Ensuring Your Physical Standby is Usable

Michael S. Abbey
Pythian
Ottawa ON, Canada

Keywords

Physical standby Disaster Recovery High Availability Standard and Enterprise Edition

Introduction

In many nooks and crannies of IT, it's all about planning and TLC (tender loving care). IT is a hotbed for situations commonly classified as manifestations of *Murphy's Law*. Wikipedia attributes this old adage to a mountaineering work written by John Sack – *anything that can possible go wrong, does*. Disaster recovery is a hotbed for just that – D I S A S T E R. The ongoing planning that goes into a strong backup solution is derailed without testing the reliability of the solution(s) of choice. The set of parentheses around the "s" in the last sentences bind a fundamental approach to backup (more than one solution) → multiple solutions not only complement one another but also remove a single point of failure.

Oracle's physical standby database feature appeared in the early 1990's with the release of Oracle7 in February of 1992. I attended the launch in New York City and the event experienced a micro disaster when one of the presenter's microphones did not work. AV personnel catapulted a replacement (and tested) device into the pit and the festivities continued. The need to verify the usability of all your backups presents itself whether you use any Oracle or 3rd party solution. This paper and the accompanying presentation at COLLABORATE14 will offer advice and methods for ensuring the physical standby database you have built are ready to be sprung into action.

The physical standby (often referred to by a more familiar trade name of *data guard*) is the Rolls Royce of disaster recovery and, once DBAs wrap their hands around this solution it jumps out as "the best." Unless a DBA or manager can emphatically answer "yes" to the following questions, with no hesitation or awkward "uh …" and "hmmm …", you cannot rely on your physical standby.

> *Do we carry out frequent testing of the physical standby from both perspectives … Oracle and operating system?*
> *Is the infrastructure in place to support a role reversal from standby to primary?*

At the end of this paper there is a checklist presented that should define the utmost TLC for all your physical standby databases. This is a perfect time for one to initiate all the calisthenics required to ensure your physical standby is usable from the following perspectives:

- The database itself
- Management scripts and routines required to bullet-proof a primary database site that ensure the transition to production standby use is transparent to the applications

The following paper discusses what needs to be done to keep surprises from creeping up after a switchover to a standby site. Most DBAs are fluent in the database-centric tasks required to ensure the physical standby is usable. This paper not only concentrates on going the distance to ensure the database itself is able to assume the role of a primary, it also goes the extra distance to avoid situations all too common after a switch to a standby site without sufficient planning:

Figure 1: Renata's latest failover; hopefully not her last

Let's get started by looking at a handful of pitfalls; these traps one can easily fall into. It is all too easy to pat yourself on the back for being so thorough during setup, then gasp when a physical standby is not usable. The balance of this paper plus accompanying presentation will arm you with a suite of queries and procedures to ensure your physical standby site is usable.

PITFALL #1

Regardless of what version of Oracle one is running and what flavour of options has been purchased, there are two processes that ensure the physical standby is ready to come to life as the primary site:

1. Log transport—the act of delivering transaction information to a standby site via the ARCH or LGWR process
2. Log application—the act of applying transaction information to a standby database via a manual or automated recovery process

Do not assume your physical standby site is usable based on confirmation that log transport and application services are working fine.

If running enterprise edition, we can set up log transport and application services to be handled by Oracle; if not we have to implement homegrown scripts to accomplish the same activities. To check up on what archived redo has not been received on the standby site, the following query from the primary site will provide the desired information:

```
select local.thread#, local.sequence#
  from (select thread#, sequence#
          from v$archived_log
         where dest_id = 1) local
 where local.sequence# not in
      (select sequence#
         from v$archived_log
        where dest_id=2
          and thread# = local.thread#);
```

Its sister query is as follows which will display the latest log sequence number that has been applied on the standby:

```
select thread#, max(sequence#) as "Last applied log"
from v$log_history
group by thread#;
```

These two queries in action could resemble:

```
SQL> select local.thread#, local.sequence#
  2    from (select thread#, sequence#
  3            from v$archived_log
  4           where dest_id = 1) local
  5   where local.sequence# not in
  6         (select sequence#
  7            from v$archived_log
  8           where dest_id=2
  9             and thread# = local.thread#);

   THREAD#  SEQUENCE#
---------- ----------
         1     100798
         1     100799
         1     100800
         1     100801
         1     100802
5 rows selected.
SQL> select thread#, max(sequence#) as "Last applied log"
  2  from v$log_history
  3 group by thread#;

   THREAD# Last applied log
---------- ----------------
         1           100796
```

The above two queries tell us:

- Redo log sequences 100798 through100802 have been successfully received on the standby site
- The last applied log sequence is 100796

These two metrics tell us that the standby is lagging the master by as much as five archived redo logs. Regardless of the size of and the nature of transaction activity reflected in those five archived redo logs that are unapplied, suffice to say the standby is not current.

PITFALL #2

A popular word in the physical standby arena when running enterprise edition is *managed recovery*. This process sits in anticipation of the arrival of new archived redo. It springs into action and applies that redo when discovered. Take the guess work out of the equation by checking on the status of the managed recovery process if so enabled.

> *Do not assume managed recovery is running without taking the time to confirm on a regular basis.*

This can be done from the operating system with a simple command as follows:

```
oracle@dlab48.pythian.com--> (pythian) ** Standby **
/home/oracle> ps -fu oracle|grep mrp|grep -v grep
oracle    2597     1  0 11:11 ?        02:21:18 ora_mrp0_pythian
```

The following query will assist determining the status of managed recovery:

```
SQL> select process,status from v$managed_standby;
PROCESS   STATUS
--------- ------------
ARCH      CONNECTED
ARCH      CONNECTED
ARCH      CONNECTED
ARCH      CONNECTED
RFS       IDLE
RFS       IDLE
RFS       IDLE
RFS       IDLE
```

The above output confirms that managed recovery is down. After starting the managed recovery process with a command similar to the following, let's run the identical query again:

```
SQL> alter database recover managed standby database disconnect from session;
Database altered.
SQL> select process,status from v$managed_standby;

PROCESS   STATUS
--------- ------------
ARCH      CONNECTED
ARCH      CONNECTED
ARCH      CONNECTED
ARCH      CONNECTED
RFS       IDLE
RFS       IDLE
RFS       IDLE
RFS       IDLE
MRP0      WAIT_FOR_LOG
9 rows selected.
```

From time-to-time the above query may display a different status for the MRP process of *APPLYING_LOG* which is a good thing. Not only is it there but it is performing its primary task.

PITFALL #3

This is where the heart of this paper stems from. Many DBAs have experienced problems opening a database after a meticulous restore/recovery activity. They have painstakingly applied all available archived redo and look in horror as the following errors are thrown as one attempts to open the database:

```
ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error below
ORA-01194: file 1 needs more recovery to be consistent
```

Readers may be aware of a handful of ways they could get the database open; regardless of that the recovered database is not yet in a state where it is up-to-date and not suffered from any (or close to any) data loss. The exact same issue can occur when trying to activate a standby database. Though not common, the above can derail best laid plans. This leads to the recommendation that follows:

Open your physical standby read only on a regular basis to ensure it can be activated.

Perform the following tasks often to confirm the physical standby's usability:

- Suspend media recovery
- Check the DATABASE_ROLE and OPEN_MODE columns in V$DATABASE
- Open the database read only using the command *alter database open read only;*

After confirmation of the success of the preceding open command, you have just confirmed that an attempt to activate the standby will work. The next listing (with comments) illustrates how this plays out:

```
-- Expecting the following to return PHYSICAL STANDBY and MOUNTED
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY MOUNTED

SQL> alter database open read only;

Database altered.
-- Expected results from next query are PHYSICAL STANDBY and READ ONLY
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY READ ONLY

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.

-- Starting with Oracle Database 10g a simple startup command could be
-- used to mount a physical standby
SQL> startup
ORACLE instance started.

Total System Global Area  409194496 bytes
Fixed Size                  2288968 bytes
Variable Size             331350712 bytes
Database Buffers           71303168 bytes
Redo Buffers                4251648 bytes
Database mounted.
Database opened.

-- Notice how the database startup command returned the standby to
-- the same state it was in when last started - READ ONLY
SQL> select database_role,open_mode from v$database;
```

```
DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY READ ONLY

-- We need to shutdown the standby and open using the "old-fashioned"
-- method
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup nomount
ORACLE instance started.

Total System Global Area  409194496 bytes
Fixed Size                  2288968 bytes
Variable Size             331350712 bytes
Database Buffers           71303168 bytes
Redo Buffers                4251648 bytes
SQL> alter database mount standby database;

Database altered.

-- Now the query against the database open mode returns MOUNTED
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY MOUNTED
```

O/S Perspective

The assortment of tasks operating system specific to support the running primary database must be present on the standby server. Not only that, it is not enough to place them there—one must ensure they are up-to-date. These scripts and text files fall into the following categories:

- The crontab entries
- Backup scripts when tasks are spawned directly from the operating system rather than cloud control
- Scripts assisting management of a site when it is playing the role of a primary including
    - forcing a log switch
    - pushing archived redo to the standby site
    - performing backups
- Scripts assisting management of a site when playing the role of a standby including
    - application of archived redo
    - cleanup of accumulated archived redo

Ensuring completeness

The secret when setting this up is making sure an exact replica of directory contents from each server is positioned on the other, in other words as 2-way structure as shown in the next diagram:

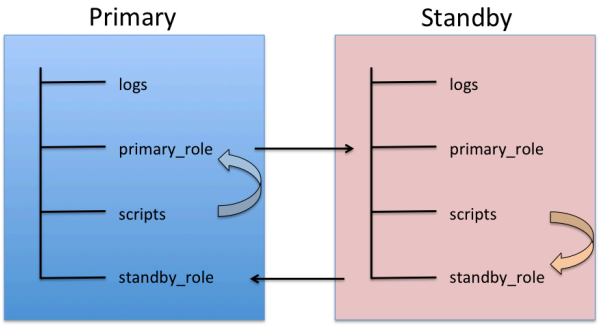

Figure 2: Ensuring completeness

Table 1 shows the repetitive tasks need to be set up as per Figure 1:

Source server	*Target server*	*Copy location*	*to*
Primary	-same-	scripts	primary_role
	Standby	primary_role	primary_role
Standby	-same-	scripts	standby_role
	Primary	standby_role	standby_role

Table 1: Repetitive tasks

This may seem like a daunting task at the outset but need not be thanks to the power of *rsync* in the Linux world and its close competition called *robocopy* on Windows.

On Windows

At one point in time robocopy was part of the Windows Resource Kit Tools but is now bundled with the product. Suppose a directory structure exists on a Windows primary database server to service the needs of many applications:

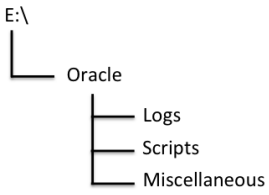

Figure 3: Directories to replicate

Robocopy is just what the doctor ordered, whose help screen is shown next:

```
------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------
  Started : Sun Jan 11 20:10:19 2016
       Simple Usage :: ROBOCOPY source destination /MIR
            source :: Source Directory (drive:\path or \\server\share\path).
         destination :: Destination Dir  (drive:\path or \\server\share\path).
               /MIR :: Mirror a complete directory tree.
    For more usage information run ROBOCOPY /?
****  /MIR can DELETE files as well as copy them !
```

A simple call to robocopy could perform the desired mirroring as follows, the switches easily explained in the robocopy verbose help:

```
robocopy \\sourceserver\share \\destinationserver\share /mir /fft /z /xa:h /w:5
```

On Linux

Nothing could be simpler on linux using the *rsync* command with all its bells and whistles. This standard utility provides a plethora of functionality far beyond the needs of this task at hand. An extract from the linux man page follows:

```
NAME
       rsync — a fast, versatile, remote (and local) file-copying tool
SYNOPSIS
       Local:  rsync [OPTION...] SRC... [DEST]
       Access via remote shell:
         Pull: rsync [OPTION...] [USER@]HOST:SRC... [DEST]
         Push: rsync [OPTION...] SRC... [USER@]HOST:DEST
       Access via rsync daemon:
         Pull: rsync [OPTION...] [USER@]HOST::SRC... [DEST]
               rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
         Push: rsync [OPTION...] SRC... [USER@]HOST::DEST
               rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
       Usages with just one SRC arg and no DEST arg will list the source files
       instead of copying.
DESCRIPTION
       Rsync is a fast and extraordinarily versatile file  copying  tool ...
```

The syntax and punctuation usage is very similar to ssh, output resembling the following—our wanting to sync the contents of the local host ~oracle/scripts with the standby site:

```
oracle@dlab48.pythian.com-->(learning)
/home/oracle/scripts> rsync -av . oracle@dlab49.pythian.com:scripts/
sending incremental file list
./
als.sql
aliases
aliases.db
anacrontab
anthy-conf
ar.sql
asound.conf
crget_primary.sh
crontab_primary
...
...
rman_backup.sh
vimrc
virc
warnquota.conf
```

```
wgetrc

sent 1481795 bytes  received 2371 bytes  2968332.00 bytes/sec
total size is 1475626  speedup is 0.99
```

Wrapup

In some ways we have only scratched the surface of the care and feeding of a physical standby database to ensure when it needs to be sprung into action it will not let you down. In reality, this session has discussed techniques and processes to follow to ensure a physical standby database is usable if and when its turn comes to service application activity.

The following checklist can be used to ensure routines are implemented highlighted by the material in this paper; some additional standby setup tasks are described in the accompanying Powerpoint slides:

[ ] Check and report on status of log shipping
[ ] Check and report on status of log application
[ ] Check and report on arrival of archived redo on standby site
[ ] Check details on last applied archived redo on standby site
[ ] Check lag on standby site
[ ] Test usability of standby by opening read only
[ ] Ensure primary crontab details exist on standby site
[ ] Ensure standby crontab details exist on primary site
[ ] Implement syncing of scripts between primary and standby site
[ ] Implement syncing of scripts between standby and master site

While this paper offered some code suggestions, the mechanisms you implement to ensure your physical standby database is usable will be driven by your preferences and the way you are used to going about doing things.

About the author

Michael S. Abbey is a seasoned presenter at Oracle shows having first cut his teeth on version 3 of the database in the mid-1980s. Michael is a successful author in the Oracle Press series, most recent work being Oracle Database 12c: Install, Configure, and Maintain Like a Professional (summer 2013). He is a team lead with Pythian, a world leader in managed and consulting services to safeguard your data and keep your business-critical systems healthy.

Contact information
(+)1-613-809-4063
abbey@pythian.com
@MichaelAbbeyCAN
www.pythian.com

Ottawa ON, Canada, GMT -5