Database Rolling Upgrade with Transient Logical Standby Database

DOAG Day High Availability

Robert Bialek
Principal Consultant
Who Am I

- Principal Consultant and Trainer at Trivadis GmbH in Munich.
  - MSc in Computer Engineering.

- Focus:
  - Oracle Database High Availability.
  - Database Architecture/Internals.
  - Backup/Recovery.
  - Troubleshooting/Performance Tuning.
  - Linux.

- Trainer for the following Trivadis courses:
  - Oracle Grid Infrastructure, RAC, Data Guard, Oracle HA.
Our company

Trivadis is a market leader in IT consulting, system integration, solution engineering and the provision of IT services focusing on Oracle and Microsoft technologies in Switzerland, Germany, Austria and Denmark. We offer our services in the following strategic business fields:

- Business Intelligence
- Application Development
- Infrastructure Engineering
- Training

Trivadis Services takes over the interacting operation of your IT systems.
With Over 600 Specialists and IT Experts in Your Region

- 14 Trivadis branches and more than 600 employees.
- 200 Service Level Agreements.
- Over 4,000 training participants.
- Research and development budget: CHF 5.0 / EUR 4 million.
- Financially self-supporting and sustainably profitable.
- Experience from more than 1,900 projects per year at over 800 customers.
Technology on its own won't help you. You need to know how to use it properly.
The Problem

Database upgrade (=modifying the data dictionary) might have a big impact on your business:

- **Downtime**, performance regression, potentially new BUGs, etc.
- Fallback is difficult after increasing the `COMPATIBLE` parameter!

Common database upgrade methods – unfortunately no one size fits all solution.

<table>
<thead>
<tr>
<th>In-Place (Direct or Intermediate)</th>
<th>Out-Of-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Upgrade Assistant (DBUA)</td>
<td>Data Pump Export/Import (database size)</td>
</tr>
<tr>
<td>Command Line Upgrade (catctl.pl/catupgrd.sql)</td>
<td>Transportable Tablespaces, ... (#user objects)</td>
</tr>
<tr>
<td>Replication Technology: GoldenGate, (Transient) Logical Standby, ... (complexity/costs)</td>
<td></td>
</tr>
</tbody>
</table>

Installed database options matter!

Most frequently used methods!
The Solution?

Reducing downtimes and/or risks can be a very time consuming, complex and costly task.

Database Rolling Upgrade using Transient Logical Standby (SQL Apply) might be a solution:

- Allows for running different database versions during rolling upgrade window.
  - Gives you more time to test the new version, before activating it.
- In case you already use Data Guard – no additional hardware/storage(∗) footprint, license fees and knowledge(∗) necessary.
- Near zero downtime – reduced to one switchover (two, if you go back).
- Fairly simple method, compared to some other replication technologies.
  - But, with some restrictions.
Agenda

1. Introduction.
2. Rolling Upgrade with Data Guard.
3. Rolling Upgrade with Active Data Guard.
Introduction
**Introduction – Big Picture (1)**

**Preparation Phase:**
- SQL Apply restrictions check
- Software installation/patches
- ...  

- Disable Broker configuration  
- Redo transport on PSTBY(*)  
- ...
Introduction – Big Picture (2)

- **Database Rolling Upgrade with Transient Logical Standby Database**

- **12.2.0.1**
  - As of 12.1.0.1+, redo transport still works!
  - SQL Apply stopped

- **11.2.0.4**
  - DBMS_LOGSTDBY.BUILD
  - **CONVERT TO LOGICAL STANDBY KEEP IDENTITY**
    - EDS on primary and standby
  - SQL Apply

- **11.2.0.4** (LSTBY)

- **DBMS_LOGSTDBY.BUILD**
  - `EDS` on primary and standby
  - `dbupgrade -n #`
  - `foreign_archivelog`

- **4**
- **5**
Introduction – Big Picture (3)

- **Database Rolling Upgrade with Transient Logical Standby Database**
- **13.07.2017**

- **11.2.0.4** PRIMARY
- **11.2.0.4** PRIMARY
- **12.2.0.1** LSTBY
- **12.2.0.1** LSTBY

Redo Transport

SYNCHRONIZED

Monitor `dba_logstdby_events`

- Drop Restore Point (3)
- Raise COMPATIBLE (*)
- Start SQL Apply
- Test, Test, Test, ...

Redo Transport

SYNCHRONIZED
Introduction – Big Picture (4)

- Flashback to Restore Point (3)
- Convert to Physical Standby
- Start from new Oracle Home
- Drop Restore Point (3)
- Increase COMPATIBLE(*)
- Remove EDS objects
- Start Redo Apply (Recreate Broker config.)
Introduction – Big Picture with Multiple Standby Databases

Default: all standby databases belong to the Trailing Group.

Trailing Group Master

**PRIMARY**

11.2.0.4

**PSTBY**

11.2.0.4

Leading Group Master

**LSTDBY**

12.2.0.1

**PSTBY**

12.2.0.1

12.2.0.1

Upgraded first
Restrictions (1)

Some data types lack native redo-based support.

- Support extended with every release. As of 12.2.0.1, the following data types are not natively supported:
  
  • BFILE, ROWID, UROWID, nested tables, objects with nested tables, identity columns, tables only with LOB, LONG, … data type.

- Operations on unsupported objects are automatically skipped.

```sql
SQL> select * from dba_logstdby_unsupported;
```

```sql
SQL> select event, status from dba_logstdby_events order by event_time;
```

<table>
<thead>
<tr>
<th>EVENT</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>create table system.t ...</td>
<td>ORA-16247: DDL skipped on internal schema</td>
</tr>
<tr>
<td>DML on &quot;BIR&quot;.&quot;DEPT_PROJECTS&quot;</td>
<td>ORA-16129: unsupported DML encountered</td>
</tr>
</tbody>
</table>
Restrictions (2)

What if my data types/tables are not supported?

<table>
<thead>
<tr>
<th>Primary (Upgrade Window)</th>
<th>Logical Standby</th>
</tr>
</thead>
<tbody>
<tr>
<td>No DML on those tables on the primary.</td>
<td>Objects remain in SYNC.</td>
</tr>
<tr>
<td><strong>Important</strong> DML executed on those tables.</td>
<td>- Check for Extended Data Support (EDS).</td>
</tr>
<tr>
<td></td>
<td>- Re-insert SQL statements on the standby.</td>
</tr>
<tr>
<td></td>
<td>- Reinstate those tables with Data Pump (small /non-critical segments).</td>
</tr>
<tr>
<td><strong>No important</strong> DML executed in those tables (e.g.: log tables).</td>
<td>DML skipped automatically. No action necessary.</td>
</tr>
</tbody>
</table>
Restrictions (3)

- Table rows in the primary database should be uniquely identified (supplemental logging!)

- SQL Apply performance:
  - Batch updates or deletes done on a primary database result in single row operation on a standby.
  - Direct path inserts are applied using a conventional path insert.
  - Parallel DML transactions are not executed in parallel.

- Some DDLs are skipped at the standby database:
  - Data Guard Concepts and Administration: “Data Type and DDL Support on a Logical Standby Database”.

```sql
SQL> SELECT * FROM DBA_LOGSTDBY_NOT_UNIQUE;
```
Stop the broker and disable the configuration.

```sql
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxPerformance;
DGMGRL> DISABLE CONFIGURATION;
SQL> ALTER SYSTEM SET dg_broker_start=FALSE;
```

For 12.1.0.1+ source configuration, set `log_archive_dest_2` parameter on the standby database (not necessary for 11.2).

```sql
SQL> ALTER SYSTEM SET log_archive_dest_2='service="DB121_SITE1.TRIVADIS.COM"','SYNC...'
```

Create restore point in both databases.

```sql
SQL> CREATE RESTORE POINT pre_upgrade122 GUARANTEE FLASHBACK DATABASE;
```
Rolling Upgrade with DG: Phase 4

- Build a Logminer Dictionary on the current primary into the redo stream.

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

- Convert the standby database to transient logical standby.

```
SQL> STARTUP MOUNT
SQL> ALTER DATABASE RECOVER TO LOGICAL STANDBY KEEP IDENTITY;
SQL> ALTER DATABASE OPEN;
```

- Before starting SQL Apply.

```
EXEC DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT', 'FALSE'); #BUG 12353543 version 11.2
EXEC DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE', 'FALSE');
EXEC DBMS_LOGSTDBY.APPLY_SET('MAX_EVENTS_RECORDED', DBMS_LOGSTDBY.MAX_EVENTS);
EXEC DBMS_LOGSTDBY.APPLY_SET('RECORD_UNSUPPORTED_OPERATIONS', 'TRUE');
```

0% OF LOGMINER DICTIONARY LOADING IS DONE

Keeps DBID and DBNAME
Rolling Upgrade with DG: Phase 4 (Optional)

- If necessary, check for **Extended Data Support** (EDS):
  - `DBA_LOGSTDBY_EDS_SUPPORTED`, `DBA_LOGSTDBY_EDS_TABLES` (11gR2).
  - Available from 10.2.0.4 and 11.1.0.7 onwards (trigger property `fire_once => false`).

![Diagram showing database upgrade process with SQL Apply tool](image)

- **PRIMARY**
  - 11.2.0.4
  - `INSERT INTO dept_projects`

- **LSTBY**
  - 12.2.0.1
  - SQL Apply

- **dept_projects**
  - Base Table – DMLs not applied!

- **SYSLSBY_####**
  - Shadow Table – DMLs applied!
Rolling Upgrade with DG: Phase 4 (Optional)

Setup primary database.

```sql
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE ( -
    table_owner => 'APP', -
    table_name => 'DEPT_PROJECTS');
```

Setup logical standby database.

```sql
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> CREATE DATABASE LINK to_primary CONNECT TO system IDENTIFIED BY <pwd>
    USING 'db_sitel.trivadis.com';
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE ( -
    table_owner => 'APP', -
    table_name => 'DEPT_PROJECTS', -
    p_dblink => 'TO_PRIMARY');
```

Table reinstantiated using Data Pump
From 12.1.0.1+, disabling transport to the standby database not necessary anymore.

```
SQL> STARTUP UPGRADE
/u00/app/oracle/product/12.2.0.1/bin/dbupgrade -n 4
```

Synchronize the standby database.

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

Optionally, change the `COMPATIBLE` parameter now and bounce the instance.

```
SQL> ALTER SYSTEM SET compatible='12.2.0.1.0' SCOPE=SPFILE;
```

```
ALERT: Compatibility of the database is changed from 11.2.0.4.0 to 12.2.0.1.0.
Switching redo format version from 11.2.0.4.0 to 12.2.0.1.0 at change 0
```
Rolling Upgrade with DG: Phase 8

- After successful testing, execute switchover.
  - On the current primary.
    
    SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;

  - On the current standby.
    
    SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL PRIMARY;

- **AFTER STARTUP** trigger does not fire after converting the database roles:
  - Use **AFTER DB_ROLE_CHANGE** event!

- Switchover **does not** cause the oraagent to start/stop services according to the defined role (Grid Infrastructure, Oracle Restart)!
Rolling Upgrade with DG: Phase 9

In case you have already increased the `COMPATIBLE` parameter on the primary database, your standby will hit an error.

- Reason: unmatched compatibility setup (will be solved later).

```
RFS[6]: New Archival REDO Branch: 948879244 Current: 947105021
ORA-00600: internal error code, arguments: [2730], [331], [1], [6], [73], [73], [512], [512], [], [], [], []
```

Flashback and convert the old primary database.

```
SQL> FLASHBACK DATABASE TO RESTORE POINT pre_upgrade122;
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

Increase `COMPATIBLE` on the new standby.

```
SQL> ALTER SYSTEM SET compatible='12.2.0.1.0' SCOPE=SPFILE;
```

Remove all GRPs
Rolling Upgrade with DG: physru.sh

Oracle shell script for rolling upgrade using SQL Apply:


```
physru.sh <user> <prim_tns> <stdby_tns> <prim_name> <stdby_name> <upgrade_version>
```

Requires three consecutive script executions, leaving for you:
- Starting database instances/listeners from new home.
- Environment changes (oratab, password file, …).
- Database upgrade, database services.
- *COMPATIBLE* parameter change at the end of the whole process.

Does not handle multiple (bystander) standby databases!
Rolling Upgrade with DG: physru.sh – First Execution

```
physru.sh sys db112_site1 db112_site2 db112_site1 db112_site2 12.2.0.1.0

Please enter the sysdba password:

### Initialize script to either start over or resume execution
...
### Stage 1: Backup user environment in case rolling upgrade is aborted
...
### Stage 2: Create transient logical standby from existing physical standby
...

NOTE: **Database DB112_SITE2 is now ready to be upgraded.** This script has left the
database open in case you want to perform any further tasks before
upgrading the database. Once the upgrade is complete, the database must
opened in READ WRITE mode before this script can be called to resume the
rolling upgrade.
```
physru.sh sys db112_site1 db112_site2 db112_site1 db112_site2 12.2.0.1.0

Please enter the sysdba password:

### Initialize script to either start over or resume execution
...

### Stage 3: Validate upgraded transient logical standby
...

### Stage 4: Switch the transient logical standby to be the new primary
...

### Stage 5: Flashback former primary to pre-upgrade restore point and convert to physical

NOTE: Database DB112_SITE1 has been shutdown, and is now ready to be started using the newer version Oracle binary. This script requires the database to be mounted (on all active instances, if RAC) before calling this script to resume the rolling upgrade.
Rolling Upgrade with DG: physru.sh – Third Execution

```bash
physru.sh sys db112_site1 db112_site2 db112_site1 db112_site2 12.2.0.1.0
Please enter the sysdba password:
### Initialize script to either start over or resume execution
...
### Stage 6: Run media recovery through upgrade redo
...
### Stage 7: Switch back to the original roles prior to the rolling upgrade

NOTE: At this point, you have the option to perform a switchover which will restore DB112_SITE1 back to a primary database and DB112_SITE2 back to a physical standby database. If you answer 'n' to the question below, DB112_SITE1 will remain a physical standby database and DB112_SITE2 will remain a primary database.

Do you want to perform a switchover? (y/n): n
```

Jul 09 12:08:05 2017 [7-1] skipping final switchover
Rolling Upgrade with DG: physru.sh – Third Execution

Execution statistics

```plaintext
### Stage 8: Statistics
script start time: 09-Jul-17 11:12:06
script finish time: 09-Jul-17 12:08:05
total script execution time: +00 00:55:59
wait time for user upgrade: +00 00:44:13
active script execution time: +00 00:11:46
transient logical creation start time: 09-Jul-17 11:12:08
transient logical creation finish time: 09-Jul-17 11:12:34
primary to logical switchover start time: 09-Jul-17 11:59:37
logical to primary switchover finish time: 09-Jul-17 11:59:42
primary services offline for: +00 00:00:05
total time former primary in physical role: +00 00:07:48
time to reach upgrade redo: +00 00:04:36
time to recover upgrade redo: +00 00:04:36

SUCCESS: The physical rolling upgrade is complete
```
Rolling Upgrade with Active Data Guard
Rolling Upgrade with ADG – Overview

- Starting with 12.1, a rolling upgrade can be performed with DBMS_ROLLING PL/SQL.
  - Requires an Active Data Guard license!
- Automates some rolling upgrade steps, leaving for you:
  - Starting database instances/listeners from new home.
  - Environment changes (oratab, password file, …).
  - Database upgrade, database services.
  - `COMPATIBLE` parameter change at the end(*) of the whole process.

Supports multiple (bystander) standby databases!
During upgrade to 12.2.0.1, Broker configuration still needs to be disabled!

Default parameters set by Broker, prior to rolling upgrade:

- `log_archive_dest_n` not set on standby.
- `log_archive_config`.

```
dg_config=(DB121N_SITE1,db121n_site2)  
dg_config=(db121n_site1)
```

```
SQL> EXEC DBMS_ROLLING.INIT_PLAN(future_primary=>'DB121N_SITE2')
ORA-45403: database DB121N_SITE2 must be specified in DG_CONFIG
```

```
SQL> EXEC DBMS_ROLLING.INIT_PLAN(future_primary=>'db121n_site2')
PL/SQL procedure successfully completed.
SQL> EXEC DBMS_ROLLING.BUILD_PLAN
ORA-45414: could not connect to a remote database
```

But, Oracle Net connections work perfectly!
Rolling Upgrade with ADG – Preparation (2)

- Changes on the primary.

```
SQL> ALTER SYSTEM SET log_archive_config='dg_config=(DB121N_SITE1, DB121N_SITE2)';
```

- Changes on all involved standby databases:
  - Depending on the topology (Trailing Group), the log_archive_dest_n parameter might need different settings on different standbys.

```
SQL> ALTER SYSTEM SET log_archive_config='dg_config=(DB121N_SITE1, DB121N_SITE2)';
SQL> ALTER SYSTEM SET log_archive_dest_2='service="DB121N_SITE1.TRIVADIS.COM"','SYNC AFFIRM delay=0 ...'valid_for=(online_logfile,primary_role)';
```
Rolling Upgrade with ADG – Execution (1)

- Designate the future primary and build the upgrade plan.

  ```sql
  SQL> EXEC DBMS_ROLLING.DESTROY_PLAN
  SQL> EXEC DBMS_ROLLING.INIT_PLAN(future_primary=>'DB121N_SITE2')
  SQL> EXEC DBMS_ROLLING.BUILD_PLAN
  ```

- Review the created plan (DBA_ROLLING_PLAN):
  - Primary and one standby database: 66 upgrade steps.
  - Primary, two standby databases (one protecting the old primary in *Trailing Group*): 93 upgrade steps.
  - Primary, two standby databases (one protecting the new primary in *Leading Group*): 194 upgrade steps.

  ```sql
  SQL> EXEC DBMS_ROLLING.SWITCHOVER
  SQL> EXEC DBMS_ROLLING.FINISH
  ```
Rolling Upgrade with ADG – Execution (2)

- Some parameters might need adaptation.

```sql
SQL> EXEC DBMS_ROLLING.SET_PARAMETER -
    2   (scope=>'DB121NS_SITE2',name=>'MEMBER',value=>'LEADING');
```

- All involved standby database need to be mounted – it’s a pity (ADG license!)

```sql
SQL> EXEC DBMS_ROLLING.BUILD_PLAN
ORA-45438: database is not in mounted mode
SQL> @dba_rolling_events.sql
detected invalid open mode on DB121NSSITE2: expected (MOUNTED) actual (OPEN READ ONLY)
```

- Unless, you exclude the standby database 😞.

```sql
SQL> EXEC DBMS_ROLLING.SET_PARAMETER -
    2   (scope=>'DB121NS_SITE2',name=>'INVOLVEMENT',value=>'NONE')
```
Rolling Upgrade with ADG – Execution (3)

Enable Broker configuration.
- Sometimes it works perfectly.

```
DGMGRL> ENABLE CONFIGURATION;
Enabled.
DGMGRL> SHOW CONFIGURATION;
Configuration - db121n
  Protection Mode: MaxAvailability
Members:
db121n_site2 - Primary database
  db121n_site1 - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS (status updated 12 seconds ago)
```

- Sometimes not.

```
DGMGRL> ENABLE CONFIGURATION;
Enabled.
DGMGRL> SHOW CONFIGURATION;
Configuration - db121n
  Protection Mode: MaxAvailability
Members:  db121n_site2 - Primary database
  db121n_site1 - Physical standby database
  Members Not Receiving Redo:
    db121ns_site2 - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
ROLLING DATABASE MAINTENANCE IN PROGRESS
```

Recreate the configuration
Conclusions
Conclusions (1)

- Rolling upgrade can reduce downtimes/risks caused by planned database upgrades.
  - Transient Logical Standby is a fairly simple method, compared to some other replication technologies.
  - Carefully check all restrictions.
  - Not well suited method, if you need a fallback to the older release after a longer period of time!

- Manual method gives you the full control, e.g.: COMPATIBLE
  - physru.sh: can be changed only at the end of the process (second downtime).
  - DBMS_ROLLING: with some manual changes also possible, but certainly not designed to do so.
Conclusions (2)

- DBMS_ROLLING
  - Great feature with some restrictions.
  - Unfortunately, you need ADG license to use it.
  - Error messages sometimes misleading.

- Upgrade with (transient) logical standby is generally a project, very careful evaluation is necessary.