Oracle Maximum
Availability Architecture

Full Stack Role Transition

Oracle DBFS and Oracle Data Guard

ORACLE WHITE PAPER | AUGUST 2016

Table of Contents

.

Introduction

This Maximum Availability Architecture (MAA) best practices paper describes a full stack disaster recovery (DR) solution using a combination of Oracle Data Guard and the Oracle Database File System (DBFS). This combination provides real time synchronization and recovery of both the database and file system. By leveraging DBFS within the same database protected by Data Guard, you can reap all of the benefits of Data Guard's high availability, data protection, and disaster recovery along with integrated point-in-time or zero data loss full stack disaster recovery. This configuration is unlike other full stack DR solutions that include a decoupled Data Guard and file system replication approach. Zero data loss and full stack synchronization is always achieved for planned maintenance operations.

For a description of Data Guard and its benefits, refer to http://www.oracle.com/technetwork/database/options/active-data-guard/overview/index.html. For a description of DBFS and its benefits, refer to the Database SecureFiles and Large Objects Developers Guide (http://docs.oracle.com/database/121/ADLOB/adlob_fs.htm#ADLOB45943). You can store application files in DBFS and use DBFS as a general purpose cluster file system.

The use of Oracle Real Application Clusters (RAC) is also included in this white paper to present the different ways to mount DBFS on multiple Oracle RAC nodes in the cluster.

The procedures in this paper were tested using Oracle Database 12*c* Release 1, and are also valid for Oracle Database 11g Release 2. This paper provides the configuration best practices to achieve full stack role transitions with Data Guard and DBFS, and it illustrates how transparent the role transition can be once the configuration is in place. Previous knowledge and understanding of Data Guard is expected.

Oracle Database File System

Mounting DBFS is done using `dbfs_client` which is only available on Linux and Solaris (Solaris 11 SRU7 and later).

DBFS supports most file system operations with these exceptions:

» `ioctl`

» file locking

» asynchronous I/O using `libaio`

» `O_DIRECT` file opens

» hard links, pipes

» other special file modes

Because there is currently no file locking support in DBFS across a cluster or distributed file system access, it is important to make sure that the same file is not written to concurrently by processes on different nodes at the same time.

Configuration

The steps below describe how to configure DBFS to work in an integrated and seamless way with Data Guard role transitions. Oracle Grid Infrastructure Cluster Ready Services (CRS) manages the automatic mounting and unmounting of DBFS.

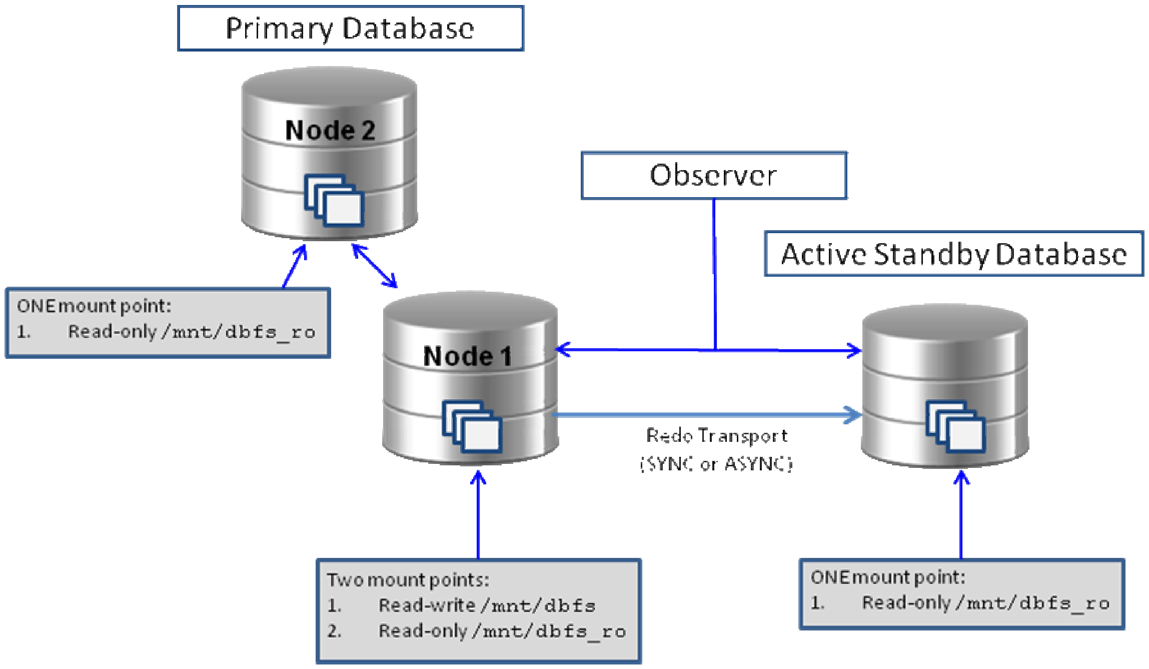

Figure 1. Example DBFS mounts before a Data Guard role transition

If it is not desirable to have read-only access to DBFS on either the Data Guard active standby or any of the Oracle RAC nodes, then the only active mount point is the read-write mount, /mnt/dbfs.

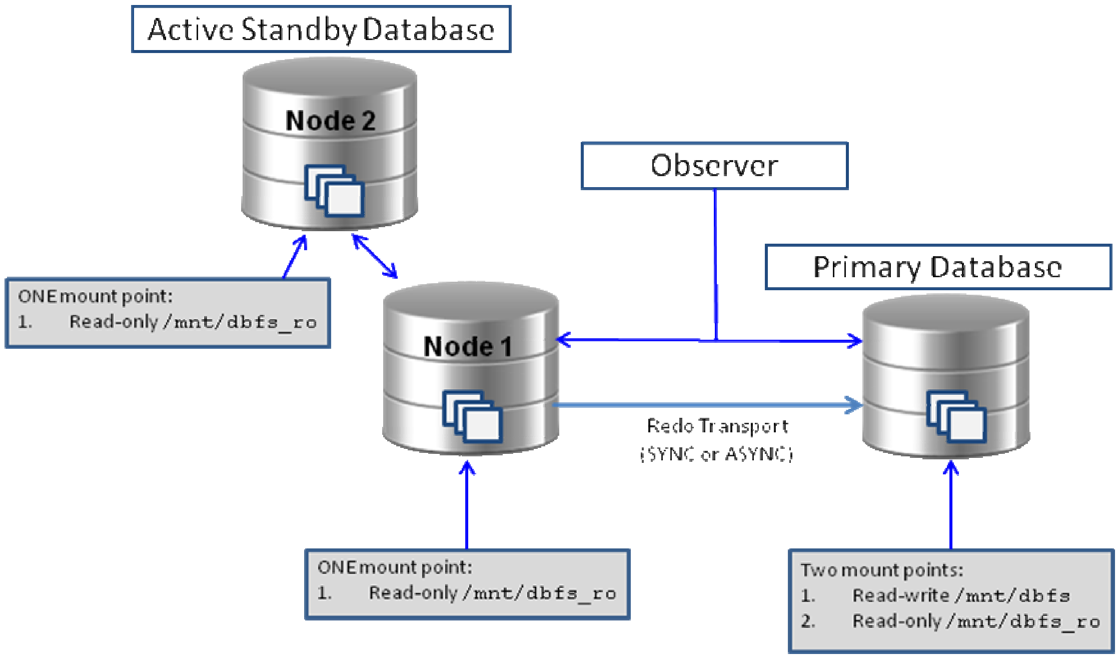

Figure 2. Example DBFS mounts after a Data Guard role transition

1. Create the DBFS File Systems

Follow the instructions provided in the My Oracle Support Document 1054431.1 to configure the required operating system package, the database user, the tablespace, and other prerequisites required for DBFS on Oracle Exadata Database Machine (Linux and Solaris). Configuration of DBFS on a non-Exadata environment is very similar, replacing the use of `dcli` with running the command manually on the Oracle RAC nodes.

If you need additional configuration information when using Linux on a non-Exadata platform, follow the instructions in My Oracle Support Document 869822.1 as a guide to install and configure the FUSE package that is required by DBFS.

Download the DBFS mounting scripts provided with My Oracle Support Document 1054431.1.

Make sure that the TNS alias defined in `tnsnames.ora` used for the DBFS connection to the database specifies the Bequeath (BEQ) protocol. For example:

```
dbfs =
   (DESCRIPTION =
     (ADDRESS =
       (PROTOCOL = BEQ)
       (PROGRAM = /u01/app/oracle/product/12.1.0.2/bin/oracle)
       (ARGV0 = oracleGGDG1)
       (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
       (ENVS = 'ORACLE_HOME=/u01/app/oracle/product/12.1.0.2,ORACLE_SID=GGDG1')
     )
   )
```

The DBFS TNS alias must be defined on all hosts where DBFS will be mounted, ensuring that the `ORACLE_HOME` and `ORACLE_SID` values are set accordingly. This includes all Oracle RAC nodes on the Data Guard primary and standby hosts.

Verify that the file system can be mounted and unmounted from the command line using `dbfs_client`.

For example:

To mount the file system:

```
% dbfs_client dbfs_user@dbfs /mnt/dbfs
% df -k |grep dbfs
```

To test the read-only mounting of DBFS, use the following example:

```
% dbfs_client -o ro dbfs_user@dbfs /mnt/dbfs_ro
% df -k |grep dbfs_ro
```

To unmount the file system:

```
% fusermount -u /mnt/dbfs
% df -k | grep dbfs
```

2. Create a Data Guard Role Based Service to Control Read-Write Access to DBFS

The role based service is required for mounting DBFS in read-write mode on the primary node following a Data Guard role transition. The role based service also dictates which nodes can access DBFS in read-only or read-write modes.

Currently, there is no file locking support in DBFS, so know if your applications will concurrently attempt to update the same files stored in DBFS from different Oracle RAC nodes.

There are three ways to prevent concurrent write access to files on DBFS when using Oracle RAC:

A. Mount DBFS in read-write mode from all Oracle RAC nodes and manage file access within your applications. If possible, make sure applications running on different nodes are writing to different files on each Oracle RAC node.

B. Mount DBFS in read-write mode from only one Oracle RAC node at a time.

C. Mount DBFS in read-write mode on a single node, and read-only mode on the other Oracle RAC nodes.

If you are using an Oracle Active Data Guard standby database, it is possible to configure the role based service so that it will allow read-only access to DBFS on the standby. This configuration is detailed below in Option C.

Option A – Manage File Access Control Through the Application

You can provide read and write access to DBFS on more than one Oracle RAC node in cases where the application is designed to prevent concurrent writes to the same file.

To enable concurrent mounting of DBFS from more than one Oracle RAC node, create the role based service by listing all of the database instance names with the `-preferred` parameter:

```
% srvctl add service -db <db_name> -service dbfs_db -role PRIMARY -preferred
<preferred_instances>
```

Make sure you create this role based service on the Data Guard primary and standby hosts.

For example:

```
% srvctl add service -db GGDG -service dbfs_db -role PRIMARY
-preferred GGDG1, GGDG2
```

NOTE: Rename GGDG1 and GGDG2 with the instance names in your environment.

To start and check the status of the service:

```
% srvctl start service –d GGDG –service dbfs_db
% srvctl status service -d GGDG -service dbfs_db
```

Option B – Mount DBFS From Only One Oracle RAC Node at a Time

Mounting DBFS from only one Oracle RAC node at a time prevents multiple processes on different nodes concurrently writing to the same files at the same time. It does not prevent processes on the same node writing to the same file.

Use the same instructions provided above in Option A to create the role based service.

Make sure you create this role based service on the Data Guard primary and standby hosts.

Option C – Mount DBFS in Read-Write Mode on One Node, and Read-Only on Other Nodes

In order to allow a single Oracle RAC node to mount DBFS in read-write mode, you must follow the instructions provided above with Option A to create the `dbfs_db` role based service.

If the standby database is **not** an Active Data Guard standby database you must create a second role based service that will be used for read-only mounting of DBFS when the standby becomes a primary. If you are using an Active Data Guard standby and require DBFS to be mounted read-only on the standby there is no need to create the second role based service.

Create the following additional role based service for the read only DBFS access:

```
% srvctl add service -db <db_name> -service dbfs_db_ro -role PRIMARY -preferred
<preferred_instances>
```

For example:

```
% srvctl add service -db GGDG -service dbfs_db_ro -role PRIMARY -preferred
GGDG1,GGDG2
```

NOTE: Rename GGDG1 and GGDG2 with the instance names in your environment that require read only access to DBFS.

Create this role based service on the Data Guard primary and standby hosts.

3. Register DBFS as a CRS Resource

DBFS must be registered with CRS so that the file system can be automatically mounted and unmounted when the role based services are started and stopped.

For all of the options below, download the `dbfs-mount.conf` and `mount-dbfs.sh` files from My Oracle Support Document 1054431.1.

There are three ways to prevent concurrent write access to files on DBFS when using Oracle RAC:

A. Mount DBFS in read-write mode from all Oracle RAC nodes and manage file access within your applications. If possible, make sure applications running on different nodes are writing to different files on each Oracle RAC node.

B. Mount DBFS in read-write mode from only one Oracle RAC node at a time.

C. Mount DBFS in read-write mode on a single node, and read-only mode on the other Oracle RAC nodes.

If you are using an Oracle Active Data Guard standby database, it is possible to configure the role based service so that it will allow read-only access to DBFS on the active standby. This configuration is detailed below in Option C.

Option A – Manage File Access Control Through the Application

Change the `dbfs_client` mount options in the `dbfs-mount.conf` file to include the `failover` parameter that disables batched writes and write waits, preventing data loss in the event of an Oracle RAC node failure or a Data Guard failover. You must also change other variables in the `dbfs-mount.conf` file, making sure you read the comments for what values to change.

Example mount options;

```
MOUNT_OPTIONS=allow_other,direct_io,failover
```

Edit the `dbfs-mount.sh` script so that it accesses the correct `dbfs-mount.conf` file:

```
CONFIG=/u01/oracle/scripts/mount-dbfs.conf
```

NOTE: Make sure the `mount-dbfs.sh` and `mount-dbfs.conf` files are copied to all of the Data Guard primary and standby Oracle RAC nodes that mount DBFS, and edit the files with the correct variable values.

It is important to test the `mount-dbfs.sh` script to ensure that DBFS can be mounted and unmounted:

To mount DBFS in read-write mode:

```
% ./mount-dbfs.sh start
```

To check the status of DBFS:

```
% ./mount-dbfs.sh status
% df -k /mnt/dbfs
```

To unmount DBFS:

```
% ./mount-dbfs.sh stop
```

Use the following example script to create the DBFS CRS resource, which will mount DBFS in read-write mode, on all of the Oracle RAC nodes in the cluster:

```
#!/bin/bash
ACTION_SCRIPT=/u01/oracle/scripts/mount-dbfs.sh
RESNAME=dbfs_mount
DEPNAME= ora.ggdg.dbfs_db.svc       #Role based service created in step 2
ORACLE_HOME=/u01/app/12.1.0.2/grid
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME \
  -type local_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup:always($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
```

Test the new DBFS CRS resource by mounting and unmounting the file system using CRS on all of the Oracle RAC nodes that will require a read-write mount to DBFS:

```
% crsctl start resource dbfs_mount
% crsctl status resource dbfs_mount

% df -k /mnt/dbfs           # Check that the file system is mounted
% crsctl stop resource dbfs_mount
```

Don’t forget to create the DBFS CRS resource on the Data Guard standby hosts. The mounting of DBFS on the standby hosts must wait until you carry out a Data Guard role transition.

Start the new CRS resource on the Data Guard primary Oracle RAC nodes, and DBFS will continue to be mounted on the primary hosts following role transitions.

Option B – Mount DBFS From Only One Oracle RAC Node at a Time

Change the `dbfs_client` mount options in the `dbfs-mount.conf` file to include the `failover` parameter that disables batched writes and write waits, preventing data loss in the event of an Oracle RAC node failure or a Data Guard failover. You must also change other variables in the `dbfs-mount.conf` file, making sure you read the comments for what values to change.

Example mount options:

```
MOUNT_OPTIONS=allow_other,direct_io,failover
```

Edit the `dbfs-mount.sh` script so that it accesses the correct `dbfs-mount.conf` file:

```
CONFIG=/u01/oracle/scripts/mount-dbfs.conf
```

NOTE: Make sure that the `mount-dbfs.sh` and `mount-dbfs.conf` files are copied to all of the Data Guard primary and standby Oracle RAC nodes that mount DBFS and edit the files with the correct variable values.

It is important to test the `mount-dbfs.sh` script to ensure that DBFS can be mounted and unmounted:

To mount DBFS in read-write mode:

```
% ./mount-dbfs.sh start
```

To check the status of DBFS:

```
% ./mount-dbfs.sh status
% df -k /mnt/dbfs
```

To unmount DBFS:

```
% ./mount-dbfs.sh stop
```

Use the following example script to create the DBFS CRS resource:

```
#!/bin/bash
ACTION_SCRIPT=/u01/oracle/scripts/mount-dbfs.sh
RESNAME=dbfs_mount
DEPNAME=ora.ggdg.dbfs_db.svc        # Role based service name created in step 2
ORACLE_HOME=/u01/app/12.1.0.2/grid
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME \
  -type cluster_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         HOSTING_MEMBERS='ggdghost01 ggdghost02', \
         PLACEMENT='restricted', \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup:always($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
```

The following `crsctl` parameters are required with DBFS mounted in read-write mode on a single node:

- `type` – ensures that the DBFS resource only runs on a single node at a time.
- `HOSTING_MEMBERS` – the ordered list of Oracle RAC nodes that can host the DBFS resource. Make sure that all of the nodes listed here are already configured with the role based service from step 2.
- `PLACEMENT` – using a value of `restricted` means that the DBFS resource can only run on the host names listed with the `HOSTING_MEMBERS` parameter.

NOTE: If Oracle RAC is not used on either the Data Guard primary or standby hosts, use the same `crsctl add resource` command as specified above in Option A.

Test the new DBFS CRS resource by mounting and unmounting the file system:

```
% crsctl start resource dbfs_mount

% crsctl status resource dbfs_mount
% df -k /mnt/dbfs           # Check that the file system is mounted
```

Relocating the resource automatically moves through the list of Oracle RAC nodes specified with the `HOSTING_MEMBERS` parameter:

```
% crsctl relocate resource dbfs_mount
% crsctl stop resource dbfs_mount
```

Don't forget to create the DBFS CRS resource on the Data Guard standby host. The mounting of DBFS in read-write mode on the standby hosts must wait until you carry out a Data Guard role transition.

Start the new CRS resource on a Data Guard primary Oracle RAC node, and DBFS will continue to be mounted in read-write mode on the primary host following role transitions.

Option C – Mount DBFS in Read-Write mode on One Node, and Read-Only on Other Nodes

To create the CRS resource for mounting DBFS in read-write mode on only one Oracle RAC node, follow the instructions under Option A.

A second mount point is created to allow read-only access to DBFS on all Oracle RAC nodes, and if applicable, the Active Data Guard standby hosts.

Copy the `dbfs-mount.conf` and `mount-dbfs.sh` files and rename them to `dbfs-mount-readonly.conf` and `mount-dbfs-readonly.sh`. Make the following edits to `dbfs-mount-readonly.conf` file:

```
MOUNT_OPTIONS=allow_other,direct_io,ro
MOUNT_POINT=/mnt/dbfs_ro
```

You might also need to make changes to other variables in the `dbfs-mount-readonly.conf` file, making sure you read the comments for what values to change.

Edit the `dbfs-mount-readonly.sh` script so that it accesses the correct `dbfs-mount-readonly.conf` file:

```
CONFIG=/u01/oracle/scripts/mount-dbfs-readonly.conf
```

Copy and edit these files on all of the Oracle RAC nodes and Active Data Guard standby hosts that need read-only access to DBFS.

Be sure to test the `mount-dbfs-readonly.sh` script to ensure that DBFS can be mounted and unmounted in read-only mode on all Oracle RAC nodes on the Data Guard primary and Active Data Guard standby hosts:

To mount DBFS in read-only mode:

```
% ./mount-dbfs-readonly.sh start
```

To check the status of DBFS:

```
% ./mount-dbfs-readonly.sh status
```

To unmount DBFS:

```
% ./mount-dbfs-readonly.sh stop
```

If you are using Active Data Guard, use the following example script to create the DBFS CRS resource when you require all of the Oracle RAC nodes in the cluster to mount DBFS in read-only mode, no matter if it is a primary or active standby database:

```
#!/bin/bash
ACTION_SCRIPT=/u01/oracle/scripts/mount-dbfs-readonly.sh
RESNAME=dbfs_mount_ro
DEPNAME=ora.ggdg.db         # Database service name
ORACLE_HOME=/u01/app/12.1.0.2/grid
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME \
  -type local_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup:always($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
```

NOTE: The `DEPNAME` value should be the database service name, and **not** a role based service name.

If you are not using Active Data Guard, use the following CRS resource creation script example, using the role based service name instead of the database service name:

```
#!/bin/bash
ACTION_SCRIPT=/u01/oracle/scripts/mount-dbfs-readonly.sh
RESNAME=dbfs_mount_ro
DEPNAME=ora.ggdg.dbfs_db.svc       # Role based service name created in step 2
ORACLE_HOME=/u01/app/12.1.0.2/grid
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME \
  -type local_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup:always($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
```

Test the new DBFS CRS resource by mounting and unmounting the file system using CRS on all of the Oracle RAC nodes that require a read-only mount to DBFS:

```
% crsctl start resource dbfs_mount_ro
% crsctl status resource dbfs_mount_ro

% df -k /mnt/dbfs_ro        # Check that the file system is mounted
% crsctl stop resource dbfs_mount_ro
```

Don’t forget to create the DBFS CRS resource on the Data Guard standby host.

Start the new CRS resource on the Data Guard primary and if using Active Data Guard, the active standby Oracle RAC nodes, and DBFS will continue to be mounted in read-only mode when the databases are started, or become a primary database in a non-Active Data Guard configuration.

Full Stack Role Transition

Once the configuration best practices presented in this white paper are implemented, a full stack role transition, including the database and all of the data residing on the DBFS file system, is carried out in the same way as a standard Data Guard role transition. When DBFS is managed by CRS and uses role based services, then a Data Guard switch over automatically unmounts the file system on the old primary database and then remounts the synchronized file system on the new primary database as part of a Data Guard switchover operation.

Data Guard Switchover

For example:

```
DGMGRL> switchover to 'GGDGS'
```

After the role transition is complete, DBFS automatically mounts in read-write mode on the current primary database. If configured, the read-only mounts are also mounted on multiple Oracle RAC nodes and/or the active standby database. Full stack switchover results in zero data loss and is leveraged as a planned maintenance activity.

Data Guard Failover

For example:

```
DGMGRL> failover to 'GGDGS'
```

After the role transition is complete, DBFS automatically mounts in read-write mode on the current primary database. If configured, the read-only mounts are also mounted on multiple Oracle RAC nodes and/or the active standby database. The database and DBFS file system fail over to the same point in time because the redo contains both database and DBFS file system changes.

If Fast Start Failover (FSFO) is configured, after an automatic failover, DBFS is automatically mounted in read-write mode on the current primary database.

Reinstating the Data Guard Standby Database

After a Data Guard failover, when reinstating the old primary database as a standby, there are no special steps required for DBFS. If an active standby database is configured with a read-only mount to DBFS, after the standby database is reinstated, DBFS automatically mounts in read-only mode and is synchronized with the rest of the database.

For example:

```
DGMGRL> reinstate database 'GGDGP'
```

Conclusion

After implementing the above configuration with DBFS and Oracle Data Guard, executing a full stack role transition is as easy as executing your standard Data Guard role transition commands. No additional steps are required to unmount or mount the DBFS file system after a role transition, whether it is a switchover or a failover.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

Integrated Cloud Applications & Platform Services

Full Stack Role Transition – Oracle DBFS and Oracle Data Guard
August 2016
Author: Stephan Haisley