DataGuard in Practice

Matthias Mann
Unicredit Business Integrated Solutions S.C.p.A.
80538 Munich, Germany

Keywords: Active DataGuard Reader Farm, Snapshot Standby, Protection Modes, Reporting,

Summary

After a short introduction into the DataGuard architecture we will present scenarios in daily business operations which employ the functionalities of DataGuard. This presentation will not be concerned with the features of DataGuard to ensure disaster recovery.
We describe

- the setup of an Active DataGuard Reader Farm consisting of two read only open standby databases as part of a core banking application
- the usage of the Snapshot Standby feature for creating consistent database exports as well as the setup of a "frozen" reporting database

We also present methods used for

- building up the standby databases
- monitoring and controlling the healthiness of the standby operation as well as
- retrieving performance data out of the standby databases.

DataGuard Architecture

DataGuard is the name used by Oracle to include the compound of a primary database and one or more standby databases. The standby databases may be physical or logical in nature. We deal in this presentation only with physical standby databases.

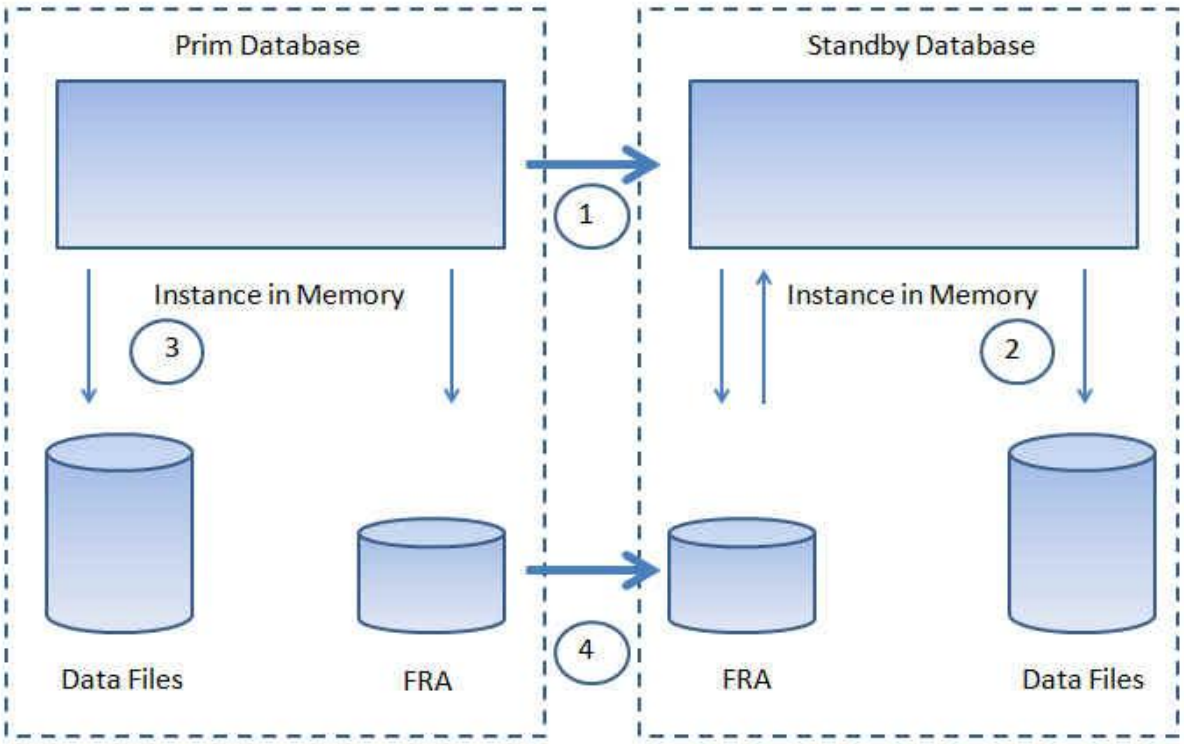

Illustration 1: High Level DataGuard architecture
The main functionalities are: 1 – redo transmission,
2 – redo validation and apply,
3 – DBWR updates primary datafiles
4 – automatic gap resolution

The physical standby database is a block by block identical copy of the primary database. The redo records are transferred by the interaction of the Log Network Server process (LNS) on the primary side and Remote File Server process (RFS) on the standby side.

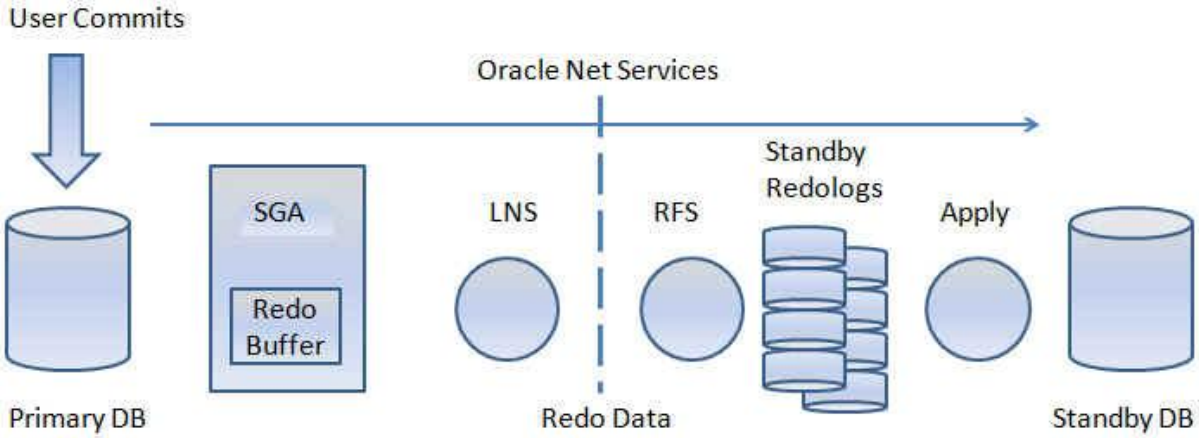

Illustration 2: Architecture of redo transport

Two redo transport methods are supported: synchronous and asynchronous.

Active DataGuard Reader Farm

Active DataGuard is a feature available starting from Oracle 11gR1 onwards. It allows for an opening of the standby database in read-only mode while still applying redo records. The Active DataGuard feature has to be licensed separately. The use case in our installation is part of a core banking system. It provides two copies of the primary database which will be used by satellite applications independently from the main core banking application.

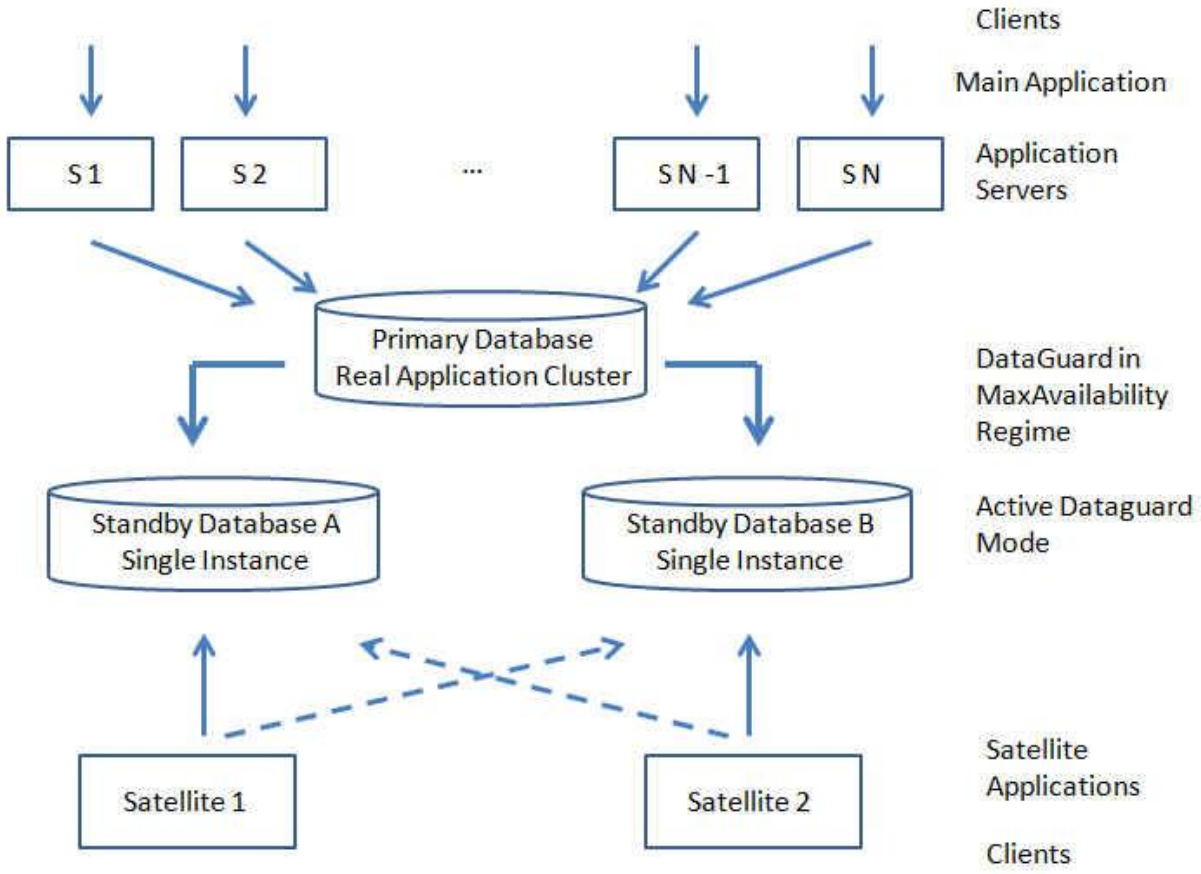

Illustration 3: use case of an Active DataGuard reader farm

The high availability and load balancing is provided by the TNS layer.

```
satellite_1 = (DESCRIPTION=(LOAD_BALANCE=OFF)(FAIL_OVER=ON)
               (ADDRESS=(PROTOCOL=TCP)(HOST=INST_A)(PORT=1521))
               (ADDRESS=(PROTOCOL=TCP)(HOST=INST_B)(PORT=1521))
               (CONNECT_DATA=(SERVICE_NAME=SATELLITE_1)))
```

and

```
satellite_2 = (DESCRIPTION=(LOAD_BALANCE=OFF)(FAIL_OVER=ON)
               (ADDRESS=(PROTOCOL=TCP)(HOST=INST_B)(PORT=1521))
               (ADDRESS=(PROTOCOL=TCP)(HOST=INST_A)(PORT=1521))
               (CONNECT_DATA=(SERVICE_NAME=SATELLITE_2)))
```

With such a setup the client has current data available and will not impact the performance of the primary database. Care has to be taken to guarantee data consistency from an application point of view. This was realized by ensuring that each atellite application connects only to one copy of the database. We run the reader farm in Maximum Availability protection mode to provide for the best possible data synchronicity without bringing the main application at risk.

Snapshot Standby Database

A very elegant feature which came up with Oracle 11 is the so called Snapshot Standby Database. The standby database is opened in read write mode while at the same time the redo records are transferred to the standby destination but are not applied. During the conversion to snapshot standby mode the database implicitly enables flashback on the standby side and starts a temporary life from a guaranteed restore point. During the back conversion all changes are thrown away and the database is flashed back until the restore point. Immediately after the standby database applies the already transferred redo records until it is in sync with the primary database again. We use this feature for two main purposes:

- creation of datapump exports of the production database for refreshing the test and quality assurance setups with up-to-date production data and
- providing frozen copies of the data for reporting purposes.

Administrative Utilities

For a smooth operation of the above described scenarios some scripts were developed for

- easy (re)build of the standby database in case it is needed
- measuring the synchronicity of the Active DataGuard standby database
- providing performance data of the standby instances via statspack snapshots
- control of the snapshot / physical conversion of the standby database

Easy build of the Standby database

In the case that the standby database becomes unusable the database has to be rebuilt. Starting from Oracle 11gR1 onwards it is possible to do this directly with the primary database as source. The process consists of two main steps:

1. Restore using the target database

```
connect target sys/***@source;
connect auxiliary sys/***@destination;
run {
  allocate channel prmy1 type disk;
  allocate auxiliary channel stdby1 type disk;
  duplicate target database for standby from active database
    spfile
      parameter_value_convert 'pattern','pattern',...
      set db_unique_name='...'
```

```
set instance_name='...'
set cluster_database='false'
set log_archive_dest_1='location=USE_DB_RECOVERY_FILE_DEST'
set db_file_name_convert='pattern','pattern'
set log_file_name_convert='pattern','pattern'
set control_files='.../control01.ctl','.../control02.ctl'
set standby_file_management = 'AUTO'
set db_create_file_dest='...'
set db_create_online_log_dest_1='...'
set db_create_online_log_dest_2='...'
set db_recovery_file_dest='...'
set db_recovery_file_dest_size='...'
set dg_broker_config_file1='...'
set dg_broker_config_file2='...'
...
}
```

2. Inclusion in the broker configuration and start of managed recovery

```
SQL> alter database mount standby database;
DGMGRL> add database destination as connect identifier is
'destination' maintained as physical;
DGMGRL>enable database destination;
```

Measuring Standby Synchronicity

For the smooth operation of the reader farm it is essential to know the synchronicity status of the standby databases. We insert on the primary side records in a table and measure if the record arrives within a certain time limit at the standby side.

```
BEGIN
  -- insert marker record
  insert into dgmonitor.dgmonitor(user_comment)
         values (to_char(sysdate, 'YYYYMMDDHH24MISS')||':MonTX')
         returning scn into v_scn;
  commit;
  -- wait a specified grace period
  dbms_lock.sleep(:timeout);
  -- find marker record on standby side
  select scn into v_scn from dgmonitor.dgmonitor@destination
             where scn=v_scn;
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
  BEGIN
    v_ts_msg:=to_char(sysdate, v_date_format, v_nls_param);
    -- status recording in alert log
    sys.dbms_system.ksdwrt(2, v_ts_msg);
    sys.dbms_system.ksdwrt(2, v_fixed_msg || v_msg);
    raise_application_error(-20000, v_msg);
  END;
END;
```

/

Control of standby operation modes

The scenario to be implemented has the following requirements:

- convert the standby database latest at **time A** to snapshot mode (freeze)

```
sqlplus -s /nolog << EOF >/dev/null 2>&1
connect / as sysdba
alter database recover managed standby database cancel;
alter database convert to snapshot standby;
alter database open;
exit;
EOF
```

- maintain snapshot mode until **time B** to have a database for reporting and analysis activities by the business departments
- convert back to physical standby mode and synchronize with primary database

```
sqlplus -s /nolog << EOF >/dev/null 2>&1
connect / as sysdba
shutdown abort
startup mount
alter database convert to physical standby;
shutdown abort;
startup mount
alter database recover managed standby database through all
switchover DISCONNECT USING CURRENT LOGFILE ;
exit;
EOF
```

- measure the apply lag constantly and convert again to snapshot mode if either the lag becomes zero or latest at **time A** (regardless what the actual lag is)

```
select sysdate,
        s.value svalue,
        r.name rname,
        d.database_role
 from   v$dataguard_stats s,
        v$database d,
        v$restore_point r
 where  s.name = 'apply lag';
```

Standby Perfstat Snapshots

The snapshots of the standby statspack need to be created in the primary database with the user stdbyperf. This user has private database links to the schema perfstat in each of the standby databases.

```
sqlplus  -s stdbyperf/*** <<EOF >/dev/null 2>&1
exec
statspack_<db_unique_name>_<instance_name>.snap(i_snap_level=>7);
exit;
EOF
```

Contact address:

Dr. Matthias Mann
Unicredit Business Integrated Solutions S.C.p.A.
Am Tucherpark 12
80538 Munich

Phone: +49 – 378 - 20314
Fax: +49 – 378 – 33 - 20314
Email matthias.mann@unicredit.de
Internet: www.unicredit.eu