Oracle Maximum
Availability Architecture

フル・スタック・ロール移行

Oracle DBFSとOracle Data Guard

Oracleホワイト・ペーパー | 2016年8月

目次

概要

Oracle Maximum Availability Architecture（MAA）ベスト・プラクティスに関するこのホワイト・ペーパーでは、Oracle Data Guard と Oracle Database File System（DBFS）を組み合わせて使用した、フル・スタック・ディザスタ・リカバリ（DR）ソリューションについて説明します。この組み合わせにより、データベースとファイル・システム双方の、リアルタイムな同期とリカバリが実現できます。DBFS を、Data Guard で保護された同じデータベース内で利用し、Data Guard の高可用性、データ保護、ディザスタ・リカバリのすべてのメリットを、統合されたポイント・イン・タイムまたはデータ損失ゼロの、フル・スタック・ディザスタ・リカバリと併せて活用できます。この構成は、Data Guard とファイル・システムが分離したレプリケーション・アプローチを含む、他のフル・スタック DR ソリューションとは異なります。計画メンテナンスの操作時に、データ損失ゼロとフル・スタック同期が常に達成されます。

Data Guard とそのメリットについては、

http://www.oracle.com/technetwork/jp/database/options/active-data-guard/overview/index.html を参照してください。DBFS とそのメリットについては、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』（https://docs.oracle.com/cd/E57425_01/121/ADLOB/GUID-B7A83817-F0D6-4A09-AE98-DFC966783109.htm）を参照してください。アプリケーション・ファイルを DBFS に保存して、DBFS を汎用クラスタ・ファイル・システムとして使用することができます。

このホワイト・ペーパーでは、Oracle Real Application Clusters（RAC）の使用についても取り上げ、DBFS をクラスタ内の複数の Oracle RAC ノードにマウントするさまざまな方法を紹介します。

このホワイト・ペーパーの手順は、Oracle Database 12c Release 1 を使ってテストされましたが、Oracle Database 11g Release 2 にも有効です。また、Data Guard と DBFS でフル・スタック・ロール移行を達成する構成のベスト・プラクティスについて説明します。適切に構成すれば、ロール移行が透過的に行われることを示します。読者に Data Guard に関する知識があり、理解していることが前提です。

Oracle Database File System

DBFSのマウントは、LinuxとSolaris（Solaris 11 SRU7以降）のみで利用できる`dbfs_client`を使って実行します。

DBFSは以下の項目を前提に、ほとんどのファイル・システム操作をサポートします。

» `ioctl`

» ファイル・ロック

» `libaio`を使用した非同期I/O

» `O_DIRECT`ファイルを開く

» ハード・リンク、パイプ

» 他の特別ファイル・モード

DBFSでは現在、クラスタまたは分散ファイル・システムに対するファイル・ロックがサポートされていないため、複数のノード上で同じファイルがプロセスによって同時並行的に書き込まれないようにすることが重要です。

構成

以下の手順は、Data Guardロール移行と統合的かつシームレスに連携するように、DBFSを構成する方法を示します。Oracle Grid InfrastructureのCluster Ready Services（CRS）は、DBFSの自動的なマウントとアンマウントを管理します。

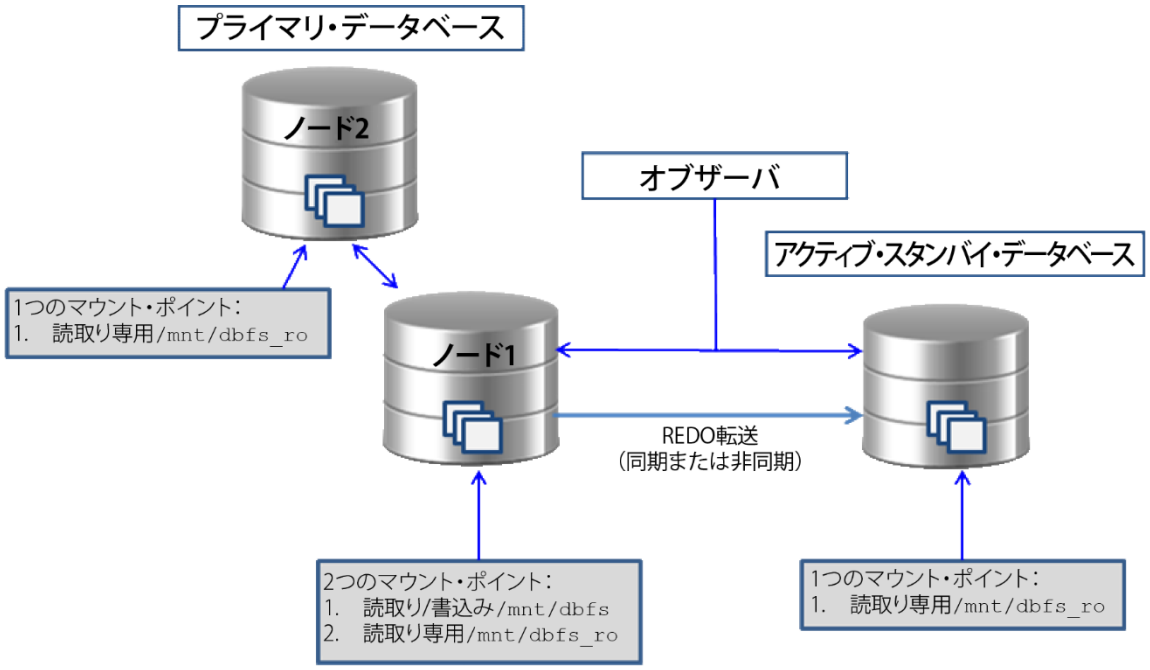

図1：Data Guardロール移行前にDBFSをマウントする例

Data Guardアクティブ・スタンバイまたはOracle RACノードのいずれかのDBFSに対し、読取り専用アクセス権を持つことが望ましくない場合、唯一のアクティブなマウント・ポイントは読取り/書込みマウント、/mnt/dbfsになります。

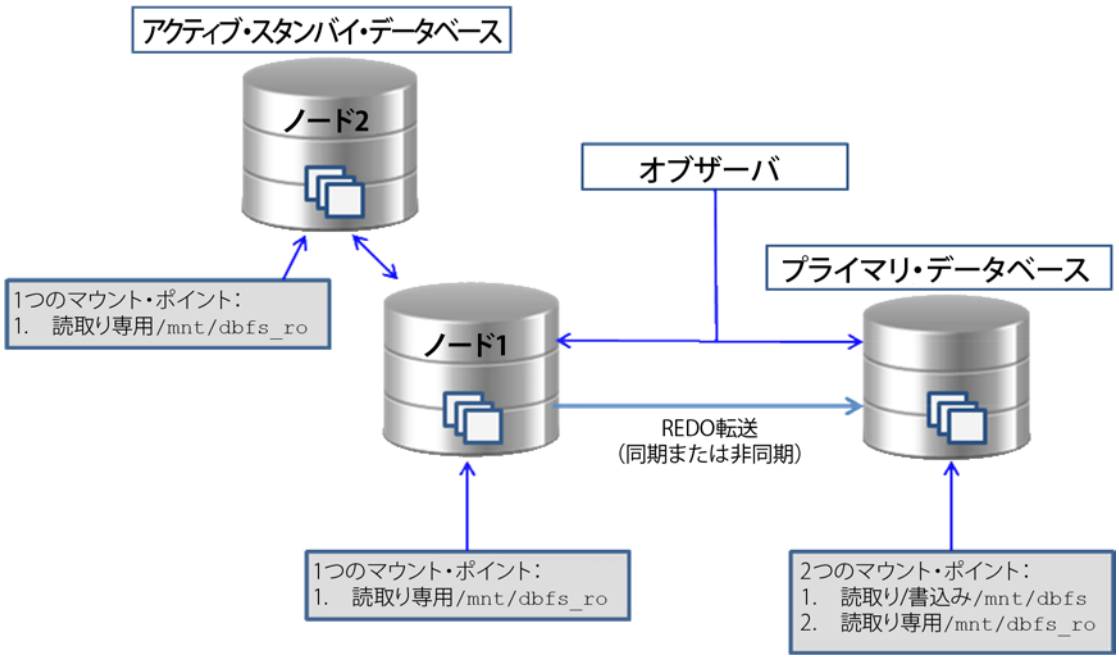

図2：Data Guardロール移行後にDBFSをマウントする例

1. DBFSファイル・システムを作成する

My Oracle Supportのドキュメント1054431.1に記載の手順に従って、必要なオペレーティング・システム・パッケージ、データベース・ユーザー、表領域、Oracle Exadata Database Machine（LinuxとSolaris）上のDBFSに必要な他の前提条件を構成します。非Oracle Exadata上のDBFSの構成も非常に近く、`dcli`を使用する代わりに、Oracle RACノード上で同じコマンドを手動にて実行します。

非Oracle Exadataプラットフォーム上でLinuxを使用していて、その他の構成情報が必要な場合は、My Oracle Supportのドキュメント869822.1の手順に従い、DBFSに必要なFUSEパッケージをインストール、構成します。

My Oracle Supportのドキュメント1054431.1で提供されているDBFSマウント・スクリプトをダウンロードします。

データベースへのDBFS接続に使われる、tnsnames.oraで定義されたTNSエイリアスで、必ずBequeath（BEQ）プロトコルを指定します。次に例を示します。

```
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')
  )
 )
```

DBFS TNSエイリアスは、DBFSがマウントされるすべてのホストで定義し、ORACLE_HOMEとORACLE_SIDの値をその定義に応じて設定する必要があります。これには、Data Guardのプライマリ・ホストとスタンバイ・ホスト上のすべてのOracle RACノードが含まれます。

ファイル・システムがマウントおよびアンマウントできることを、dbfs_clientを使ってコマンドラインから確認します。次に例を示します。

ファイル・システムをマウントする場合

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

DBFSの読取り専用のマウントをテストするには、次の例を使用します。

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

ファイル・システムをアンマウントする場合

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

2. DBFSへの読取り/書込みアクセス制御をするためのData Guardロールベース・サービスを作成する

Data Guardロール移行後、DBFSを読取り/書込みモードでプライマリ・ノードにマウントするには、ロールベース・サービスが必要です。ロールベース・サービスは、どのノードが読取り専用モードまたは読取り/書込みモードでDBFSにアクセスできるかを指示します。

現在、DBFSではファイル・ロックがサポートされていません。そのため、アプリケーションが複数のOracle RACノードから、DBFS上の同じファイルを同時に更新するか否かの確認をしてください。

Oracle RACを使用する場合、DBFSへの同時書込みアクセスを回避する方法は3つあります。

A. すべてのOracle RACノードからDBFSを読取り/書込みモードでマウントし、ファイル・アクセスをアプリケーション内で管理する。可能な場合は、複数のノードで実行中のアプリケーションが各Oracle RACノード上の異なるファイルに書込みを実行するようにする。

B. 一度にOracle RACの1ノードのみからDBFSを読取り/書込みモードでマウントする。

C. DBFSに対しOracle RACの1ノードは読取り/書込みモードでマウントし、他のノードは読取り専用モードでマウントする。

Oracle Active Data Guardスタンバイ・データベースを使用している場合は、スタンバイ上のDBFSへの読取り専用アクセスを許可するように、ロールベース・サービスを構成できます。この構成については、以下のオプションCで詳述します。

オプションA – アプリケーションを介したファイル・アクセス制御の管理

アプリケーションにおいて、同じファイルへの同時書込みを回避するように設計されている場合は、複数のOracle RACノード上のDBFSに対する読取りおよび書込みアクセスを許可できます。

複数のOracle RACノードからDBFSを同時にマウントできるようにするためには、すべてのデータベース・インスタンスの名前を、-preferredパラメータで一覧表示したロールベース・サービスを作成します。

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

このロールベース・サービスを、必ずData Guardプライマリ・ホストとスタンバイ・ホスト上に作成してください。

次に例を示します。

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

注：GGDG1とGGDG2を環境内のインスタンス名に置き換えます。

サービスを開始してステータスを確認する場合

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

オプションB - 一度にOracle RACの1ノードのみからDBFSマウント

一度に1つのみのOracle RACノードからDBFSをマウントすると、複数ノード上の複数プロセスが、同時に同じファイルに書き込むことを防ぐことができます。同じノード上のプロセスによる同じファイルへの書込みは妨げません。

上記のオプションAと同じ手順を使って、ロールベース・サービスを作成します。

このロールベース・サービスを、必ずData Guardプライマリ・ホストとスタンバイ・ホスト上に作成してください。

オプションC - 1ノードで読取り/書込みモードのDBFSマウントおよび他のノードで読取り専用モードのDBFSマウント

1つのOracle RACノードが読取り/書込みモードでDBFSマウントをできるようにするには、上記のオプションAの手順に従って、`dbfs_db`ロールベース・サービスを作成する必要があります。

スタンバイ・データベースが、Active Data Guardスタンバイ・データベースではない場合、スタンバイがプライマリになるときにDBFSの読取り専用マウントに使用される、2番目のロールベース・サービスを作成する必要があります。Active Data Guardスタンバイを使用していて、DBFSをスタンバイに読取り専用でマウントする必要がある場合は、2番目のロールベース・サービスを作成する必要はありません。

読取り専用DBFSアクセスに用いる追加のロールベース・サービスを作成します。

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

次に例を示します。

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

注：GGDG1とGGDG2を、DBFSへの読取りアクセスを要求する環境内のインスタンスの名前に置き換えます。

このロールベース・サービスをData Guardのプライマリ・ホストとスタンバイ・ホスト上に作成してください。

3. DBFSをCRSリソースとして登録する

ロールベース・サービスが開始および停止したときに、ファイル・システムを自動的にマウント/アンマウントできるよう、DBFSをCRSに登録する必要があります。

以下のすべてのオプションで、My Oracle Supportのドキュメント1054431.1からdbf -mount.confファイルとmount-dbfs.shファイルをダウンロードします。

Oracle RACを使用する場合、DBFSへの同時書込みアクセスを回避する方法は3つあります。

A. すべてのOracle RACノードからDBFSを読取り/書込みモードでマウントし、ファイル・アクセスをアプリケーション内で管理する。可能な場合は、複数のノードで実行中のアプリケーションが各Oracle RACノード上の異なるファイルに書込みを実行するようにする。

B. 一度にOracle RACの1ノードのみからDBFSを読取り/書込みモードでマウントする。

C. DBFSに対しOracle RACの1ノードは読取り/書込みモードでマウントし、他のノードは読取り専用モードでマウントする。

Oracle Active Data Guardスタンバイ・データベースを使用している場合は、アクティブ・スタンバイ上のDBFSへの読取り専用アクセスを許可するようにロールベース・サービスを構成できます。この構成については、以下のオプションCで詳述します。

オプションA – アプリケーションを介したファイル・アクセス制御の管理

`dbfs-mount.conf`ファイルの`dbfs_client`マウント・オプションであるfailoverパラメータの指定により、バッチ書込みと書込み待機を無効にし、Oracle RACノードの障害またはData Guardのフェイルオーバー時にデータ損失を防ぎます。dbfs-mount.confファイルにおける他の変数も、値に関するコメントを確認の上、変更してください。

マウント・オプションの例を示します。

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

適切な`dbfs-mount.conf`ファイルにアクセスするよう、`dbfs-mount.sh`スクリプトを編集します。

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

注：DBFSをマウントする、すべてのData Guardプライマリとスタンバイの Oracle RACノードに、`mount-dbfs.sh` ファイルと`mount-dbfs.conf`ファイルをコピーして、ファイルを適切な変数値で編集します。

`mount-dbfs.sh`スクリプトをテストして、DBFSをマウントまたはアンマウントできることの確認が重要です。

DBFSを読取り/書込みモードでマウントする場合

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

DBFSのステータスをチェックする場合

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

DBFSをアンマウントする場合

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

以下のスクリプト例を使用して、DBFSを読取り/書込みモードでクラスタ内のすべてのOracle RACノードにマウントする、DBFS CRSリソースを作成します。

```
#!/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"
```

新しいDBFS CRSリソースをテストします。それには、DBFSへの読取り/書込みマウントが必要なすべてのOracle RACノードで、CRSを使ったファイル・システムのマウント/アンマウントをします。

```
% 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
```

Data Guardのスタンバイ・ホスト上にDBFS CRSリソースを必ず作成してください。Data Guardロール移行を実行するまで、スタンバイ・ホストへのDBFSのマウントを待機する必要があります。

Data GuardプライマリのOracle RACノード上で新しいCRSリソースを開始すると、DBFSはロール移行に続いて、プライマリ・ホストにマウントされます。

オプションB - 一度にOracle RACの1ノードのみからのDBFSマウント

`dbfs-mount.conf`ファイルの`dbfs_client`マウント・オプションであるfailoverパラメータの指定により、バッチ書込みと書込み待機を無効にし、Oracle RACノードの障害またはData Guardのフェイルオーバー時にデータ損失を防ぎます。`dbfs-mount.conf`ファイルにおける他の変数も、値に関するコメントを確認の上、変更してください。

マウント・オプションの例を示します。

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

適切なdbfs-mount.confファイルにアクセスするよう、dbfs-mount.shスクリプトを編集します。

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

注：DBFSをマウントするすべてのData Guardプライマリとスタンバイの Oracle RACノードに、`mount-dbfs.sh`ファイルと`mount-dbfs.conf`ファイルをコピーして、ファイルを適切な変数値で編集します。

mount-dbfs.shスクリプトをテストして、DBFSをマウントまたはアンマウントできることの確認が重要です。DBFSを読取り/書込みモードでマウントする場合

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

DBFSのステータスをチェックする場合

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

DBFSをアンマウントする場合

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

以下のスクリプト例を使用して、DBFS CRSリソースを作成します。

```
#!/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"
```

1つのノードでDBFSを読取り/書込みモードをマウントする場合、以下の`crsctl`パラメータが必要です。

» `type` – DBFSリソースは一度に1つのノードのみで実行します。

» `HOSTING_MEMBERS` – DBFSリソースをホストできるOracle RACノードを順番に並べたリスト。ここに一覧にされているすべてのノードがすでに手順2のロールベース・サービスで構成されていることを確認します。

» `PLACEMENT` – `restricted`の値を使用すると、DBFSリソースは、`HOSTING_MEMBERS`パラメータで一覧表示された名前のホスト上でのみ実行できます。

注：Oracle RACがData Guardのプライマリ・ホストまたはスタンバイ・ホストのいずれでも使用されていない場合は、上記のオプションAで指定したものと同じ`crsctl add resource`コマンドを使用します。

ファイル・システムをマウント/アンマウントして、新しいDBFS CRSリソースをテストします。

```
% crsctl start resource dbfs_mount

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

リソースを再配置すると、HOSTING_MEMBERSパラメータで指定されたOracle RACノードの一覧内で、そのリソースの位置が自動的に変わります。

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

Data Guardのスタンバイ・ホスト上に、DBFS CRSリソースを必ず作成してください。Data Guardロール移行を実行するまで、読取り/書込みノードでのスタンバイ・ホストへのDBFSのマウントを待機する必要があります。

Data GuardプライマリのOracle RACノード上で新しいCRSリソースを開始すると、DBFSはロール移行に続いて、読取り/書込みモードでプライマリ・ホストにマウントされます。

オプションC - 1ノードで読取り/書込みモードのDBFSマウントおよび他のノードで読取り専用モードのDBFSマウント

1つのみのOracle RACノードにDBFSを読取り/書込みモードでマウントするためのCRSリソースを作成するには、オプションAの手順に従います。

すべてのOracle RACノード、および該当する場合はActive Data Guardのスタンバイ・ホスト上のDBFSへの読み取り専用アクセスを可能にする2番目のマウント・ポイントが、作成されます。

dbfs-mount.confファイルとmount-dbfs.shファイルをコピーして、dbfs-mount-readonly.confおよびmount-dbfs-readonly.shに名前を変更します。dbfs-mount-readonly.confファイルに以下の編集を行います。

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

また、dbfs-mount-readonly.confファイルの他の変数も、値に関するコメントを確認の上、必要に応じて変更してください。

適切なdbfs-mount-readonly.confファイルにアクセスするようにdbfs-mount-readonly.shスクリプトを編集します。

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

すべてのOracle RACノード、およびDBFSへの読取り専用アクセスが必要なActive Data Guardのスタンバイ・ホスト上のこれらのファイルを、コピーして編集します。

mount-dbfs-readonly.shスクリプトをテストして、Data Guardのプライマリおよび Active Data Guardのスタンバイ・ホスト上のすべてのOracle RACノード上で、読取り専用モードのDBFSマウント/アンマウントができることを確認します。

DBFSを読取り専用モードでマウントする場合

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

DBFSのステータスをチェックする場合

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

DBFSをアンマウントする場合

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

Active Data Guardを使用しており、プライマリまたはアクティブ・スタンバイ・データベースかどうかに関係なく、クラスタ内のすべてのOracle RACノードにDBFSを読取り専用モードでマウントする必要がある場合は、以下のスクリプト例を使用して、DBFS CRSリソースを作成します。

```
#!/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"
```

注：DEPNAME値は、ロールベース・サービス名ではなく、データベース・サービス名にする必要があります。

Active Data Guardを使用していない場合は、以下のCRSリソース作成スクリプト例を使用して、データベース・サービス名の代わりにロールベース・サービス名を使用します。

```
#!/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"
```

新しいDBFS CRSリソースをテストします。それには、DBFSへの読取り専用マウントが必要なすべてのOracle RACノードのCRSを使って、ファイル・システムをマウント/アンマウントします。

```
% 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
```

Data Guardのスタンバイ・ホスト上にDBFS CRSリソースを必ず作成してください。

Active Data Guard、アクティブ・スタンバイのOracle RACノードを使用している場合に、Data Guardプライマリ上で新しいCRSリソースを開始すると、データベースが起動されるか、非Active Data Guard構成でプライマリ・データベースになったときに、DBFSは引き続き読取り専用モードでマウントされます。

フル・スタック・ロール移行

このホワイト・ペーパーで紹介した構成ベスト・プラクティスを実装すると、データベース、およびDBFSファイル・システムにあるすべてのデータを含むフル・スタック・ロール移行が、標準的なData Guardロール移行と同じように実行されます。DBFSがCRSによって管理され、ロールベース・サービスを使用する場合、Data Guardスイッチオーバーにより、旧プライマリ・データベース上のファイル・システムが自動的にアンマウントされ、Data Guardスイッチオーバー操作の一部として、同期されたファイル・システムが新しいプライマリ・データベースに再マウントされます。

Data Guardのスイッチオーバー

次に例を示します。

```
DGMGRL> switchover to ‘GGDGS’
```

ロール移行の完了後、DBFSは現在のプライマリ・データベースに読取り/書込みモードで自動的にマウントされます。読取り専用マウント（構成されている場合）も複数のOracle RACノードやアクティブ・スタンバイ・データベースにマウントされます。フル・スタック・スイッチオーバーは、データ損失をゼロにし、計画メンテナンス・アクティビティとして利用されます。

Data Guardのフェイルオーバー

次に例を示します。

```
DGMGRL> failover to ‘GGDGS’
```

ロール移行の完了後、DBFSは現在のプライマリ・データベースに読取り/書込みモードで自動的にマウントされます。読取り専用マウント（構成されている場合）も複数のOracle RACノードやアクティブ・スタンバイ・データベースにマウントされます。REDOにデータベースとDBFSファイル・システム双方の変更が含まれているため、データベースとDBFSファイル・システムは同じポイントにフェイルオーバーします。

ファスト・スタート・フェイルオーバー（FSFO）が構成されている場合、自動フェイルオーバーの後、DBFSは現在のプライマリ・データベースに読取り/書込みモードで自動的にマウントされます。

Data Guardスタンバイ・データベースの復元

Data Guardのフェイルオーバー後、旧プライマリ・データベースをスタンバイとして復元する際、DBFSで必要な特別な手順はありません。アクティブ・スタンバイ・データベースにDBFSへの読取り専用マウントが構成されている場合、スタンバイ・データベースの復元後、DBFSは読取り専用モードで自動的にマウントされ、データベースの残りの部分と同期されます。

次に例を示します。

```
DGMGRL> reinstate database ‘GGDGP’
```

まとめ

DBFSとOracle Data Guardに上記の構成を実装した後、フル・スタック・ロール移行を実行することは、標準的なData Guardロール移行コマンドを実行することと同じくらい簡単です。スイッチオーバーかフェイルオーバーかに関係なく、ロール移行後は、DBFSファイル・システムをアンマウントまたはマウントするために必要な追加の手順はありません。

Oracle Corporation, World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065, USA

著者：Michael Ramchand、Peter Wilson、Martien Ouwens

海外からのお問い合わせ窓口

電話：+1.650.506.7000

ファクシミリ：+1.650.506.7200

Integrated Cloud Applications & Platform Services

CONNECT WITH US

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

フル・スタック・ロール移行 – Oracle DBFSおよびOracle Data Guard
2016年8月
著者：Stephan Haisley

Oracle is committed to developing practices and products that help protect the environment