Oracle Logical Standby –The best of both worlds

TJ Mitra
Datacom Systems Limited

Oracle Corporation has continually improved the High Availability (HA) capabilities of the Oracle RDBMS product. These capabilities have become vital, in order to support the complex, 24 by 7 environments, installed at many customer sites. Oracle's initial HA options were Read-only snapshot & Multi master replication. With Oracle 8i, the Oracle Data Guard 'product' was released. That introduced the notion of a Physical standby database. Oracle 9i v 9.2 introduced Logical Standby which has subsequently been enhanced in Oracle 10G.

This article describes the steps required to set up a logical standby database based on my personal experience with a major Datacom client. As part of the discussion I intend to address the following:

- The differences between Oracle Physical Standby & Logical Standby
- Limitations of Logical Standby
- Building an Oracle9i Logical Standby database
- How to switchover to Logical Standby database and Switchback
- Fail over to the Logical standby database
- Describe the Capability of Enhancing Logical Standby for Reporting
- Monitoring of Logical Standby and handling SQL Apply
- Mention the enhancements of Oracle 10G Logical Standby

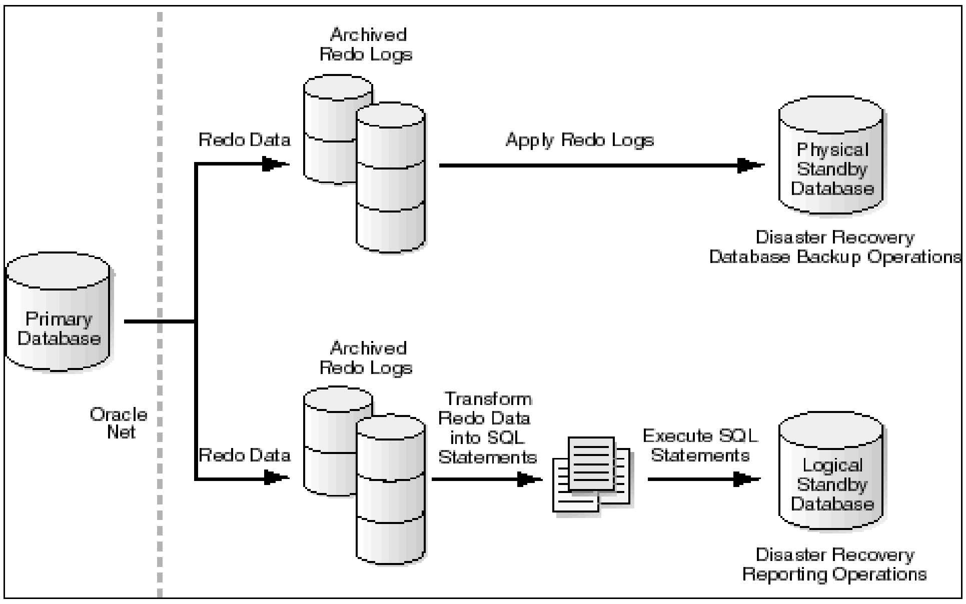
Differences between Oracle Physical Standby & Logical Standby

Oracle Physical Standby gives an environment where a Standby database, originally set up from a backup of the Primary, follows the Primary by way of applying Archive logs generated at the Primary. An Oracle Physical Standby database, at any point in time, can either be in Managed Recovery Mode or in Read Only mode. Switching between these two states can be achieved without loosing any thread of continuity though while the Standby Database is in the Read Only Mode, as logs can not be applied, it will lag behind Primary until it is put back in Managed Recovery mode again.

Oracle Logical Standby gives an environment where a Standby database, also set up from a backup of the Primary, follows the Primary by way of applying sql statements from the Archive logs mined by the Log Miner. As a result, Oracle Logical Standby, at any point in time, can remain open even when the Sql statements are being applied. So, in effect, the Logical Standby can be used for HA without any lag, as well as a Reports/Query processing Server. Additional indexes or tables can be created on the Logical Standby databases for effective query performance.

Though a Logical Standby database is originally built replicating the objects of the primary, business requirements can also lead it towards Data Divergence from the Primary.

The schematic diagram for Oracle Data Guard is best described by the following
(Ref: Oracle Data Guard Concepts & Administration 9.2)


Archived Redo Logs
Redo Data
Apply Redo Logs
Physical Standby Database
Disaster Recovery Database Backup Operations
Primary Database
Oracle Net
Archived Redo Logs
Redo Data
Transform Redo Data into SQL Statements
Execute SQL Statements
Logical Standby Database
Disaster Recovery Reporting Operations

Data Guard consists of 3 main services, namely:

- Log Transport Services, which control transfer of redo data from Primary to Standby in automated manner
- Log Apply Services, which apply redo data in the Standby
- Role Management Services, which handle the change of role of a database from a standby database to a primary database, or from a primary database to a standby database using either a switchover or a failover operation.

In the present case, LGWR, instead of ARCH, has been used on Primary to ship redo data to Standby. Oracle Data Guard Remote File Server Process (RFS) on the Standby is recipient of redo data from Primary.

Logical Standby Process (LSP) on Standby is responsible to apply Sql translated redo data on the Logical Standby database.

Protection Modes of Oracle Data Guard:

Oracle Data Guard offers 3 protection modes namely Maximum Protection, Maximum Availability and Maximum Availability.

Maximum Protection protection offers Zero data loss by way of making sure that transaction redo data is written in the Local redo log as well as in the Standby redo log(SRL) in the Standby before the transaction commits. For this requirement this mode will only operate with LGWR as Redo writer process and the Network transmission mode is SYNC and Disk Write option is AFFIRM. If a fault occurs preventing write to SRL, the Primary database will shut down.

Maximum Availability protection is very similar to Maximum Protection with a difference that the Primary will not shut down if writing to SRL fails. During that period, it will operate in resync mode until the fault is corrected and all log gaps have been resolved. When all log gaps have been resolved, the primary database automatically resumes operating in maximum availability mode.

Maximum performance mode operates without affecting Primary performance in any way. In this mode, the redo data is transferred asynchronously over the network and without any affirmation of Disk write. Either LGWR or ARCH can be used for redo data shipping.

The default protection mode is MAXIMUM PERFORMANCE. To change it to Maximum Protection or Availability, following steps are needed

a) For the MAXIMUM PROTECTION mode, the standby must be up and mounted.

```
SQL> alter system set log_archive_dest_2='service=STBY LGWR SYNC';
SQL> alter system set log_archive_dest_state_2=enable;
```

These parameters should be set in the init.ora or spfile.

b) The primary will need to be closed and in a mounted state.

```
SQL> shutdown immediate
SQL> startup mount
```

c) Change the protection mode and open:

```
SQL> alter database set standby database
    to maximize <PROTECTION | AVAILABILITY >;
SQL> alter database open;
```

Limitations of Logical Standby

Before setting up a logical standby database, one must make sure the logical standby database can maintain the data types and tables in the primary database.

The following lists indicate which of the various database objects are supported in logical standby databases.

Supported Data types for Oracle 9.2 Logical Standby

```
CHAR
NCHAR
VARCHAR2 and VARCHAR
NVARCHAR2
NUMBER
DATE
TIMESTAMP
TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
RAW
CLOB
BLOB
```

Unsupported Datatypes

Logical standby databases do not support columns of the following data types:

```
NCLOB
LONG
LONG RAW
BFILE
ROWID
UROWID
User-defined type (including object types, REFs, varrays, and nested tables).
```

Unsupported Tables and Sequences

```
Tables and sequences in the SYS schema
Tables with unsupported data types
Tables used to support functional indexes
Tables used to support materialized views
Global temporary tables
```

Database view DBA_LOGSTDBY_UNSUPPORTED could be checked to determine whether the primary database contains unsupported objects.

If the primary database contains unsupported tables, log apply services automatically exclude the tables when applying redo logs to the logical standby database. An error is returned if the logical standby database attempts to apply DML changes for a table that contains a column of an unsupported data type. Logical Standby will also not work if Oracle Label Security Features has been implemented in the Primary.

Oracle 10G has extended the support for more data types – like Long & Long Raw data types are supported.

Building an Oracle9i Logical Standby database

Oracle Metalink Document 186150.1 can be used for setting up the Logical Standby

- First ensure that the Primary database is in archivelog mode & archiving is enabled.
- Apply force logging in the Primary to log any unlogged operation performed

```
SQL> alter database force logging;
```

- Then check whether all tables have Primary or Unique indexes.
- Check whether Supplemental Logging is enabled in the primary database

```
SQL> SELECT SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI FROM
V$DATABASE;

SUP SUP
----- -----
NO  NO
```

- If not, enable supplemental logging

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE INDEX)
COLUMNS;

Database altered.

SQL> alter system switch logfile;
```

- Enabling Supplemental Logging is an essential requirement for Logical Standby. Supplemental Logging enables usage of log-miner features which is required in applications that applies reconstructed SQL statements to a different database and must identify the update statement by a set of columns that uniquely identify the row (for example, a primary key), not by the ROWID shown in the reconstructed SQL returned by the V$LOGMNR_CONTENTS view, because the Rowid of one database will be different and therefore meaningless in another Database
- Set Log_parallelism =1

- Create an alternate tablespace in the primary database for logical standby system tables.

```
SQL> EXECUTE DBMS_LOGMNR_D.SET_TABLESPACE('logical_tblspce');
```

This is required for switchover operation.

- Then in the Primary database do the following

Take a cold backup of production database, create a backup controlfile in database mounted state, open the database and create a logminer dictionary by running

```
SQL> EXECUTE DBMS_LOGSTDBY.BUILD;
```

- Archive the current online redo log.

```
SQL> ALTER SYSTEM archive log current;
```

- Identify the archived redo log that contains the logminer dictionary for use in the standby creation process.

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE DICTIONARY_BEGIN='YES'  and
STANDBY_DEST='NO';
```

- Copy backup datafiles, backup control file and the latest archived redo log that was identified in the previous steps, and a copy of the primary initialization parameter file to the standby host.

- Init parameters in the Primary should include the following parameters related to Standby set up:

```
*.db_name='smdbprd'
*.log_archive_dest_1='LOCATION=e:\oracle\admin\smdbprd\archive'
*.log_archive_dest_state_2=enable
*.log_archive_dest_2='service=smdbprd_stdby lgwr'
*.log_archive_dest_state_2=enable
*.parallel_max_servers=9
*.standby_archive_dest=e:\oracle\admin\smdbprd\archive
*.standby_file_management=auto
```

- Init parameters in the Logical Standby should include the following parameters:

```
*.db_name='smdbprd'
*.instance_name='smdbprd'
*.log_archive_dest_1='LOCATION=e:\oracle\admin\smdbprd\archive'
*.log_archive_dest_state_1=enable
*.log_archive_dest_2='service=smdbprd lgwr'
*.log_archive_dest_state_2=defer
*.parallel_max_servers=9
*.standby_archive_dest='e:\oracle\admin\smdbprd\archive'
*.standby_file_management=auto
```

Having symmetrical init parameters in both primary & Standby makes it easier during switchover & switchback operations and they are meaningful only in the respective role of the databases e.g.

```
log_archive_dest_1 : used both when Primary and when Standby
```

```
log_archive_dest_2 : used when Primary, ignored when Standby
standby_archive_dest : used when Standby, ignored when Primary
standby_file_management : used when Standby, ignored when Primary
```

On the Standby Host, do the following:

- In the Standby, database guard should be turned on

```
SQL> ALTER DATABASE GUARD ALL;
```

- Open the logical Standby followed by a shutdown with immediate or normal.

```
SQL> ALTER DATABASE OPEN RESETLOGS;
SQL> SHUTDOWN IMMEDIATE;
```

- Mount the logical standby in exclusive mode and with the dbnewid utility reset the database name.

```
SQL> startup mount exclusive
SQL> exit
```

Though the Primary and Standby can operate with the same dbname & dbid, Oracle advises to change them

```
 $ nid target=sys/password dbname=DB2 setname=yes
```

Shutdown the database and change the DB_NAME init.ora parameter to the new name in the initialization file and change the ORACLE_SID.

```
SQL> SHUTDOWN IMMEDIATE
```

- Open the logical standby database in exclusive mode.

```
SQL> STARTUP MOUNT EXCLUSIVE;

SQL> ALTER DATABASE OPEN;
```

- Drop each current temporary file from the standby database

```
SQL> ALTER DATABASE TEMPFILE 'tempfilename' DROP;
```

- Add a new temporary file, for example:

```
  SQL> ALTER TABLESPACE TEMP ADD TEMPFILE  ‘c:\oracle\oradata\smdbprd\temp01.dbf' SIZE
1000M REUSE;
```

Adequate temporary space is essential for the initial load of the logminer dictionary.

The archive log identified during the logminer dictionary build step should now be registered in the Logical Standby database.

```
  SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
'e:\oracle\admin\smdbprd\archive\smdbprd_nnn.arc';
```

- Use the following ALTER DATABASE statement and include the INITIAL keyword to begin SQL apply operations for the first time on the logical standby.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY INITIAL;
```

- Configure listeners on the primary & logical standby databases.

Primary Listener Entries:

```
SMDBPRD_LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = DB_HOST_1)(PORT = 1531))
      )
    )
  )

SID_LIST_SMDBPRD_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = smdbprd)
      (ORACLE_HOME = c:\oracle\ora92)
      (SID_NAME = smdbprd)
    )
  )
```

Standby Listener Entries:

```
SMDBPRD_STDBY_LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = DB_HOST_2)(PORT = 1531))
      )
    )
  )
SID_LIST_SMDBPRD_STDBY_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = smdbprd)
      (ORACLE_HOME = c:\oracle\ora92)
      (SID_NAME = smdbprd)
    )   )
```

- Start the listeners on the primary & standby host. Once both standby and production listeners are running manually register the service names to ensure listener registration.

```
SQL> ALTER SYSTEM REGISTER  ;
```

- Create a net service name that the logical standby database can use to connect to the primary database within the tnsnames.ora on the standby host.

Logical Standby tnsnames.ora

```
# TNSNAMES.ORA Network Configuration File:
# C:\oracle\ora92\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.
SMDBPRD =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = DB_HOST_1)(PORT = 1531))
    )
    (CONNECT_DATA =
      (SID = smdbprd)
      (SERVER = DEDICATED)
    )
  )
```

On the primary database create a net service name that the primary database can use to connect to the logical standby database

Primary tnsnames.ora

```
smdbprd_stdby =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = tcp)(HOST = DB_HOST_2)(PORT = 1531))
    )
    (CONNECT_DATA =
      (SID = smdbprd)
            (SERVER = DEDICATED)
    )
  )
```

- Enable archiving to the logical standby database.

```
SQL> ALTER SYSTEM SET log_archive_dest_2='service=smdbprd_stdby lgwr';
SQL> ALTER SYSTEM SET log_archive_dest_state_2=enable;
```

Check these values are also set in the init.ora file.

- Start archiving the current redo logs.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

During Switchover & Switchback, the DBLINKS to STANDBY & PRIMARY are respectively used, so DB_LINKS have to be created in the respective databases:

In Primary, create db_link as follows

```
SQL> Create database link STANDBY connect to sys identified by password using ‘smdbprd_stdby’;
```

In Standby, create db_link as follows

```
SQL> Create database link PRIMARY connect to sys identified by password using ‘smdbprd’;
```

Few database views related to Logical Standby
Dba_logstdby Progress

The DBA_LOGSTDBY_PROGRESS view describes the progress of SQL apply operations on the logical standby database. The APPLIED_SCN indicates that committed transactions at or below that SCN have been applied. The NEWEST_SCN is the maximum SCN to which data could be applied if no more logs were received. This is usually the MAX(NEXT_CHANGE#)-1 from DBA_LOGSTDBY_LOG. When the value of NEWEST_SCN and APPLIED_SCN are   the equal then all available changes have been applied. If the APPLIED_SCN is below NEWEST_SCN and is increasing then SQL apply is currently processing changes.

```
SQL> set numwidth 15
SQL> select
        (case
         when newest_scn = applied_scn then 'Done'
         when newest_scn <= applied_scn + 9 then 'Done?'
         when newest_scn > (select max(next_change#) from dba_logstdby_log)
         then 'Near done'
         when (select count(*) from dba_logstdby_log
         where (next_change#, thread#) not in
         (select first_change#, thread# from dba_logstdby_log)) > 1
         then 'Gap'
         when newest_scn > applied_scn then 'Not Done'
         else '---' end) "Fin?",
         newest_scn, applied_scn, read_scn from dba_logstdby_progress;

            Fin?         NEWEST_SCN    APPLIED_SCN       READ_SCN
            ---------   --------------------   ------------------- ----------------------
            Done            1683678            1683678              1669292
SQL>
```

Another time running this sql shows a gap exists.

```
SQL> set numwidth 15
SQL> select
  2  (case
  3  when newest_scn = applied_scn then 'Done'
  4  when newest_scn <= applied_scn + 9 then 'Done?'
  5  when newest_scn > (select max(next_change#) from dba_logstdby_log)
  6  then 'Near done'
  7  when (select count(*) from dba_logstdby_log
  8  where (next_change#, thread#) not in
  9  (select first_change#, thread# from dba_logstdby_log)) > 1
 10  then 'Gap'
 11  when newest_scn > applied_scn then 'Not Done'
 12  else '---' end) "Fin?",
 13  newest_scn, applied_scn, read_scn from dba_logstdby_progress;

Fin?        NEWEST_SCN    APPLIED_SCN      READ_SCN
---------   ---------------      ---------------          ---------------
Gap           42262288           41918378           41922186

SQL>
```

Dba_logstdby_events

This view gives a history on logical standby apply activity. This history also appears in the Standby alert log.

```
SQL> set numwidth 15
SQL> select to_char(event_time, 'MM/DD HH24:MI:SS') time,
  2  commit_scn, current_scn, event, status
  3  from dba_logstdby_events
  4  order by event_time, commit_scn, current_scn;

TIME               COMMIT_SCN     CURRENT_SCN
-------------- --------------- ---------------
EVENT
--------------------------------------------------------------------------------
STATUS
------------------------------------------------
03/29 10:34:16

ORA-16111: log mining and apply setting up

03/30 01:17:41        41549580        41549566
create table users_info as select * from dba_users
ORA-16204: DDL successfully applied

03/30 01:49:32        41550864        41550857
drop table users_info
ORA-16204: DDL successfully applied

03/31 12:13:29        41628665        41628662
alter user system identified by  VALUES '141791286B2F8A93'
ORA-16204: DDL successfully applied
03/31 12:13:37        41632279        41632276
alter user scott identified by  VALUES 'AF7AE51CFE460189'
ORA-16204: DDL successfully applied

03/31 12:26:45        41632854        41632852
alter database force logging
ORA-16205: DDL skipped due to skip setting

04/01 16:22:14

ORA-16128: User initiated shut down successfully completed 04/01 16:23:36

ORA-16111: log mining and apply setting up

04/01 16:25:54        41699308        41699306
ALTER DATABASE OPEN
ORA-16205: DDL skipped due to skip setting

04/01 16:25:55        41699378        41699361
create table user_info as select * from dba_users
ORA-16204: DDL successfully applied

04/01 16:36:43        41699818        41699811
drop table user_info
ORA-16204: DDL successfully applied
```

```
04/05 14:25:33        41922189        41922187
grant sysdba to rman
ORA-01031: insufficient privileges
04/05 14:25:33
ORA-16111: log mining and apply setting up
04/05 14:25:33
ORA-16111: log mining and apply setting up
04/05 14:25:41        41922189        41922187
grant sysdba to rman
ORA-01031: insufficient privileges
04/07 17:33:25
ORA-16111: log mining and apply setting up
04/07 17:33:35        41922189        41922187
grant sysdba to rman
ORA-01031: insufficient privileges
04/07 17:33:35
```

This shows Sql Apply has been going fine before being stuck after a sysdba grant is issued on Primary.

Other related views are:
V$archive_dest
V$archive_dest_status
V$logstdby
V$logstdby_stats
dba_logstdby_log

Graceful Switchover to Logical Standby database and Switchback

Switchover steps:

For switchover to standby, following steps are performed on the Primary

```
C:\>set ORACLE_SID=smdbprd
C:\>sqlplus
SQL*Plus: Release 9.2.0.6.0 - Production on Wed Jun 22 11:26:52 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Enter user-name: sys/manager as sysdba

Connected to:
Oracle9i Enterprise Edition Release 9.2.0.6.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.6.0 - Production

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL> show parameter log_archive_dest
```

```
NAME                             TYPE        VALUE
---------------------------------- ----------- ------------------------------
log_archive_dest_1               string      LOCATION=e:\oracle\admin\smdbprd\archive
log_archive_dest_2               string      service=smdbprd_stdby lgwr
.
.
log_archive_dest_state_1         string      enable
log_archive_dest_state_2         string      enable

SQL> select db_link,host  from dba_db_links;

STANDBY, smdbprd_stdby

SQL> select guard_status from v$database;

GUARD_STATUS
-----------------------
NONE

SQL> alter database commit to switchover to logical standby;

Database altered.

SQL> select database_role from v$database;

DATABASE_ROLE
---------------------------
LOGICAL STANDBY

SQL> alter system set log_archive_dest_state_2=defer;

System altered.

SQL> select guard_status from v$database;

GUARD_STATUS
-----------------------
ALL

SQL> alter database start logical standby apply new primary "STANDBY";

Database altered.

SQL>
```

For switchover to primary, following steps are performed on the standby

```
C:\>set ORACLE_SID=smdbprd

C:\>sqlplus

SQL*Plus: Release 9.2.0.6.0 - Production on Wed Jun 22 11:29:50 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Enter user-name: sys/manager as sysdba

Connected to:
```

```
Oracle9i Enterprise Edition Release 9.2.0.6.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.6.0 – Production

SQL> select database_role from v$database;

DATABASE_ROLE
---------------------------
LOGICAL STANDBY

SQL> show parameter log_archive_dest

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
log_archive_dest_1                   string      LOCATION=e:\oracle\admin\smdbprd_archive
log_archive_dest_2                   string      service=smdbprd lgwr
.
.
log_archive_dest_state_1             string      enable
log_archive_dest_state_2             string      defer

SQL> select db_link, host from dba_db_links;

DB_LINK          HOST
--------------   --------------------------------------------------------------
PRIMARY     smdbprd

SQL>  select guard_status from v$database;

GUARD_STATUS
-----------------------
ALL

SQL> select database_role from v$database;

DATABASE_ROLE
--------------------------
LOGICAL STANDBY

SQL> alter database commit to switchover to primary;

Database altered.

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL> select guard_status from v$database;

GUARD_STATUS
------------------------
NONE

SQL> alter system set log_archive_dest_state_2=enable;

System altered.
```

```
SQL>
```

Switchback Steps:

For Switchback, the reverse operation applies.

For Switchback to original Primary, first start with the Current Primary (Original Standby):

```
SQL> select database_role from v$database;

DATABASE_ROLE
------------------------
PRIMARY

SQL> select guard_status from v$database;

GUARD_STATUS
----------------------
NONE

SQL> alter database commit to switchover to logical standby;

Database altered.

SQL> select database_role from v$database;

DATABASE_ROLE
---------------------------
LOGICAL STANDBY

SQL> select guard_status from v$database;

GUARD_STATUS
----------------------
ALL

SQL> alter system set log_archive_dest_state_2=defer;

System altered.

SQL> alter database start logical standby apply new primary "PRIMARY";

Database altered.

SQL>
```

Issue the following statements in the Original Primary (Current Standby) to complete the switchback operation:

```
SQL> select database_role from v$database;

DATABASE_ROLE
----------------------------
LOGICAL STANDBY

SQL> select guard_status from v$database;
```

```
GUARD_STATUS
------------------------
ALL

SQL> alter database commit to switchover to primary;

Database altered.

SQL> alter system set log_archive_dest_state_2=enable;

System altered.

SQL> select guard_status from v$database;

GUARD_STATUS
------------------------
NONE

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL>
SQL> alter system switch logfile;

System altered.

SQL>
```

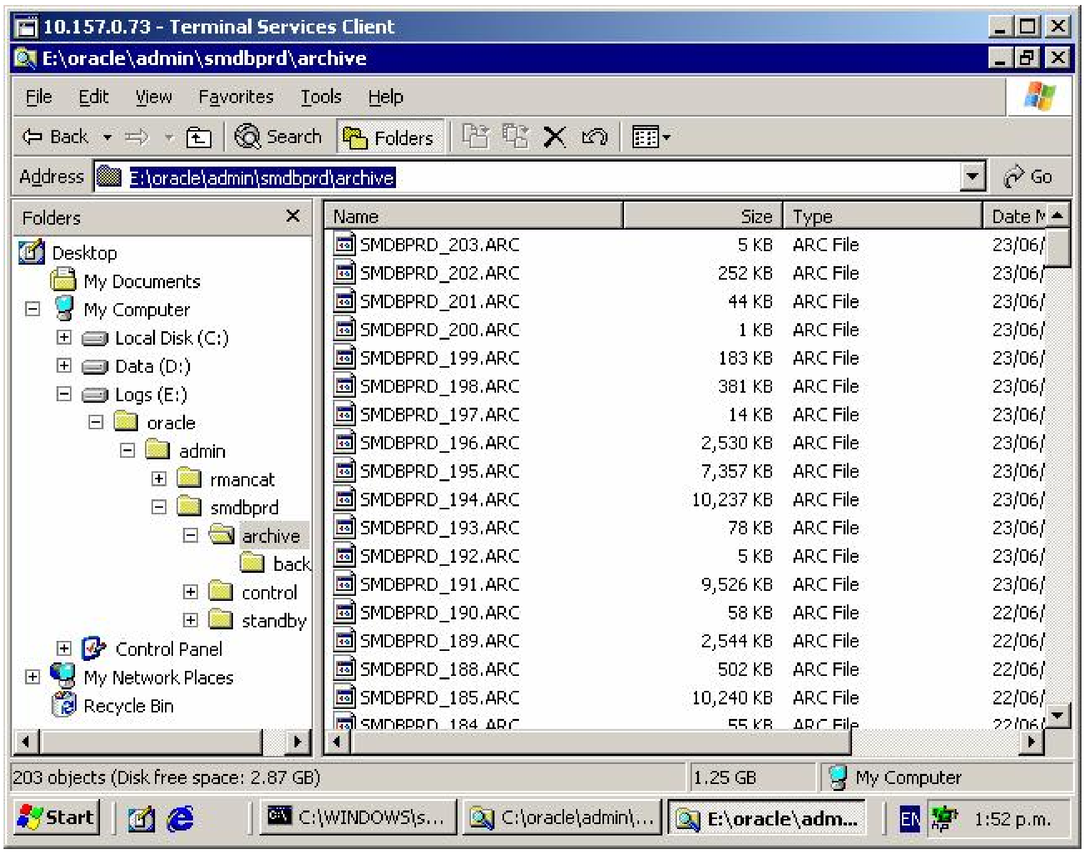
Failover to Logical Standby Database

Failover is a scenario when the Primary Database is lost for an indefinite period of time and the Business makes a decision to activate the Standby database as Primary and switching all application processing to the new Primary.

During Failover, care must be taken that upto the last log available on the standby host is applied in the Logical Standby Database before activating it, otherwise some latest data could be lost.

Following is the way to find out whether the last log has been applied or not.

The archive log directory shows that the last log generated on the Logical Standby is numbered 203.



```
10.157.0.73 - Terminal Services Client
E:\oracle\admin\smdbprd\archive
Address  E:\oracle\admin\smdbprd\archive

Name               Size       Type      Date M
SMDBPRD_203.ARC    5 KB       ARC File  23/06/
SMDBPRD_202.ARC    252 KB     ARC File  23/06/
SMDBPRD_201.ARC    44 KB      ARC File  23/06/
SMDBPRD_200.ARC    1 KB       ARC File  23/06/
SMDBPRD_199.ARC    183 KB     ARC File  23/06/
SMDBPRD_198.ARC    381 KB     ARC File  23/06/
SMDBPRD_197.ARC    14 KB      ARC File  23/06/
SMDBPRD_196.ARC    2,530 KB   ARC File  23/06/
SMDBPRD_195.ARC    7,357 KB   ARC File  23/06/
SMDBPRD_194.ARC    10,237 KB  ARC File  23/06/
SMDBPRD_193.ARC    78 KB      ARC File  23/06/
SMDBPRD_192.ARC    5 KB       ARC File  23/06/
SMDBPRD_191.ARC    9,526 KB   ARC File  23/06/
SMDBPRD_190.ARC    58 KB      ARC File  22/06/
SMDBPRD_189.ARC    2,544 KB   ARC File  22/06/
SMDBPRD_188.ARC    502 KB     ARC File  22/06/
SMDBPRD_185.ARC    10,240 KB  ARC File  22/06/

203 objects (Disk free space: 2.87 GB)   1.25 GB   My Computer
```

But the following SQL shows that 201 & 202 are current which means 203 has not been applied.

```
SQL> column trd format 99
SQL> select thread# trd, sequence#,
  2  first_change#, next_change#,
  3  dict_begin beg, dict_end end,
  4  to_char(timestamp, 'hh:mi:ss') timestamp,
  5  (case when l.next_change# < p.read_scn then 'YES'
  6  when l.first_change# < p.applied_scn then 'CURRENT'
  7  else 'NO' end) applied  from dba_logstdby_log l, dba_logstdby_progress p
  8  order by thread#, first_change#;

TRD  SEQUENCE# FIRST_CHANGE# NEXT_CHANGE# BEG END TIMESTAM APPLIED
--- ---------- ------------- ------------ --- --- -------- -------
  1       195     46315707     46321378 NO  NO  11:51:08 YES
  1       196     46321378     46321767 YES YES 11:50:07 YES
  1       197     46321767     46321828 NO  NO  11:48:36 YES
  1       198     46321828     46323667 NO  NO  12:33:37 YES
  1       199     46323667     46324624 NO  NO  01:12:07 YES
  1       200     46324624     46324625 NO  NO  01:12:07 YES
  1       201     46324625     46324753 NO  NO  01:12:45 CURRENT
  1       202     46324753     46326106 NO  NO  01:46:33 CURRENT

8 rows selected.
```

Now if you try to register log 201 & log 202 in the logical standby database you get the following messages:

```
SQL> alter database register logical logfile
```

```
  2  'E:\oracle\admin\smdbprd\archive\smdbprd_201.arc';
alter database register logical logfile
*
ERROR at line 1:
ORA-01339: logfile is too old


SQL> alter database register logical logfile
  2  'E:\oracle\admin\smdbprd\archive\smdbprd_202.arc';
alter database register logical logfile
*
ERROR at line 1:
ORA-01289: cannot add duplicate logfile
```

Now register the last log 203

```
SQL>  alter database register logical logfile
  2  'E:\oracle\admin\smdbprd\archive\smdbprd_203.arc';

Database altered.
```

Now if we run the previous sql to find out the latest status, it shows log 203 as current.

```
SQL> column trd format 99
SQL> select thread# trd, sequence#,
  2  first_change#, next_change#,
  3  dict_begin beg, dict_end end,
  4  to_char(timestamp, 'hh:mi:ss') timestamp,
  5  (case when l.next_change# < p.read_scn then 'YES'
  6  when l.first_change# < p.applied_scn then 'CURRENT'
  7  else 'NO' end) applied  from dba_logstdby_log l, dba_logstdby_progress p
  8  order by thread#, first_change#;

TRD  SEQUENCE# FIRST_CHANGE# NEXT_CHANGE# BEG END TIMESTAM APPLIED
--- ---------- ------------- ------------ --- --- -------- -------
  1       195      46315707     46321378 NO  NO  11:51:08 YES
  1       196      46321378     46321767 YES YES 11:50:07 YES
  1       197      46321767     46321828 NO  NO  11:48:36 YES
  1       198      46321828     46323667 NO  NO  12:33:37 YES
  1       199      46323667     46324624 NO  NO  01:12:07 YES
  1       200      46324624     46324625 NO  NO  01:12:07 YES
  1       201      46324625     46324753 NO  NO  01:12:45 YES
  1       202      46324753     46326106 NO  NO  01:46:33 YES
  1       203      46326106     46326125 NO  NO  01:51:14 CURRENT

9 rows selected.
```

Also the applied_scn & newest_scn shows the same values:

```
SQL> select applied_scn,newest_scn from dba_logstdby_progress;

APPLIED_SCN NEWEST_SCN
----------- ----------
   46326124   46326124
```

So, now the Sql apply can be stopped in the Logical Standby and it can be activated to become the new Primary:

```
SQL> alter database stop logical standby apply;

Database altered.

SQL> select database_role from v$database;

DATABASE_ROLE
------------------------
LOGICAL STANDBY

SQL> alter database activate logical standby database;

Database altered.

SQL> select database_role from v$database;

DATABASE_ROLE
------------------------
PRIMARY

SQL>
```

How to address ORA-332 error when registering partial archive log into the Logical Standby after Primary Server goes down

When using logical standby database, if the primary database server is shut down or crashes without first shutting down the primary database, partial archive logs on the standby database cannot be registered and will give ORA-332 error.

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
'E:\oracle\admin\smdbprd\archive\smdbprd_203.arc';
ALTER DATABASE REGISTER LOGICAL  LOGFILE
'E:\ORACLE\ADMIN\SMDBPRD\SMDBPRD_203.ARC'
*
ERROR at line 1:
ORA-332: archived log is too small - may be incompletely archived
ORA-334: archived log: 'E:\ORACLE\ADMIN\SMDBPRD\SMDBPRD_203.ARC'
```

Oracle Metalink Document 233253.1 gives the following reason for this problem:

Cause

This issue is related to the TCP/IP keepalive parameter that is set at the operating system level. The keepalive interval determines how often a packet is sent from one server to another to determine if the connection is still alive. The Remote File Server (RFS)
process on the standby uses the TCP/IP keepalive mechanism to determine if the primary server has disconnected or if there is just a brief network problem.

As long as the RFS process on the standby database thinks that the primary database server is still available, the archive log will on the standby not be closed gracefully. If the OS keepalive packet sent from the standby to the primary is not returned, then the standby server will realize that the primary server is gone and the RFS process will close the archive log gracefully. Once the archive log is closed gracefully, it can be registered successfully. If the archive log is registered before the file is closed, then the ORA-332 occurs.

Fix

To reduce the amount of time it takes for the archive log to close gracefully once the primary server has gone down, decrease the TCP/IP keepalive interval at the OS level on the standby server. This effectively reduces the amount of time that the RFS process
will wait before determining that the connection to the primary server is lost. Once the keepalive time has expired and the archive log has been closed, then the archive log can be registered successfully.

To change tcp_keepalive_interval on Solaris, do the following as root:

```
# ndd /dev/tcp tcp_keepalive_interval 7200000
```

The unit being milliseconds, this number indicates 2 hours.
To change this value you use ndd with the '-set' flag, e.g.: for 3 minutes

```
  # ndd -set /dev/tcp tcp_keepalive_interval 180000
```

To implement the TCP/IP KeepAlive mechanism in Windows NT, click:

```
 Start > Run > Regedit.exe
In the Registry, navigate to:
  HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/Tcpip/Parameters

  Edit - Add value: KeepAliveTime (REG_DWORD)
    Decimal  -> 180000 (for example = 3 minutes)
```

The parameter will not be present the first time you modify the parameter, so you will need to add it. The default value is 2 hours.

Fixing stoppage of Logical Standby Apply

Under few conditions, Sql Apply will stop in Logical standby. Following is the way to restart the Logical Standby Apply

Giving sysdba grants to a user in Primary will stop Sql Apply in the Logical Standby. To fix the stopped sql apply, the Logical Standby database can be temporarily taken out of the Guarded Status and the sql can be given manually on the Standby and then it can be put back to Guarded status.

dba_logstdby_events view in Logical Standby shows that sql apply has stopped on the event 'grant sysdba to rman':

```
SQL> select * from dba_logstdby_events;
.
.
TIME             COMMIT_SCN    CURRENT_SCN
--------------    ---------------        ---------------
EVENT
--------------------------------------------------------------------------------

STATUS
--------------------------------------------------------------------------------
```

```
06/21 16:55:04

ORA-16111: log mining and apply setting up


TIME           COMMIT_SCN     CURRENT_SCN
-------------- --------------- ---------------
EVENT
--------------------------------------------------------------------------------

STATUS
--------------------------------------------------------------------------------

06/21 16:55:08       44778824        44778822
grant sysdba to rman

ORA-01031: insufficient privileges
```

So, the remedial action would be to stop the logical standby apply, bypass Guard, give grant manually, re-enable Guard, skip the failed transaction and re-start the logical standby apply:

```
SQL> alter database stop logical standby apply;

Database altered.

SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;

PL/SQL procedure successfully completed.

SQL> grant sysdba to rman;

Grant succeeded.

SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;

PL/SQL procedure successfully completed.

SQL> SELECT XIDUSN, XIDSLT, XIDSQN FROM DBA_LOGSTDBY_EVENTS
  2  WHERE EVENT_TIME = (SELECT MAX(EVENT_TIME) FROM DBA_LOGSTDBY_EVENTS);

     XIDUSN     XIDSLT     XIDSQN
--------------- --------------- ---------------
          4          38        1860

SQL> EXECUTE DBMS_LOGSTDBY.SKIP_TRANSACTION(4,38,1860);

PL/SQL procedure successfully completed.

SQL> alter database start logical standby apply;

Database altered.

SQL>
```

Logical Standby Monitoring

Logical Standby Monitoring includes two aspects:

- Check whether Logical Standby Apply is running (Monitoring the running of Sql Apply)
- Check, even if Logical Standby Apply is running, whether the transactions are not advancing in the Standby database and the Standby database has started lagging behind the Primary beyond the SLA agreed threshold (Logical Standby Lag Monitoring)

Monitoring the running of Sql Apply

The following script can find out whether Sql Apply is running in the Logical Standby. This is useful when Log Standby Apply has been stopped for some reason. It looks at the v$logstdby view. If Logical Standby Apply is running there will be 10 rows in the view v$logstdby with the following display or else it will return 'no rows selected' when Apply doesn't run:

```
SQL> select * from v$logstdby;

   SERIAL# LOGSTDBY_ID PID          TYPE                         STATUS_CODE
---------- ----------- ------------ ---------------------------- -----------
STATUS
--------------------------------------------------------------------------------
  HIGH_SCN
----------
         7          -1 12520        COORDINATOR                        16116
ORA-16116: no work available


         5           0 12522        READER                             16116
ORA-16116: no work available


         1           1 12524        BUILDER                            16116
ORA-16116: no work available
   1669283

         1           2 12526        PREPARER                           16116
ORA-16116: no work available
   1669258

         1           3 12528        ANALYZER                           16116
ORA-16116: no work available
   1669259

         1           4 12530        APPLIER                            16116
ORA-16116: no work available
   1669214

         1           5 12532        APPLIER                            16116
ORA-16116: no work available
   1669216

         1           6 12534        APPLIER                            16116
ORA-16116: no work available
   1669257

         1           7 12536        APPLIER                            16116
```

```
ORA-16116: no work available
  1669259

      1         8 12538       APPLIER                          16116
ORA-16116: no work available
  1669172


10 rows selected.

SQL>
```

Script : Check_logical_standby_apply_running.sh

This script checks whether Sql Apply is running in Logical Standby:

```
#!/bin/ksh
set -x
# Script to check the Logical Standby Apply is running

PATH=$PATH:/usr/local/bin
export PATH

ORACLE_SID=smdbprd_stdby;     export ORACLE_SID
ORAENV_ASK=NO
. /usr/local/bin/oraenv
PATH=$PATH:$ORACLE_HOME/bin
export PATH

USER=sys;                 export USER
PASS=password                        ; export PASS

#logical_standby_check
sqlplus  > /tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst 2>/dev/null <<!
$USER/$PASS@$ORACLE_SID as sysdba
set heading off
set numwidth 15
select type,status,high_scn from v\$logstdby;
exit
!

logical_standby_check=`awk '/rows/ { print $1 } '
/tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst `
echo $logical_standby_check

#database role check
sqlplus  > /tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst 2>/dev/null <<!
$USER/$PASS@$ORACLE_SID as sysdba
set heading off
select database_role from v\$database;
exit
!

database_role_check=`awk '/LOGICAL/ { print $1 $2} '
/tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst `
echo $database_role_check
```

```
if [ $logical_standby_check = "no"  ]; then
  if [ $database_role_check = "LOGICALSTANDBY" ]; then
echo  "ERROR SMDBPRD Logical Standby Apply is not running" >>
/tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst
date>> /tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst
echo "At Standby host, Please check whether logical standby apply is enabled otherwise enable it." >>
/tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst
echo "To activate log transfer from Primary to Secondary, Please switch logfile twice at Primary host."
>> /tmp/"$ORACLE_SID"_logical_standby_apply_check_tmp.lst
  fi
fi
```

Logical Standby Lag Monitoring

Logical Standby Lag Monitoring is different from the Physical Standby Lag monitoring.

In case of a Physical Standby scenario, as the Primary & Standby database maintain the same log thread, a log gap can be monitored by running the following sql on both the databases and comparing the difference between the Primary & Standby sequences against a threshold:

```
select sequence# from v$log_history where recid = ( select max(recid) from v$log_history);
```

But as Logical Standby is an entirely different database from the Primary, this concept can not be applied there.

So the check for log gap between a Primary & it's Logical Standby has to be performed by a script similar to following one:

Script : Check_logical_standby_log_gap.sh

```
#!/bin/ksh
# Script to check the log gap between the Primary and the Standby database

integer log_seq_done
integer log_seq_to_be_done
integer log_gap

PATH=$PATH:/usr/local/bin
export PATH

ORACLE_SID=smdbprd_stdby;    export ORACLE_SID
PATH=$PATH:/cygdrive/c/oracle/ora92/bin
export PATH

USER=sys;                 export USER
PASS=password;            export PASS

#find log_seq_done
sqlplus -s  > /tmp/"$ORACLE_SID"_tmp.lst 2>/dev/null <<!
$USER/$PASS@$ORACLE_SID as sysdba
set heading off
SELECT max(SEQUENCE#)
  FROM DBA_LOGSTDBY_LOG L, DBA_LOGSTDBY_PROGRESS P
  where L.NEXT_CHANGE# < P.READ_SCN;
exit
```

```
!

log_seq_done=`awk '/ / {print $1} ' /tmp/"$ORACLE_SID"_tmp.lst `
echo $log_seq_done

#find log_seq_to_be_done
sqlplus -s  > /tmp/"$ORACLE_SID"_tmp.lst 2>/dev/null  <<!
$USER/$PASS@$ORACLE_SID as sysdba
set heading off
SELECT max(SEQUENCE#)
     FROM DBA_LOGSTDBY_LOG L, DBA_LOGSTDBY_PROGRESS P
   where L.NEXT_CHANGE# >= P.READ_SCN
     or L.FIRST_CHANGE# >= P.APPLIED_SCN;
exit
!

log_seq_to_be_done=`awk ' / / {print $1} ' /tmp/"$ORACLE_SID"_tmp.lst `
echo $log_seq_to_be_done

echo "Log Sequence Done is $log_seq_done " > /tmp/"$ORACLE_SID"_tmp.lst
echo "Log Sequence To Be Done is $log_seq_to_be_done " >> /tmp/"$ORACLE_SID"_tmp.lst

let log_gap=$log_seq_to_be_done-$log_seq_done
echo $log_gap

if [ $log_gap -ge 5 ]; then
echo "ERROR SMDBPRD Standby is trailing more than threshold">>/tmp/"$ORACLE_SID"_tmp.lst
date>> /tmp/"$ORACLE_SID"_tmp.lst
echo "At Standby host, Please check whether logical standby apply is enabled otherwise enable it." >>
/tmp/"$ORACLE_SID"_tmp.lst
echo "To activate log transfer from Primary to Secondary, Please switch logfile twice at Primary host."
>> /tmp/"$ORACLE_SID"_tmp.lst
```

How to determine the Gap between the sent logs and the received logs

To find out the gap between the sent logs and the received logs, following script can be run in the Primary

```
SELECT MAX(R.SEQUENCE#) LAST_SEQ_RECD, MAX(L.SEQUENCE#) LAST_SEQ_SENT FROM
 V$ARCHIVED_LOG R, V$LOG L WHERE
 R.DEST_ID=2 AND L.ARCHIVED='YES';

LAST_SEQ_RECD LAST_SEQ_SENT
------------------------ ------------------------
         80                     90
```

What to do if tables become out of sync

Tables can become out of sync with the primary for any one of the following reasons:

1. Having the SQL Apply engine skip transactions without properly issuing a compensating transaction.

2. Providing users improper access to objects being maintained by SQL Apply.
3. Performing unsupported operations on the primary.
4. Modifying user data as the SYS user.

In a situation like this, it is best to use DBMS_LOGSTDBY.INSTANTIATE_TABLE procedure to synchronize the tables in the Logical Standby with the Primary.

First stop SQL Apply on the logical standby.

```
SQL> alter database stop logical standby apply;
```

Execute dbms_logstdby.instantiate_table procedure in the standby

```
SQL> exec dbms_logstdby.instantiate_table('SCOTT','EMP','PRIMARY');
```

The first argument is the schema owner of the table, the second argument is the name of the table to be instantiated and third is the name of the dblink to the primary.

Should the tables contain referential integrity constraints, then the constraints have to be dropped and after the instantiation of the tables the constraints could be rebuilt.

How to add index or materialized views on the tables in Logical Standby

Should the Logical Standby be used for Query processing, then additional indexes and/or materialsed views could be created in the Standby for better query performance.

Following is the way to do that:

```
SQL> desc dept

 Name                          Null?    Type

 -------------------------    -------- -------------------

 DEPTNO                                 NUMBER(2)

 DNAME                                  VARCHAR2(14)

 LOC                                    VARCHAR2(13)

SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;

PL/SQL procedure successfully completed.

SQL> create index deptno_idx on dept(DEPTNO);

Index created.

SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

```
PL/SQL procedure successfully completed.
```

Skipping Sql in Logical Standby

Should there be a requirement to skip a DDLs & DMLs for a particular schema object, following steps are done:

```
SQL> alter database stop logical standby apply;
SQL> execute dbms_logstdby.skip('SCHEMA_DDL', 'SCOTT', 'DEPT');
SQL> execute dbms_logstdby.skip('DML', 'SCOTT', 'DEPT');
SQL> alter database start logical standby apply;
```

To unskip, dbms_logstdby.unskip procedure should be used.

Oracle's suggestions for Standby over the Local Area Network(LAN)/Metropolitan Area Network(MAN)/Wide Area Network(WAN)

Oracle has done some very good testing with Network Bandwidth, Latency and Return Trip Time(RTT) and given best practice suggestions for the Standby implemented over LAN, MAN & WAN (Ref : Oracle 9i Data Guard: Primary Site and Network Configuration best practices).

For LAN the recommendation is:

- Use Maximum Protection or Maximum Availability modes for Zero data loss
- For very good performance and minimal risk of transaction loss in the event of a disaster, use Maximum Performance Mode, with LGWR ASYNC and 10MB LSN Async buffer (AYNC=20480).

For MAN
- Use Maximum Protection or Maximum Availability modes for Zero data loss

For WAN
- For very good performance and minimal risk of transaction loss in the event of a disaster, use Maximum Performance Mode, with LGWR ASYNC and 10MB LSN Async buffer (ASYNC=20480).
- Set SDU=32k for Oracle Net Connections between Primary & Standby
- Use SSH port forwarding with compression with a large RTT when using Maximum Performance mode

Troubleshooting & Tuning SQL APPLY (Ref: Oracle10g Data Guard SQL Apply Troubleshooting & Oracle 9i Data Guard Sql Apply Best Practices)

- Tune the memory allocated to Sql Apply

Sql Apply uses shared pool for its operation. Make sure Shared Pool is not constrained. If required increase Shared Pool. Also the memory allocated to Sql Apply can be increased by running the following command

```
SQL> exec dbms_logstdby.apply_set('MAX_SGA',50);
```

- Tune the number of parallel processes (apply slaves) serving Sql Apply.

This is dependent on parallel_max_server parameter on the Standby.

To Change the value of parallel processes, the following command can be used

```
SQL> exec dbms_logstdby.apply_set('MAX_SERVERS',9);
```

- Check for ITL pressure

If in the alert log of the Standby the following warning message appears that would indicate problem is due to ITL waits

```
WARNING: the following transaction makes no progress
WARNING: in the last 30 seconds for the given message!
WARNING: xid =
0x0006.005.000029fa cscn = 2152982, message# = 2, slavid = 17
```

Oracle keeps note of which rows are locked by which transaction in an area at the top of each data block known as the 'interested transaction list'. The number of ITL slots in any block in an object is controlled by the INITRANS and MAXTRANS attributes. If there is no free 'ITL', then ITL waits will occur.

Following Sql can be used to report which objects have been subjected to ITL pressure

```
SQL> select segment_owner, segment_name, segment_type
    from v$segment_statistics
    where statistic_name = 'ITL waits'
    and value > 0
    order by value;
```

This statement will report all database segments that have had ITL pressure at some time since the instance was last started.

To resolve ITL pressure, following steps could be performed

```
SQL> alter database stop logical standby apply;
SQL> execute dbms_logstdby.guard_bypass_on;
SQL> alter table emp initrans <#initrans>;
SQL> execute dbms_logstdby.guard_bypass_off;
SQL> alter database start logical standby apply;
```

- Check for Long Running Transactions

Sometimes, if the following message appears in the alert log of the Logical Standby, it’s an indication that SQL Apply engine is executing a single SQL statement for an extended period.

```
WARNING: the following transaction makes no progress
WARNING: in the last 30 seconds for the given message!
WARNING: xid =
0x0016.007.000017b6 cscn = 1550349, message# = 28, slavid = 1
knacrb: no offending session found (not ITL pressure)
```

But, this is not for ITL Pressure, most of the time it is for Full table Scans.

Check whether the Logical Standby is experiencing Full table Scan which will affect Sql Apply performance.

A batch update of a table in the primary database translates into multiple updates in the logical standby database. Thus, a batch update in the primary database on a table that has no primary or unique key constraints may do multiple full table scans in the logical standby database and result in poor performance.

If a single delete statement on the primary deletes 1000 rows that translates to 1000 individual delete statements on the logical standby. If SQL apply operations are performing full table scans during those 1000 delete statements then performance will be significantly impacted. Adding indexes on the effected tables could help.

Also, long running transactions could also be the result of DDL operations being replicated to the standby database, such as the creation or rebuild of and index in the Primary.

- Decide on the Transaction_consistency parameter of DBMS_LOGSTDBY.APPLY_SET procedure.

The three values of Transaction_consistency are FULL, READ_ONLY & NONE.

FULL is the default which applies the sql statements on the Logical Standby in the same order as it has committed in the Primary. This option has lowest performance.

READ_ONLY option gives better performance by way applying sql statements in the Standby differently to the order of the commits of sql statements in Primary but generates a read consistency view of data for any report/query run on the Standby.

NONE option applies Sql statements in the Standby differently to the order of commits of sql statements in Primary and also doesn't maintain read consistency for the reports run on Standby. This option gives best performance.

- Run DBMS_STATS in the Logical Standby

As DMLs run and Statistics become stale, Oracle advises to run dbms_stats in Logical Standby because though Logical Standby is logically equivalent to Primary, the Sql Apply operation could execute the workload in Standby in a different way to that in Primary.

Enhancements in Oracle 10G Logical Standby

- Creation of Logical Standby without any downtime of the Primary

In Oracle 9i v9.2, creation of a logical standby needs a cold backup of the Primary or if it is to be from the online backup then either the Primary has to be quiesced or special procedure as per Oracle Metalink document 278371.1 has to followed without quiesceing the database but a short period of shutdown is still needed.

Oracle 10G has removed this restriction by introducing the concept of Logical Standby Controlfile which allows on-line backup of the Primary database to be used for creation of Logical Standby

Below is the syntax to create a Logical Standby Controlfile:

```
SQL>alter database create logical standby controlfile as '/tmp/logical_stdby.ctl';
```

- Usage of Standby Redo Log(SRL) in Logical Standby

In Oracle 9i v9.2, usage of Standby redo log file was limited to Physical Standby.

Oracle 10G supports the use of Standby Redo Log file for Logical Standby. In Oracle 10G Logical Standby, Log Writer or the Archiver can transfer data to Standby Redo log, thereby eliminating the need to register Partial Archived redo log after the Primary database crash.

The main advantage of Standby Redo Logs is that every entry written into the Online RedoLogs of the Primary Database is transferred to the Standby site and written into the Standby Redo Logs at the same time, therefore, the probability of data loss on the Standby Database is greatly reduced.

With the introduction of support for standby redo log files, it is now possible to have a logical standby database be part of a Data Guard configuration running in maximum protection mode.

- Oracle 10G introduces Real-Time Apply feature of Standby

In Oracle 9i Data Guard, the redo data gets archived from Standby Redo Logs to Archive Redo logs at the Standby database and then they are applied to the Standby Database.

In Oracle 10G, with the Real Time Apply feature, the redo data can be applied to Standby Database (Physical or Logical) as soon as redo data gets written in the Standby Redo Logs.

To enable Real-time Apply feature in the Logical Standby database, Apply process is started using the following command:

```
SQL> alter database start logical standby apply immediate;
```

- Oracle 10G has introduced 'prepare to switchover' command for switchover & switchback optimisation.

Oracle 10G 'alter database prepare switchover to logical standby' command tells Primary to be prepared for the role reversal.

'alter database prepare switchover to primary' command tells Logical Standby to start building Log Miner Dictionary and sending it to the new Standby before the switchover.

After initial preparation phase, switchover can be initiated.

- Oracle 10G has simplified skip failed transaction feature

In Oracle 9i Data Guard, when a transaction is skipped at the standby, the procedure is to manually apply the failed transaction, then finding out the XIDUSN, XIDSLT, and XIDSQN of the failed transaction and doing an explicit skip of that transaction by doing dbms_logstdby.skip(XIDUSN,XIDSLT,XIDSQN) before starting Logical Standby Apply.

Oracle 10G has simplified procedure of skipping the last failed transaction & restarting Sql Apply with the following command:

```
SQL> alter database start logical standby apply skip failed transaction;
```

- Oracle 10G LNS buffer has been raised from 10MB to 50MB

When using LGWR ASYNC mechanism for redo data transport, the Log writer uses LogWriter Network Server (LNS) process to transmit the data to the Standby. In Oracle 9i Data Guard, the max size of LNS buffer is 10MB.

If Network is busy or if the Network has low bandwidth and/or high latency and/or there is intense log generation in the Primary, then LNS ASYNC Buffer could become and remain filled up for some time and in that case LGWR could stall and transfer control to Archiver to perform log shipping. LGWR will take its role back again when LNS Buffer becomes free.

```
In Oracle 10G, the max size of LNS Buffer has been increased to 50MB.
```

- Integration of Oracle Flashback with Oracle 10G Logical Standby

To prevent a corruption of the Primary be propagated to the Standby should a delay in Sql Apply be intended, then either DELAY option in LOG_ARCHIVE_DEST_2 or APPLY_DELAY parameter of the SQL APPLY engine has to be used.

Oracle 10G has simplified this scenario by integrating Oracle Flashback with Logical Standby which can be used to rewind the Standby database to a time before the corruption.

- Simplified identification of Applied Archive logs that can be purged in Logical Standby

In Oracle 9i, the following script is needed to run in Standby to identify the logs that been applied:

```
SQL> SELECT FILE_NAME FROM DBA_LOGSTDBY_LOG WHERE
     NEXT_CHANGE# < (SELECT READ_SCN FROM DBA_LOGSTDBY_PROGRESS);
```

Where as in Oracle 10G the following command will show that

```
SQL> select from DBA_LOGMNR_PURGED_LOG;
```

After deleting the logfiles using an O.S. utility, clear the information about the purged logfiles from that view by running

```
SQL> EXEC DBMS_LOGSTDBY.PURGE_SESSION;
```

In conclusion, we have seen Oracle Data Guard Logical Standby is one step ahead of Physical Standby. With continued improvements in functionality and data type support, implementing Dataguard Logical standby Databases will become the standard for effective Disaster Recovery and Reporting environments..

Ref: The supporting materials in this paper are numerous Oracle documents and white papers from Oracle Metalink, OTN & Oracle Data Guard Documentations.