Cross-Platform Database Migration with Oracle Data Guard

a CERN Case Study

Background

Many data centers nowadays run their production Oracle database systems on commodity hardware, mostly relying on software solutions for high availability and redundancy, software such as RAC and ASM. Such an approach, although very cost effective and more and more popular, may lead to quite frequent hardware changes due to relatively short commodity hardware's lifecycle, especially comparing to some branded enterprise products. Many commodity hardware vendors give non-extensible 3-years warranty only, which, after all, may impose database migrations every 3 years or even more often. Doing that without significantly compromising service availability, becomes a challenge as database systems grow larger and larger.

CERN is the European Organization for Nuclear Research, the **world's largest particle physics centre**. It sits astride the Franco-Swiss border near Geneva.

CERN is a laboratory where scientists unite to study the building blocks of matter and the forces that hold them together.

It was also where the **W**orld **W**eb **W**eb was born in 1989

Recently at CERN we were facing exactly this problem. Few years ago we have chosen Oracle RAC 10g on Linux, on non-branded hardware as our main platform. Initially a small setup, it has been growing very quickly and at present it consists of ~110 CPU nodes and ~110 SAN-attached disk arrays. Out of these bricks we have built 7 production multi-terabyte Real Applications Clusters and several smaller 2-node RACs for tests, development and application validation. As our production databases are being continuously accessed by the physics community from all over the world, 24x7 availability was from the beginning one of the most fundamental requirements.

Environment

- Oracle 10g RAC
- Oracle 10g ASM for Mirror & Stripe
- > 100 Database Servers
- > 100 SAN attached Arrays
- RedHat Linux
- 7 Production Clusters
- Multi-TB databases
- 24x7 Availability

Problem

How to migrate to new servers & disk when commodity h/w is at end-of-life?

Solution

Use Oracle Data Guard

Several months ago we decided to replace some going out of warranty CPU servers and disk arrays with the new hardware. The main goal was to minimize service downtime associated with this operation. After checking available options and understanding our internal constraints it turned out that there were few options open to us. On the one hand simple approaches like stopping the original system and copying over all the data to the new machines were not really acceptable due to the significant downtime they would impose. On the other hand incremental hardware replacement (by dropping nodes and adding new boxes one by one) were simply too complicated and too laborious. What we really needed was a way to duplicate the old database, synchronize the duplicate with the original and then quickly switch over applications to use the new database. This is where we turned to Oracle Data Guard for the solution.

Solution - Oracle Data Guard

Oracle Data Guard is a well known and mature feature of Oracle Database Enterprise Edition. It arrived in Oracle 8i (known then as Standby Server), then improved in Oracle 9i and renamed Data Guard. It can be seamlessly used together with other Oracle high availability solutions such as Real Application Cluster (RAC) or Recovery Manager (RMAN).

Data Guard is widely used as a disaster recovery or more generally speaking as a high availability solution, providing an easy way to minimize unplanned outages.

We realised that Data Guard can also be very useful for minimizing planned downtimes, especially those associated with database migration. We believe the reason why Data Guard is not routinely used for such migrations is that most database setups are still based on Enterprise-class SAN technology. Such technology provides tools and features to automate migration, albeit at a cost.

In the CERN setup the TBs of database data was located on commodity disk. ASM was used for mirroring and stripping across arrays. There was no Enterprise-class SAN. Once this commodity disk became obsolete, the data had to be moved to the new hardware and in a way that minimized downtime. Oracle Data Guard could do this.

A detailed description of the process we followed can be found at the end of this document. In essence the process follows the standard setup of a Standby database, as depicted in Diagram A. First the new server and disk hardware is setup as a Standby. Second, the Data Guard standby is synchronized, effectively mirroring the live database on the new hardware. Lastly, a Switchover to the new hardware can be performed during a suitable maintenance window.

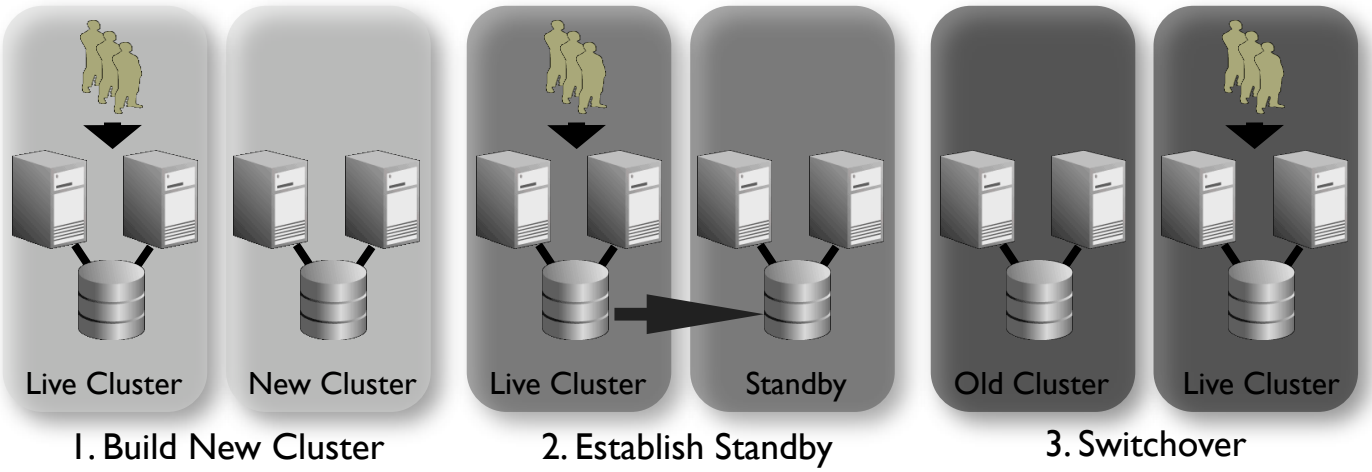

Diagram A: The Switchover Process

Summary

The presented approach not only offers an elegant way of performing Oracle database migrations, more importantly (and what was our initial goal) it enables such migrations with minimum impact on database users. Despite this unquestionable advantage, the approach is still simple and provides for free a safety net of a kind, against unexpected problems.

Finally, although the procedure presented here covers only the simplest possible case, it can be easily modified and extended to cover also some extra operations including: OS upgrade, cluster resize, storage management layer change and probably many others.

During the first half of 2007, at CERN we used the DataGuard-based migration procedure to move about 20 RAC databases and in all the cases it worked very well. The associated database downtime never exceeded 1 hour and did depend mainly on two factors: the amount of redo information being generated on the source system and the time required to move DNS aliases assigned to cluster nodes.

Although this 1 hour of downtime per database was completely acceptable in our case, with some network and Data Guard tuning it could be further limited making the whole procedure even more transparent.

Contacts

CERN
Maria Girone
Jacek Wojcieszuk

CERN Openlab
Dawid Wojcik

Oracle (for Openlab)
Graeme Kerr

Detailed Solution

The procedure presented below shows how Oracle Data Guard Physical Standby can be used to simplify and make more transparent migration of a RAC database from one hardware setup to another.

In the command examples below it is assumed that the database being migrated is named 'orcl' and that it is a 2 node RAC with ASM on Linux. In the scope of the article this database is called primary or the source one, while the database migrated to is referenced to as standby or destination.

1. Configure the New Hardware

First we need to configure hardware and software on the target system. The main steps are:

- Assemble and configure destination database hardware. As we are going to use Physical Standby Database feature it is important that both source and destination hardware follow the same architecture (e.g. x86).

- Install and configure the operating system on all nodes of the new RAC. OS versions on source and destination do not have to strictly match so database migration can be a good opportunity to do an operating system upgrade.

- Install (and patch if necessary) Oracle Clusterware on all the new nodes.

- Install on the new machines Oracle Database Server binaries. Taking into account that RDBMS software versions on the source and the destination systems have to match, we strongly recommend using cloning procedure described in the Oracle documentation.

- Once the Oracle software is in place configure listeners on all new nodes (preferably using `netca`).

- Configure shared storage. The configuration on the destination system does not have to be identical to the one on the source RAC e.g. it is possible to have ASM on the source and OCFS on the target, however as usual homogeneity simplifies the migration procedure. In case you use ASM startup ASM instances on all new nodes and create ASM diskgroups.

```
OLD_DB =
   (DESCRIPTION =
     (ADDRESS = (PROTOCOL = TCP)
                (HOST = oldnode1-vip)
                (PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)
                (HOST = oldnode2-vip)
                (PORT = 1521))
     (LOAD_BALANCE = yes)
     (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = orcl.cern.ch)))
```

Fig 1. New tnsnames.ora entry

- Copy over a tnsnames.ora file from the primary system and edit it replacing all machine names or IP addresses with those of nodes of the new cluster. Add a new entry with 'OLD_DB' (Fig. 1) alias pointing to the source database. Deploy the file on all nodes of the new cluster.

- Create password files (on all new nodes) with the same SYS password as on the database being

```
# on the first node of the new hardware:
orapwd file=$ORACLE_HOME/dbs/orapworcl1 password=XXXXXXXX
# on the second node of the new hardware:
orapwd file=$ORACLE_HOME/dbs/orapworcl2 password=XXXXXXXX
```

Fig 2. Create Password Files

migrated. (Fig. 2)

- Configure at least one new machine to access backups of the source database.

2. Prepare the Primary Database

Meanwhile the DBA can prepare the primary database with the following steps:

- Force logging to ensure that all changes from the primary database will propagate to the standby one:

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

FOR
---
NO
SQL> alter database force logging;

Database altered.
```

- Take a full backup of your primary database or if you already have one make sure you have at least one incremental backup level 1 taken after enabling force logging.

- Backup controlfile for standby:

```
rman target /
RMAN> backup current controlfile for standby;
```

- On all nodes add in tnsnames.ora an entry pointing to the destination database. Name it 'NEW_DB'.

```
NEW_DB =
      (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = newnode1-vip)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = newnode2-vip)(PORT = 1521))
      (LOAD_BALANCE = yes)
      (CONNECT_DATA =
       (SERVER = DEDICATED)
       (SERVICE_NAME = orcl.cern.ch)
       )
      )
```

- Backup the SPFILE as a PFILE and copy it to one of the new nodes:

```
SQL> create pfile='/tmp/initDB.ora' from spfile;

File created.

scp /tmp/initDB.ora newnode1:/tmp
```

3. Prepare the Standby Database

Again on the destination system:

- Modify the PFILE copied from the source database:

```
# Add Data Guard related parameters:
*.log_archive_dest_2='service=OLD_DB valid_for=(online_logfiles,primary_role)'
*.standby_file_management=auto
*.fal_server='OLD_DB'
*.fal_client='NEW_DB'
*.service_names='orcl.cern.ch'

# If there is any difference in the shared storage configuration between
# the source and the destination use conversion parameters
# *.db_file_name_convert=...
# *.log_file_name_convert=...

# Remove the 'control_files' parameter
# Remove all automatic memory allocation parameters
# (the ones with names preceded by double underscore)
# Modify other parameters if needed:
# *.db_recovery_file_dest=...
# *.db_recovery_file_dest_size=...
# *.db_create_file_dest=...
# *.background_core_dump=...
# ...
```

- On all nodes create dump directories if they do not exist.

- Deploy the PFILE as SPFILE on the shared storage and make sure that default local init files point to it:

```
SQL> create spfile='+data_diskgroup/orcl/spfileorcl.ora' from      pfile='/
tmp/initDB.ora';
echo "spfile='+data_diskgroup/orcl/spfileorcl.ora'" > $ORACLE_HOME/dbs/
initorcl1.ora
# and on the other node:
echo "spfile='+data_diskgroup/orcl/spfileorcl.ora'" > $ORACLE_HOME/dbs/
initorcl2.ora
```

- Startup one Oracle instance. Make sure that ORACLE_HOME and ORACLE_SID environment variables are set properly.

```
SQL> startup nomount
```

- Duplicate the source database using RMAN:

```
rman target SYS@OLD_DB auxiliary / nocatalog
RMAN> DUPLICATE TARGET DATABASE FOR STANDBY;
```

- Connect to the running standby database instance and start redo apply:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

- On one of the nodes of the destination cluster, register the database and database instances:

```
srvctl add database -d orcl -o $ORACLE_HOME
srvctl add instance -d orcl -i orcl1 -n newnode1
srvctl modify instance -d orcl -i orcl1 -s +ASM1
srvctl add instance -d orcl -i orcl2 -n newnode2
srvctl modify instance -d orcl -i orcl2 -s +ASM2
```

- If you use custom Oracle services create them on the destination database as copies from the one being migrated. E.g:

```
srvctl add service -d orcl -s orcl_loadbalanced -r orcl1,orcl2 -P BASIC
srvctl add service -d orcl -s orcl_noloadbalanced -r orcl1 -a orcl2 -P BASIC
```

- On all nodes of the standby cluster put appropriate DB instance related entries in the /etc/oratab file

4. ***Start Standby Synchronization.***

- Up to this point both source and destination databases are running independently. In order to start synchronization we have to modify few initialization parameters on the source system:

```
SQL> alter system set log_archive_dest_2='service=NEW_DB
valid_for=(online_logfiles,primary_role)' scope=both sid='*';
SQL> alter system set standby_file_management=auto scope=both sid='*';
```

- To check whether redo stream flows correctly, list shipped archivelogs on the destination:

```
SQL> select sequence#, first_time, next_time, applied from v$archived_log
order by 1;
```

- Archive current redo logs on the primary system:

```
SQL> alter system archive log current;
```

- And repeat the previous query on the destination. The number of rows returned by the query should increase. At steady state all archivelogs should be applied, except for the most recent N-1 logs (where N=number of enabled redo threads).

5. Switchover to New Hardware.

In the current state the production system can run till the next maintenance window. During the scheduled intervention a DBA can proceed with Data Guard switchover and complete the migration. The steps are following:

- Shutdown all services on the primary cluster

```
srvctl stop service -d orcl
```

- Shutdown all the primary database instances but one

```
srvctl stop instance –d orcl –i orcl2
```

- Connect to the only running instance of the primary cluster and switch the primary database to the standby role:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY WITH SESSION
SHUTDOWN;
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT
```

- On the standby system, switch the destination database to the primary role (all standby instances but one have to be down).

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
SQL> ALTER DATABASE OPEN;
```

- Again on the destination system, start other instances of the new database and services:

```
srvctl start instance –d orcl –i orcl2
srvctl start service –d orcl
```

6. Reconfigure Network and Decommission old Hardware.

At this point the migration is finished and the applications can start accessing the database. The way the redirection of applications to the new system can be done strongly depends on the Oracle Net names services configuration. In our environment we always assign DNS aliases to all machines and we expose those aliases to the users. In such a case switching applications to the new system is as simple as moving few DNS aliases.

The old database is still the member of our Data Guard configuration which means we can always start redo apply there and switch back to it in case of some unexpected problems. This also means that before we can dismiss it we will need to reset all the Data Guard related parameters, namely:

```
fal_server, fal_client, log_archive_dest_2, standby_file_management
```

Optionally we may also want to disable forced logging on the migrated database:

```
SQL> alter database no force logging;
```