for the transaction to be applied. The second transaction is available on the standby database at 9:35:02 a.m. and takes only 1 second to be applied. In this example scenario, consider the effect of the following parameter settings:

- If PRESERVE_COMMIT_ORDER=TRUE, then SQL Apply commits the first transaction to the standby database at 9:40:00 a.m. and commits the second transaction at 9:40:01 a.m.
- If PRESERVE_COMMIT_ORDER=FALSE then SQL Apply commits the first transaction to the standby database at 9:40:00a.m., but because the second transaction was available on the standby database at 9:35:02 a.m. and takes only 1 second to commit, the second transaction would be seen if the standby database was queried at 9:35:03 a.m.

Note: If transaction 1 and transaction 2 share a common row then transaction 2 will not be executed until after transaction 1 has been committed regardless of the setting of the PRESERVE_COMMIT_ORDER parameter.

Example 2: If transaction 1 inserts 100 rows into the employee table, and transaction 2 updates one of these 100 rows with a new salary, then the two transactions (because they are related) are always applied in sequence.

Performance Impact of PRESERVE_COMMIT_ORDER

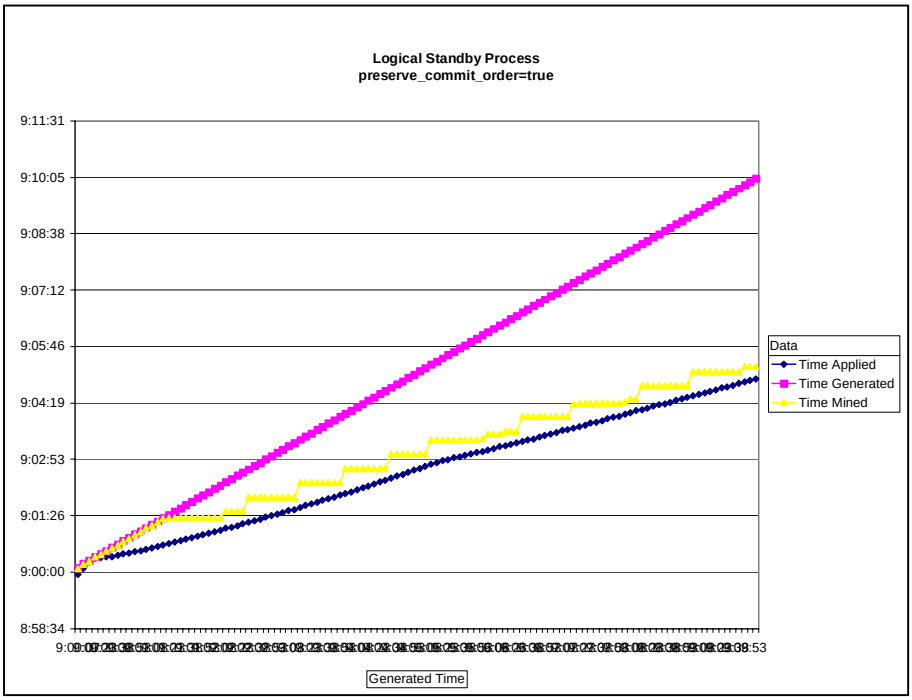
The following table shows different workloads and performance improvements observed from testing Oracle Database 11*g* Release 1 with the `PRESERVE_COMMIT_ORDER` parameter set to `FALSE` compared to when `PRESERVE_COMMIT_ORDER` is set to `TRUE`.

Workload	**Improvement (decline) when `PRESERVE_COMMIT_ORDER` is set to `FALSE` for Oracle Database 11*g* Release 1**
LOB `INSERT` Workload (TPS)[4]	(5.00%)
SwingBench OLTP Workload	
SwingBench against a small data set nonpartitioned schema (TPS)	40%
SwingBench against a small data set partitioned schema (TPS)	80%
SwingBench against a large data set partitioned schema (TPS)	90%
Single Transaction Mass `INSERT` Workload	
Mass `INSERT` of 1-million rows (redo size)	5%
Mass `INSERT` of 8-million rows (redo size)	(3%)

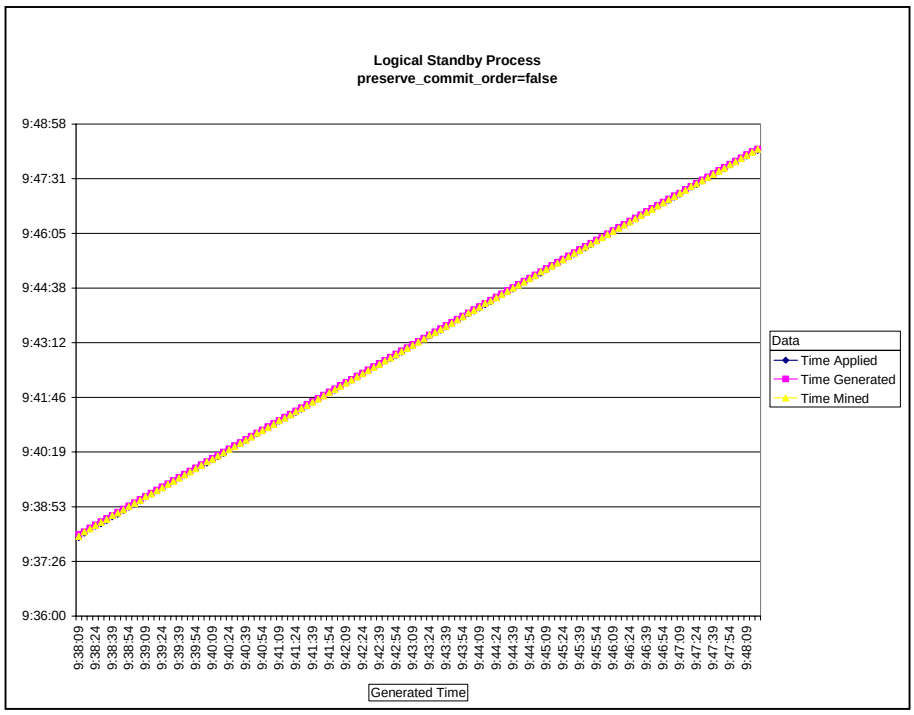
Note: The SwingBench OLTP workloads benefit significantly when the `PRESERVE_COMMIT_ORDER` parameter is set to `FALSE`. This is due to the high concurrency of transactions, while the LOB `INSERT` workload and the Single Transaction workloads do not benefit as much.

[4] This workload models a document management company.

The following chart shows the Order Entry workload throughput when the PRESERVE_COMMIT_ORDER parameter is set to TRUE which is the default.



The following chart shows the Order Entry workload throughput when the PRESERVE_COMMIT_ORDER parameter is set to FALSE



Although the number of Applier processes remains the same, the graph shows that when SQL Apply is running with the PRESERVE_COMMIT_ORDER parameter set to FALSE, then the apply time on the logical standby database can keep up to date with the primary database generated time.

REFERENCES

1. Oracle Maximum Availability Architecture Web site
   http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm
2. *Oracle Data Guard Concepts and Administration 11g Release 1 (11.1)*
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28294
3. *Oracle Data Guard Broker 11g Release 1 (11.1)*
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28295
4. *Oracle Performance and Tuning Guide 11g Release 1 (11.1)*
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28274
5. *Oracle Database SecureFiles and Large Objects Developer's Guide*
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28393
6. Oracle*Metalink* Note 603361.1 "Developer and DBA Tips for Pro-Actively Optimizing SQL Apply"

SQL Apply Best Practices: Oracle Data Guard 11*g* Release 1
September 2008
Author: Andrew Babb
Contributing Authors: Joydip Kundu, Joseph Meeks, Lawrence To

Oracle USA, Inc.
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com