最終更新 2013/04/07

## RAC テクノロジの概要

- RAC 環境を構成するソフトウェア
- リソース管理
- インスタンスリカバリ

## インストール

- 事前準備
- Clusterware のインストール
- 自動ストレージ管理のインストール
- データベースソフトウェアのインストール
- Enterprise Manager Grid Control のインストール
- 管理エージェントのインストール

## データベースの作成

- 事前準備
- クラスタデータベースの作成
- シングル環境から RAC 環境への変換

## RAC データベースの管理

- Enterprise Manager (Grid Control)
- RAC データベースの起動と停止
- RAC 環境での初期化パラメータ
- RAC 環境における REDO と UNDO
- RAC 環境におけるその他の構成

## 自動ストレージ管理の構成

- RAC 環境における ASM インスタンス
- ASM ディスクグループ
- ASM ファイル

## RAC データベースのバックアップリカバリ

- RAC 環境でのアーカイブログモード
- RAC 環境で RMAN 構成
- RAC データベースのバックアップ
- RAC データベースのリカバリ

## RAC データベースの監視とチューニング

- RAC 固有のチューニング
- RAC 固有の待機イベント
- RAC 固有のシステム統計情報
- RAC 環境のチューニング方法
- RAC 環境のスナップショット

## サービス

- RAC 環境におけるサービス
- サービスの作成
- サービスの管理
- サービスと Oracle 機能の連携

## 接続の高可用性

- 接続時ロードバランシングの構成
- 高速アプリケーション通知 (FAN)
- 透過的アプリケーションフェイルオーバー (TAF)

## 高可用性の設計

- 環境に応じた最大可用性アーキテクチャ
- Data Guard
- RAC と Data Guard の連携
- RAC 環境へのパッチ適用
- 最適なストレージ構成

## Oracle Clusterware の管理

- Oracle Clusterware スタック
- クラスタリソースの管理
- Oracle Clusterware によるアプリケーションの保護
- CRS フレームワークの利用
- ネットワーク構成の変更

## 投票ディスクと OCR ファイルの調整

- 投票ディスク構成
- 投票ディスクのバックアップ/リカバリ
- Oracle Cluster Registry (OCR) の構成
- OCR ファイルのバックアップ/リカバリ

## Oracle Clusterware コンポーネント診断

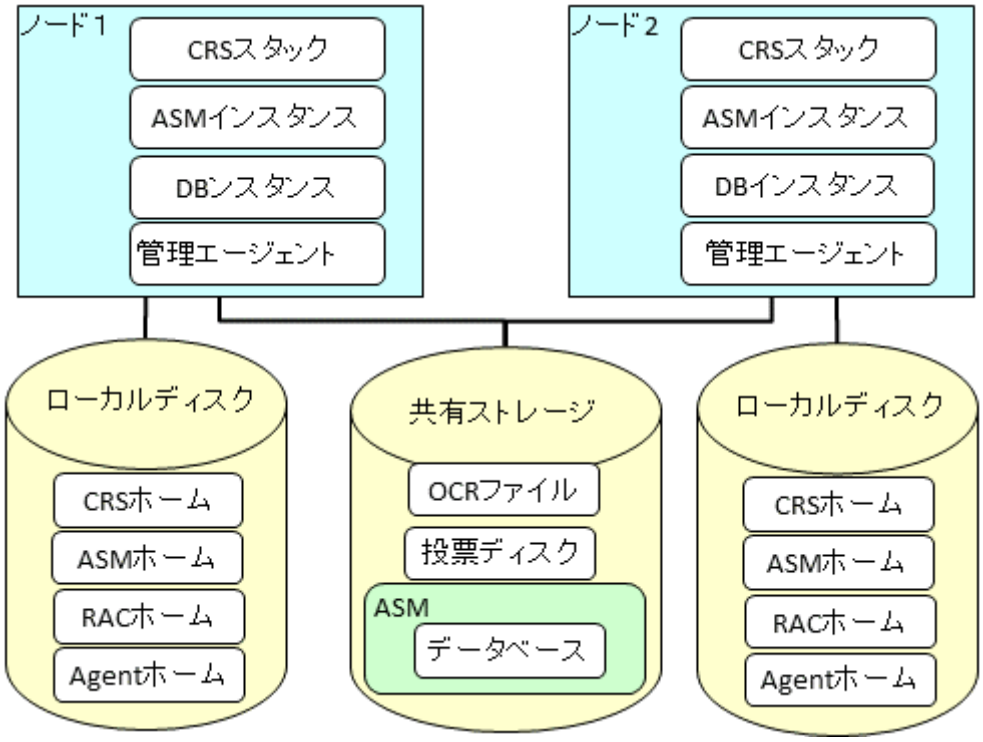
- クラスタ検証ユーティリティ (cluvfy)
- Oracle Clusterware の診断 (ログ)
- 各種デバッグ

## ノード追加と削除

- 新規ノードの追加
- ノードの削除

## ■RAC テクノロジの概要

### RAC 環境を構成するソフトウェア




### インストールソフトウェア

| ソフトウェア | 説明 |
|---|---|
| ClusterWare<br>(CRS ホーム) | Oracle Clusterware。DB やクラスタ構成を管理する（ORA_CRS_HOME）<br>※ORACLE_BASE 配下に作成することも可能だが非推奨 |
| ASM ｲﾝｽﾀﾝｽ | 自動管理ストレージ管理を利用する場合にインストールする（OCR や投票ディスクの配置は出来ない）<br>Clusterware とは別の場所にインストールすることで個別のバージョンアップが可能になる |
| RAC ﾎｰﾑ | データベースソフトウェアがインストールされる（ORACLE_HOME）<br>Oracle Clusterware がインストールされた状態で OUI を起動することで RAC 環境を構築可能 |
| 管理ｴｰｼﾞｪﾝﾄ | Oracle では OEM と Grid Control の両方がサポートされているが Grid Control を利用する際に必要 |

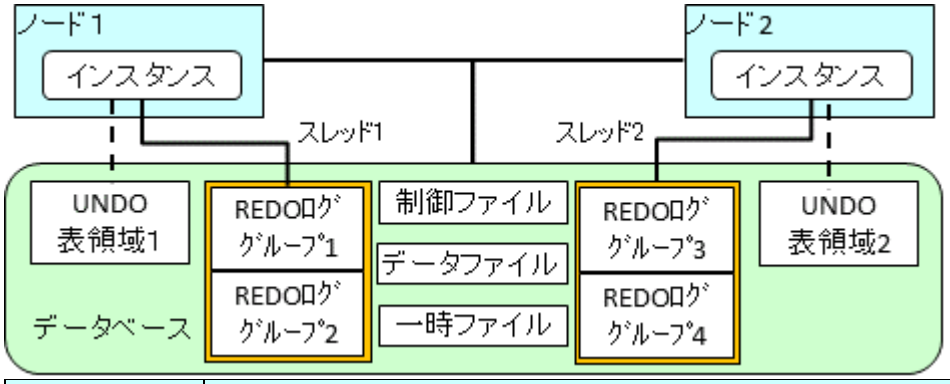
### CRS スタック

| プロセス名 | 説明 |
|---|---|
| CSSD | Cluster Synchronization Services デーモン。ASM のグループサービスの提供及びノード間の状態の監視を行う<br>インターコネクトを使用しノード間の同期を取り、インターコネクトに障害が発生したらノードを再起動する |
| CRSD | Cluster Ready Services デーモン。クラスタリソース（インスタンス、リスナー、VIP など Oracle Cluesterware に登録したアプリケーション）の起動、停止、監視を行う。クラスタリソース障害発生時はリソース再起動する |
| EVMD | Event Manager デーモン。クラスタイベント（ノード、クラスタの起動停止など）を受信し必要に応じてクラスタイベントの転送を行う |

### 共有ストレージ上のファイル

| ファイル名 | 説明 |
|---|---|
| OCR ファイル | Oracle Cluster Registry ファイル。ｸﾗｽﾀﾘｿｰｽに関する情報を格納。CRSD がｸﾗｽﾀﾘｿｰｽを管理する時に使用する |
| 投票ディスク | インターコネクトを利用したノード間通信を管理しているファイル。CSSD がノード間障害の監視に使用している |

### クラスタデータベース




| 名称 | 説明 |
|---|---|
| REDOロググループ | インスタンス毎に2グループ必要で、各インスタンスからREDOロググループを識別する情報をスレッドと呼ぶ<br>グループ毎にメンバーファイルが紐付く（複数指定することでミラーリングさせることが可能） |
| UNDO表領域 | 自動UNDO管理を使用する場合は各インスタンス毎に異なるUNDO表領域を割り当てる |

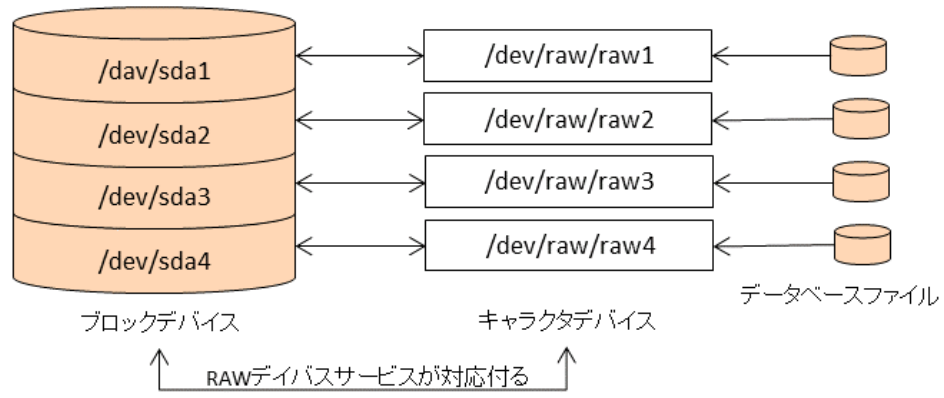
### 共有ストレージの技術と扱えるファイル

| タイプ | DBファイル | OCR/投票/ﾊﾞｲﾅﾘ/ﾊﾟｽﾜｰﾄﾞ | ｱｰｶｲﾌﾞﾛｸﾞ/ﾌﾗｯｼｭﾘｶﾊﾞﾘ |
|---|---|---|---|
| RAW | ○ | ○ | × |
| CFS | ○ | ○ | ○ |
| ASM | ○ | × | ○ |

### > RAWデバイス

1つのディスクパーティションに1つのファイルを配置する。OSからアクセスする為にRAWデバイスの設定が必要

OSが管理しないRAW(生)デバイスの為、ファイルシステムのようなキャッシュがなく書き込み速度が向上する

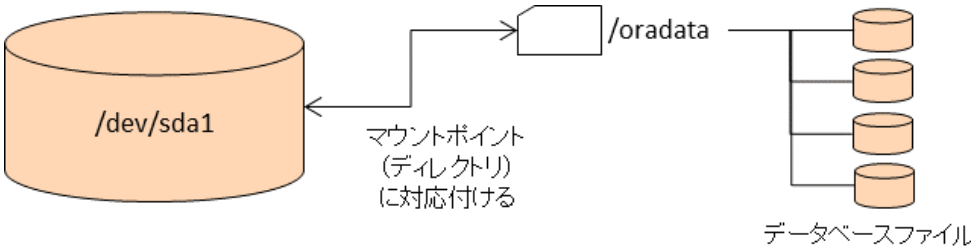



### > クラスタファイルシステム

共有ストレージ上での稼働を前提として開発されたクラスタファイル向けのファイルシステム

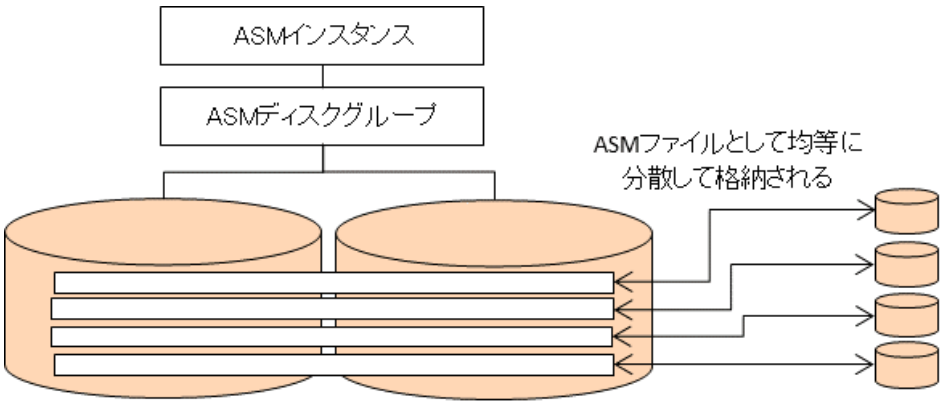
DBであれば自動拡張ONの構成も可能。LinuxとWindows向けのクラスタファイルシステムとして

OCFS(Oracle Cluster File System)がある



### ＞ 自動ストレージ管理：ASM

OracleDB 専用に開発されたストレージ技術。RAW デバイスのようにボリュームマネージャを準備する必要もなく、クラスタファイルシステムを用意する必要もない。ASM インスタンスという特殊な Oracle インスタンスを用意し「ASM ディスクグループ」という単位でストレージを管理する



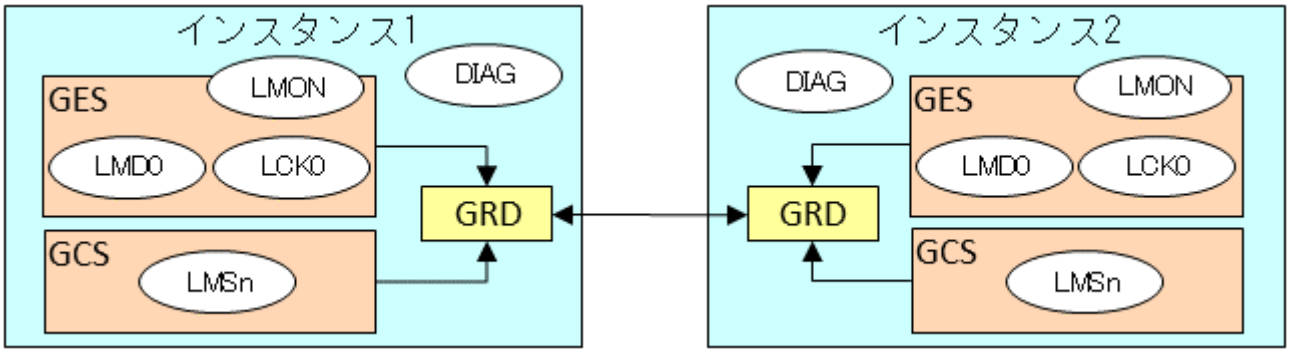

## リソース管理

### グローバルリソース

インスタンス間の同期のために使用されるリソース

グローバルリソースを管理するサービスが「グローバルキャッシュサービス(GCS)」と「グローバルエンキューサービス(GES)」

GCS と GES の情報は「グローバルリソースディレクトリ(GRD)」共有プール内で管理されインスタンス間で情報交換が行われる




| プロセス | 説明 |
|---|---|
| LMON | クラスタにインスタンスの追加・切り離しが行われることを検知しグローバルリソースの再構築を行う（ｸﾞﾛｰﾊﾞﾙｴﾝｷｭｰｻｰﾋﾞｽﾓﾆﾀ） |
| LMDO | ﾃﾞｯﾄﾞﾛｯｸの管理やロックモードの変換（排他モード、共有モードなど）の GES リソース調整を行う（ｸﾞﾛｰﾊﾞﾙｴﾝｷｭｰｻｰﾋﾞｽﾃﾞｰﾓﾝ） |
| LCK0 | 不正なロック、使用中ロックリストなどを作成し GES リソース要求を処理する（ﾛｯｸﾌﾟﾛｾｽ） |
| LMSn | インスタンス間でのメッセージの流れを制御し、キャッシュー慣性のためのデータブロック転送等の GCS リソース調整を行う（ｸﾞﾛｰﾊﾞﾙｷｬｯｼｭｻｰﾋﾞｽﾌﾟﾛｾｽ） |
| DIAG | インスタンス内の状態を監視し、インスタンス内のプロセスが異常終了した場合に診断データを保持する（診断プロセス） |
| **名称** | **説明** |
| GCS | キャッシュフュージョンという仕組みを利用しキャッシュー貫性を提供するサービス<br>① ノード A がデータを取得（データはディスクから読み取られバッファキャッシュに蓄えられる）<br>② ノード B が同じデータを取得を要求 ※<br>③ ノード A はノード B へブロック転送を行う（ブロック転送によりディスクの I/O は発生しない）<br>※ノード B はリソースマスターへ要求を行い、リソースマスターはブロックを保持しているノード A へ転送指示を行う |
| GES | キャッシュー貫性以外の排他制御を提供するサービス。DML によるロック処理などノード A で行ロックが発生しているテーブルに対してノード B が同じ行ロックする SQL をした場合、行ロックを保持したままキャッシュフュージョンが行われる |

| | |
|---|---|
| ﾘｿｰｽﾏｽﾀｰ | GCS リソース（データブロック）と GES リソース（エンキュー）の対象リソースに関する全ての情報管理を任されたインスタンス<br>起動している何れかのインスタンス（自動的に分散）がリソースマスターになる |
| ﾊﾟｽﾄ<br>ｲﾒｰｼﾞ | データブロックの要求が現行ブロックの時、キャッシュフュージョンによってブロック転送を行った保持側インスタンスのデータ<br>ブロックは使用済みバッファとして保持されディスクに書き込まれるまでパストイメージの状態になる<br>最新状態のブロックを保持するインスタンスがクラッシュした場合、<br>パストイメージを開始地点としてリカバリを実行することで高速なリカバリが可能となる |
| 再ﾏｽﾀﾘﾝｸﾞ | どのインスタンスがリソースマスターになるかは自動的に決定され適切に分散される<br>より多くのリソースを参照しているインスタンスにリソースマスターを再配置する |

## インスタンスリカバリとクラッシュリカバリ

| リカバリタイプ | 障害インスタンス | リカバリタイミング |
|---|---|---|
| インスタンスリカバリ | 一部のインスタンス | 障害発生後、残存インスタンスによって実行される<br>グローバルリソースのリカバリが必要になる |
| クラッシュリカバリ | 全てのインスタンス | 次回いずれかのインスタンスが起動した時に実行される<br>シングル環境のリカバリと同じ |

### インスタンスリカバリの流れ

**① 残存インスタンスが障害を検知**

**② リソース再構築**

LMON が GES と GCS の再マスタリングを行う（GRD(グローバルリソースディレクトリ)のリカバリ）

再マスタリング中は残存インスタンスでも GES と GCS を必要とする処理は一時停止する

**③ リカバリ対象ブロックの識別**

SMON がリカバリの必要なデータブロックを識別する（1st pass log read）

REDO ログファイルを読み込みデータファイルに書き込まれていないデータブロックを識別する（非同期 I/O）

**④ リカバリ対象ブロックの取得**

SMON がリカバリに必要なブロックを取得する。残存インスタンスにパストイメージが存在すればそれを使用し

存在しなければデータファイルからデータブロックを読み込む。リカバリ対象となるデータブロックの取得が完了すると

残存インスタンスの一時停止していた処理は再開され他のデータブロックは全て使用可能になる

**⑤ ロールフォワード**

SMON によって REDO レコードがデータブロックに適用される（2nd pass log read）

障害が発生したスレッドの REDO ログファイルが再度読み込まれる。ロールフォワードが完了したデータブロックは使用可能になる

**⑥ ロールバック**

SMON またはデータブロックにアクセスしたサーバープロセスによってコミットされていないトランザクションの

ロールバック（UNDO データの適用）が行われる。すべてのロールバックが完了すればリカバリ完了

## ■インストール

**事前準備** (RHEL4.0 にインストールする場合)

Linux 環境の場合は、hangcheck-timer モジュールを組み込み、カーネルがハングアップしているか監視させる

lsmod で hangcheck-timer が組み込まれていなければ/etc/rc.local に以下の行を追加する

```
/sbin/insmod /lib/modules/`uname -r`/kernel/drivers/char/hangcheck-timer.ko hangcheck_tick=10 hangcheck_margin=40
```

10 秒ごとにノードの死活確認し、40 秒間応答がなければシステムが再起動する（50 秒 OS 無応答であれば再起動）

**・各ノードで OSINVENTORY グループ、OSDBA グループ、Oracle オーナーとなる OS ユーザーを作成する**

```
# groupadd oinstall                  // OSINVENTORY グループ作成
# groupadd dba                       // OSDBA グループ作成
# useradd -g oinstall -G dba oracle // Oracle ユーザー作成
```

DB スケジューラの外部ジョブの為に非特権ユーザー（nobody）の存在を確認する。存在しない場合は作成する

**・シェル制限**

PAM によるユーザー認証に対してオープン可能な最大ファイル数と、使用可能なプロセス最大数を追加する

/etc/security/limits.conf

```
* soft nproc 2047
* hard nproc 16384
* soft nofile 1024
* hard nofile 65536
```

/etc/pam.d/login

```
session required /lib/security/pam_limits.so
session required pam_limits.so
```

soft は個々ユーザーで変更可能なので Oracle オーナーのログインシェルなどで hard と同じ値にしておく

最大ファイル数：ulimit -n 65536

最大プロセス数：ulimit -u 1024

**・カーネルパラメーター**

/etc/sysctl.conf

| | |
|---|---|
| kernel.sem = 256 32000 100 142 | セマフォ ID ごとの最大セマフォ数、システム全体のセマフォ数<br>セマフォコールごとの最大操作数、セマフォ ID の数 |
| kernel.shmmni = 4096 | 共有メモリ・セグメントのバイト単位の大きさの下限 |
| kernel.shmmax = 2147483648 | 共有メモリ・セグメントのバイト単位の大きさの上限 |
| kernel.shmall = 2097152 | システム全体の共有メモリ・ページの最大数 |
| fs.file-max = 131072 | 各プロセスがオープンできる最大ファイル数 |
| net.ipv4.ip_local_port_range = 1024 65000 | TCP/IP の送信用ポート範囲を指定 |
| net.core.rmem_default = 262133 | 受信ソケットバッファのデフォルトサイズ（byte） |
| net.core.wmem_default = 262133 | 送信ソケットバッファのデフォルトサイズ（byte） |
| net.core.rmem_max = 1048576 | 受信ソケットバッファの最大サイズ（byte） |
| net.core.wmem_max = 1048576 | 送信ソケットバッファの最大サイズ（byte） |

## ユーザー等価性

RAC 構成の各ノードでユーザーID/グループ ID が同じユーザーを用意しノード間でパスワードを聞かれなく

SSH または RSH でログインできる環境を用意する。SSH での構成方法を以下に示す

**① 各ノードで公開鍵と秘密鍵を作成**

```
$ cd ~oracle/.ssh
$ ssh-keygen -t rsa    // 公開鍵作成
$ ssh-keygen -t dsa    // 秘密鍵作成
```

パスフレーズを入力した場合はインストールセッションで ssh-agent を起動し ssh-add コマンドでパスフレーズを設定する

**② 各ノードに公開鍵と秘密鍵を連結した authorized_keys を作成する**

```
$ ssh <node1> cat /home/oracle/.ssh/id_rsa.pub >> authorized_keys
  ⇒yes ⇒oracle ユーザーのパスワード
$ ssh <node1> cat /home/oracle/.ssh/id_dsa.pub >> authorized_keys
$ ssh <node2> cat /home/oracle/.ssh/id_rsa.pub >> authorized_keys
  ⇒yes ⇒oracle ユーザーのパスワード
$ ssh <node2> cat /home/oracle/.ssh/id_dsa.pub >> authorized_keys
$ scp authorized_keys <node2>:/home/oracle/.ssh
```

**③ 各ノードでパスワードなしで接続可能かテストを行う**

## ディレクトリ構成例

```
/u01                  インストールルートディレクトリ
  crs_home            $ORA_CRS_HOME  ディレクトリ所有者を root、グループを INVENTORY グループ（oinstall）にする
  app                   root 権限で実行されるプログラムが多数ある為、$ORACLE_BASE 配下に作成しないことを推奨
    oracle
      oraInventory
      product         $ORACLE_BASE  ディレクトリ所有者：oracle、グループ：oinstall
        10.2.0
          asm_1       ASM のインストールディレクトリ
          db_1        DB のインストールディレクトリ（$ORACLE_HOME）
          agent10g
```

## ハードウェア構成

### ＞ ネットワーク

NIC は外部と接続するパブリックネットワークと、ノード間通信のインターコネクトの 2 種類を作成する

NIC の定義順序はノードで全て合わせること。（例）eth0 がパブリック、eth1 がインターコネクト

Oracle Net Client の為に作成される IP アドレスとして VIP がある

ノード障害時に VIP が移動し、クライアントがダウンしたノードに接続しても直ぐにエラーを返し、

生きてるノードに接続できる。VIP がないとタイムアウトを待って生きてるノードに接続されるので時間がかかる

### ＞ 共有ストレージ

クラスタファイルシステム、RAW デバイス、自動ストレージ管理(ASM)の技術をどこで使用するか決定しておく

Oracle Clusterware のインストール時に OCR ファイルと投票ディスクが作成されるが、ASM には作成できない

RAW デバイスでの権限付与

| ファイル | ユーザー | グループ | 権限 |
|---|---|---|---|
| OCR ファイル | root | OSINVENTORY | 640 |
| 投票ディスク | oracle | OSINVENTORY | 660 |

## Clusterware のインストール

### インストール前の検証

```
./cluvfy/runcluvfy.sh stage -pre crsinst -n <NODE1>,<NODE2>[,····]
```

パブリックネットワークがプライベートクラスだと VIP 検証がエラーになるが無視して OK

## Clusterware のインストール

oracle ユーザーで Clusterware の　./runInstaller

**① Oracle Inventory をインストールするディレクトリとグループを指定**

$ORACLE_BASE/oraInventory と oracle ユーザーの group が自動で入っている

**② ホーム詳細の指定**

任意の名前、パスは$ORA_CRS_HOME 変数（Cluster のインストールディレクトリ）の値を入れる

**③ クラスタ構成の指定**

それぞれのホスト名は、/etc/hosts で IP アドレスに関連付けておく必要がある

| パブリックノード名 | プライベートノード名 | 仮想ホスト名 |
|---|---|---|
| パブリックネットワークのホスト名 | インターコネクトのホスト名 | VIP のホスト名 |

クラスタ構成ファイルを使用する場合は、上記 3 カラムをスペース区切りで 1 行ずつ記述したファイルを用意する

**④ ネットワークインターフェース指定**

パブリックネットワークと、プライベートネットワーク（インターコネクト用）の IF 名とネットワークアドレス

**⑤ OCR と投票ディスクの場所の指定**

共有ストレージが冗長構成になっている場合は「外部冗長性」を選択

明示的に複数ファイル（ORC は 2 つ、投票ディスクは 3 つ必要）を指定する場合は「通常の冗長性」を選択する

RAW デバイスを指定する場合はデバイス名を指定する

**⑥ ローカルノードにファイルがコピーされリモートにコピーされていく**

**⑦ 構成スクリプトの実行**

root ユーザーになり全てのノードで、orainstRoot.sh 、root.sh を実行する（root.sh は 1 ノードずつ実行すること）

最初に root.sh を実行するノードで OCR と投票ディスクが作成される

最後に root.sh を実行するノードで VIPCA が起動されノードアプリケーション(GSD、ONS、VIP)が作成される

パブリックネットワークがプライベートアドレスだと VIP 作成でエラーになるが無視しても良い

**⑧ VIPCA を手動で実行する**

root ユーザーで　$ORA_CRS_HOME/bin/vipca

パブリックにする NIC を指定後、必要な情報を入力し OK 押下

| ノード名 | IP 別名 | IP アドレス | サブネットマスク |
|---|---|---|---|
| ホスト名(パブリックネットワーク) | VIP の別名 | VIP の IP アドレス | VIP のサブネットマスク |

## Clusterware インストール後の確認

各ノードで CRS スタックが実行される（crsd.bin、evmd.bin、ocssd.bin）ことを確認（ ps -ef|grep d.bin ）

CRS スタックは OS 起動時に自動起動するように/etc/inittab ファイルの最終行に追記される

以下のコンポーネントをまとめてノードアプリケーションと呼ぶ

| ｺﾝﾎﾟｰﾈﾝﾄ | 説明 |
|---|---|
| VIP | パブリックネットワークに作成する別名 IP アドレス。ノード停止やパブリックネットワーク停止時に残存ノードにフェイルオーバーされる為、ネットワーク障害を素早く検知できる |
| GSD | グローバルサービスデーモン。アプリケーション（DBCA、NETCA など）を RAC モードで動作するために必要なプロセス<br>GSD の起動は、CRS スタック（CSSD、CRSD、EVMD）が起動している必要がある |
| ONS | 通知サービス（Oracle Notification Service）リモートノードへのイベント通知を行うサービス |
| リスナー | Oracle Net で使用されるサーバー側のプロセス（この時点では作成されていない） |

srvctl status nodeapps -n ノード名 または crs_stat -t コマンドで起動状態の確認ができる

```
ora.orac1.gsd  application    ONLINE    ONLINE    orac1
ora.orac1.ons  application    ONLINE    ONLINE    orac1
ora.orac1.vip  application    ONLINE    ONLINE    orac1
ora.orac2.gsd  application    ONLINE    ONLINE    orac2
ora.orac2.ons  application    ONLINE    ONLINE    orac2
ora.orac2.vip  application    ONLINE    ONLINE    orac2
```

crs_stat -t のターゲット列が ONLINE であればｺﾝﾎﾟｰﾈﾝﾄが CRSD（Cluster Ready Servicesﾃﾞｰﾓﾝ）によって監視されてる

cluvfy stage -post crsinst -n <ﾉｰﾄﾞ名>,<ﾉｰﾄﾞ名> でインストール後の検証

### 構成のバックアップ

cp -ip root.sh root.sh.back

dd if=<投票ディスクの raw デバイス> of=<バックアップファイル名>

※raw ディスクは raw -qa で確認できる。CRS スタック起動中のままでもバックアップ可能

### 自動ストレージ管理のインストール

・単一ポイント障害を避けることを目的としており必須ステップではない

・ASM インスタンスのために行われるソフトウェアインストールである

### ASM インストール前の検証

cluvfy stage -pre dbinst -n <ﾉｰﾄﾞ名>,<ﾉｰﾄﾞ名>

構成されているノード一覧は、olsnodes -n コマンドで確認可能

### ASM のインストール

oracle ユーザーで Database の ./runInstaller

**① ホームの詳細の指定**

ASM のインストールディレクトリを指定。OracleDatabase とは別のディレクトリにすること

**② ハードウェアのクラスタインストールモードの指定**

クラスタインストールとノード名を全てチェックする

この画面を表示するには RAC 構成のノードアプリケーションが実行されている必要がある

**③ 構成オプションの選択**

ASM の構成を選択し、ASM インスタンスと、ASM ディスクグループを作成する

デフォルトでは読み書き可能なディスクパーティションを候補として表示する

あとから ASM ホームにインストールされた DBCA を使用して構成することも可能

**④ インストールの実行**

ローカルにファイルがコピーされた後、リモートへコピーされる

ここでリスナーが作成され ASM インスタンスと ASM ディスクグループが作成される

**⑤ 構成スクリプトの実行**

root ユーザーになり各ノードで、root.sh を実行する

**⑥ インストール後の確認**

crs_stat -t で、.asm と .lsnr の 2 種類が各ノードで追加されていること

```
ora....SM1.asm application    ONLINE    ONLINE    orac1    ノード 1 の ASM
ora....C1.lsnr application    ONLINE    ONLINE    orac1    ノード 1 のリスナー
ora....SM2.asm application    ONLINE    ONLINE    orac2    ノード 2 の ASM
ora....C2.lsnr application    ONLINE    ONLINE    orac2    ノード 2 のリスナー
```

## データベースソフトウェアのインストール

・Oracle Clusterware を事前にインストールしておく

・事前に CRS スタックを起動しておく

・Oracle Clusterware と$ORACLE_HOME は異なるディレクトリを指定する

・ディレクトリは$ORACLE_BASE 配下にする

・oracle ユーザーで Database の　./runInstaller　を実行

**① ホームの詳細の指定**

OracleDatabase のインストールディレクトリを指定

**② ハードウェアのクラスタインストールモードの指定**

クラスタインストールとノード名を全てチェックする

この画面が表示するには OUI を実行するノードでノードアプリケーションが実行されている必要がある

**③ 構成オプションの選択**

後から DBCA を使用して DB を作成する場合は　データベース・ソフトウェアのみインストール　を選択する

**④ インストールの実行**

ローカルにファイルがコピーされた後、リモートへコピーされる

**⑤ 構成スクリプトの実行**

root ユーザー各ノードで、root.sh を実行する（/etc/oratab ファイルと/usr/local/bin ディレクトリにスクリプトが作成される） 既に ASM ホームが作成されている場合は、上書きするかどうかの確認が出る。そのまま Enter で OK

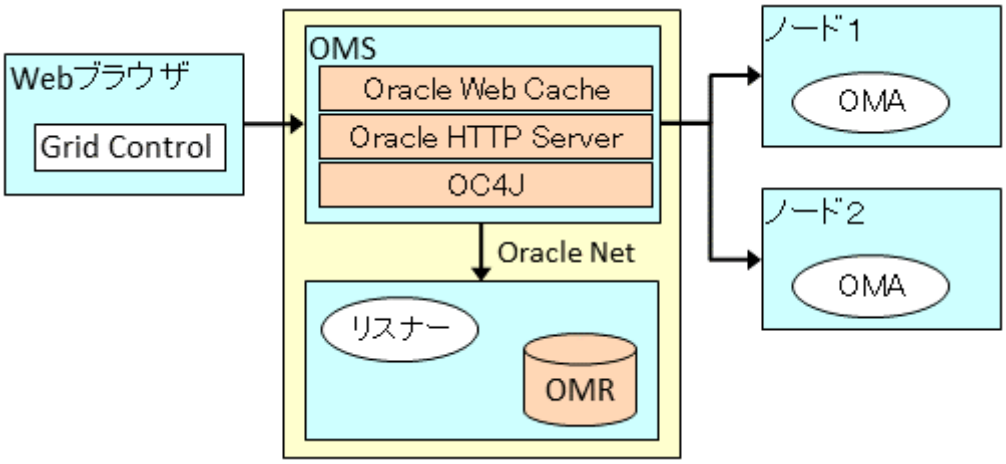
## Enterprise Manager Grid Control のインストール

・中間層（管理サービス、管理リポジトリ）と管理エージェントは事前にインストールが必要

・管理サービスと管理リポジトリ間は Oracle　Net（リスナー経由）を利用する

・oracle ユーザーで Enterprise Manager の　./runInstaller　を実行

### Enterprise Manager Grid Control の環境




| 構成 | 説明 |
|---|---|
| OMS | Oracle Management Service<br>Grid Control のための Web サーバーと Web アプリケーション。Oracle HTTP Server、Oracle Web Cache、OC4J が含まれる<br>デフォルトの URL は　http://ホスト名:4889/em |
| OMR | Oracle Management Repository<br>Enterprise Manager の情報を一元管理するデータベース。SYSMAN ユーザが所有している（LISTENER ポートで接続） |
| OMA | Oracle Management Agent<br>各ノードで Grid Control からの指示を実行する代理プロセス |

**① インストール場所の選択**

指定したディレクトリ配下に db10g、oms10g、agent10g サブディレクトリが作成される

**② 構成の指定**

インストールタイプで「既存データベース」を選択した場合は、既存 DB のホスト名、リスナーポート、サービス名、SYS ユーザーのパスワードの入力。「新規データベース」の場合は、DB 名とファイルの保存場所を指定する

**③ セキュリティオプション**

管理エージェントとの通信に使用するパスワード、リポジトリ DB 内のアカウントのパスワードを指定する

**④ インストールの実行**

OMR、OMS、OMA の順番でインストールされる

**⑤ 構成スクリプトの実行**

root ユーザーで、orainstRoot.sh、allroot.sh（db10g、oms10g、agent10g の各ディレクトリに root.sh を実行）を実行

## 管理エージェントのインストール

・Grid コントロールに接続して DL する方法とバイナリからインストールする方法（oracle ユーザーで runInstaller）がある

・oracle ユーザーで Enterprise Manager の　./runInstaller　を実行

### ① インストール・タイプの指定

「その他の管理エージェント」を選択する

### ② インストール場所の指定

指定したディレクトリ配下に agent10g サブディレクトリが作成される

### ③ ハードウェアのクラスタ・インストール・モードの指定

全てのノードが起動している状態で「クラスタ・インストール」を選択する

### ④ Oracle Management Service の場所の指定

管理サービス（OMS）がインストールされているホスト名と、接続するポート番号（デフォルトは 4889）を入力

### ⑤ エージェント登録パスワード指定

Grid Control インストール時に設定した管理エージェントのパスワードを入力

### ⑥ インストール

ローカルにインストールされた後、リモートへインストールされる

### ⑦ 構成スクリプトの実行

root ユーザーで root.sh の実行（/etc/oratab ファイルと/usr/local/bin ディレクトリにスクリプトが作成される）

各アプリを手動で起動する方法

| アプリ | 説明 |
|---|---|
| OMR | export ORACLE_SID=<SID>　sqlplus / as sysdba で接続し、startup |
| OMS | $ORACLE_HOME/opmn/bin/opmnctl startall |
| OMA | $ORACLE_HOME/bin/emctl start agent |

## ■データベースの作成

### 事前準備

| 環境変数 | 説明 |
|---|---|
| ORACLE_BASE | Oracle ソフトウェアのルートディレクトリ |
| ORACLE_HOME | Oracle ソフトウェア（ASM、DB）のインストールディレクトリ |
| ORA_CRS_HOME | Oracle Clusterware のインストールディレクトリ（パスは ORACLE_BASE に含まれないことを推奨） |
| ORACLE_SID | インスタンスの識別名。各ノードで異なる値にする |

CRS スタックが起動しているか確認（crsctl check crs）

DBCA がクラスタモードを認識する為に GSD が起動していることを確認（crs_stat -t）

DB 作成準備が整っているか確認（$ORA_CRS_HOME/bin/cluvfy stage -pre dbcfg -n <ノードリスト> -d $ORACLE_HOME）

### ASM ディスクグループの確認

DB を ASM で作成する場合は事前に ASM ディスクグループを作成しておく

<ASM ホーム>/bin/dbca で DBCA を起動し確認したり作成したりする

### クラスタデータベースの作成

・RAC 環境の DB は共有ストレージに配置され、各ノード毎にインスタンスが作成される

・DBCA で作成可能なのは、パスワードファイル、ASM インスタンス、サービス構成、クラスタリソース、Oracle Net 構成

・oracle ユーザーで、$ORACLE_HOME/bin/dbca を実行

#### ① データベースタイプの選択

「Oracle Real Application Clusters データベース」を選択する

#### ② 実行操作、ノード選択、テンプレート選択

データベース作成 ⇒ ノードは「すべて選択」⇒ テンプレートは「汎用」を選択していく

#### ③ データベース識別情報

グローバル・データベース名と、接頭辞を入力する

| 名称 | 説明 |
|---|---|
| SID 接頭辞 | ORACLE_SID 環境変数。DBCA は SID 接頭辞にノード番号を追加しインスタンス名を自動構成する |
| グローバルデータベース名 | ORCL.oracle.com → SERVICE_NAMES 初期化パラメータ<br>ORCL → DB_NAME 初期化パラメータ<br>oracle.com → DB_DOMAIN 初期化パラメータ |

#### ④ 管理オプション

Enterprise Manager を利用するか選択。管理エージェントが起動していれば Grid Control が選択された状態になっている

#### ⑤ データベース資格証明

SYS、SYSMAN、DBSNMP（Grid Control を使用時に作成される）ユーザーのパスワード指定

**⑥ 記憶域オプション**

DB に使用する共有ストレージ技術を選択。クラスタ・ファイルシステム、ASM、RAW の何れか

RAW デバイスではファイルをどの raw デバイスに格納するか記述したマッピングファイルを読み込むことが可能

```
spfile=/dev/raw/raw3      spfile、control<No>、redo<ｽﾚｯﾄﾞ番号>_<番号>の 3 つは固定の名前
control1=/dav/raw/raw8
control2=/dav/raw/raw9
redo1_1=/dav/raw/raw10
redo1_2=/dav/raw/raw11
system=/dev/raw/raw4      sysytem や、users などの表領域名はそのまま指定する
sysaux=/dev/raw/raw5
undotbs1=/dev/raw/raw6
users=/dev/raw/raw7
```

**⑦ データベースファイルの位置**

作成する DB ファイルを共通の位置に作成するか OMF（Oracle Managed Files）を使用するか選択する

OMF を使用する場合「REDO ログ゛及び制御ファイルの多重化」ボタンをクリックし多重化のための場所指定も可能

指定しない場合はフラッシュリカバリ領域に多重化ファイルが作成される

ASM ディスクグループ名を指定する場合は、先頭に＋を付けて指定する

**⑧ リカバリ構成**

フラッシュリカバリ領域やアーカイブログモードの有効化を指定

初期化パラメータ、DB_RECOVERY_FILE_DEST、DB_RECOVERY_FILE_DEST_SIZE が構成される

**⑨ データベースコンテンツ**

サンプルスキーマの使用有無、独自スクリプト実行指示。一番最初に選択するデータベーステンプレートで

カスタムデータベースを選択している場合は、ここで各種 Oracle 機能のインストールを選択する

**⑩ データベースサービス画面**

サービスの構成、TAF ポリシーを含むサービス構成の作成を行う。あとから DBCA で作成することも可能

**⑪ 初期化パラメータ**

メモリサイズや、キャラクタセットなどを指定。カスタムデータベースを選択している場合はブロックサイズの変更も可能

**⑫ データベース記憶域**

データベースファイルの場所やサイズを調整。カスタムデータベースを選択している場合は表領域の編集も可能

**⑬ データベース作成**

RAC 環境のクラスタ DB は DBCA で即時に作成され OCR ファイルに登録する作業も行われる

DB 作成スクリプトを保存しておき後で DB 作成することも可能だが、その場合は手動でクラスタリソースを登録する

クラスタリソースに登録されていることを確認する（crs_stat -t）

```
ora.orcl.db    application    ONLINE    ONLINE    orac1    グローバルデータベース
ora....l1.inst application    ONLINE    ONLINE    orac1    ノード 1 のインスタンス DB
ora....l2.inst application    ONLINE    ONLINE    orac2    ノード 2 のインスタンス DB
```

## シングル環境から RAC 環境への変換

DBCA、Enterprise Manager、rconfig コマンドの何れかを使用し既存のシングル環境 DB をクラスターDB に変換することが可能

| ルール | DBCA | rconfig | Grid Control |
|---|---|---|---|
| サポートするストレージ技術 | CFS、RAW、ASM | RAW、ASM | CFS |
| ｼﾝｸﾞﾙと RAC ｲﾝｽﾀﾝｽを起動するﾉｰﾄﾞ | 同一ノードでも異なるノードでも可能 | 同一ノードのみ可能 | 同一ノードのみ可能 |

### DBCA を使用した変換

**① 単一インスタンスのテンプレート化**

・DBCA を実行する

・実行する操作で「テンプレートの管理」を選択

・管理操作で「既存データベースを使用（データおよび構造)」

・ソースデータベースでテンプレートを作成するインスタンスを選択

・テンプレートプロパティで、テンプレートファイルの名前と保存場所を指定

・データベース関連ファイルの位置で、テンプレートを使用して DB を作成した時のファイル配置場所を指定

元の DB と同じ場所か、OFA に従って変更するか

**② Oracle Clusterware と RAC ソフトウェアのインストール**

RAC ソフトウェアのインストールは構成オプションで「データベースソフトウェアのみインストール」

**③ テンプレートファイルのコピー**

移行先の RAC 環境（<RAC ホーム>/assistants/dbca/templates）に①で作成した以下のテンプレートファイルを配置する

データベース構造ファイル（.dbc)、事前構成済みデータベースイメージファイル（.dfb)、 制御ファイル（.ctl）

**④ データベースの作成**

移行先の RAC 環境にて DBCA を起動し、転送しておいたテンプレートファイルを作成しクラスタ DB を作成する

### rconfig コマンドを使用した変換

雛形ファイル（$ORACLE_HOME/assistants/rconfig/sampleXMLs/ConvertToRac.xml）をコピーして作成する

<n:Convert verify="YES | NO | ONLY">で事前確認の動作を調整可能

| 設定値 | 説明 |
|---|---|
| YES | 前提条件を満たしていることを確認し変換を行う |
| NO | 前提条件を満たしていることを確認しないで変換を行う |
| ONLY | 前提条件を満たしていることの確認のみを行い変換作業は行わない |

作成した XML ファイルを引数に rconfig コマンドを実行する

$ORACLE_HOME/bin/rconfig　XML ファイル

### Grid Control を使用した変換

① 単一インスタンス DB が Grid Control に登録されていれば変換ウィザードを使用してクラスタ DB に変換可能

② Grid Control にログインし[ターゲット]タブ - [Database]をクリック

③ 単一のインスタンス DB 名をクリックし[管理]タブの[データベース変更]セクションにある

[クラスタデータベースへの変換]をクリック、ウィザードに沿って変換を行う

## ■RAC データベースの管理

Enterprise Manager (Grid Control)

クラスタ：〈クラスター名〉

└ クラスタ・データベース：〈グローバルデータベース名〉

└ データベース・インスタンス名：〈インスタンス名〉

### [クラスタ]ページ


| タブ | 説明 |
|---|---|
| ホーム | クラスタを構成するホストに関する情報やクラスタ DB へのリンクが表示される<br>[構成]セクションで、OS やハードの情報確認が可能 |
| パフォーマンス | 各ノードの CPU、メモリ、ディスク I/O などの負荷情報を表示 |
| ターゲット | Grid Control に含まれるターゲットの中でクラスタ配下に含まれるターゲット一覧を表示 |
| インターコネクト | 各ノードの NIC に関する情報を表示する |
| トポロジ | クラスターデータベースなどの配置構成表示 |

### [クラスタデータベース]ページ

| タブ | 説明 |
|---|---|
| ホーム | DB 全体のサマリー情報 |
| パフォーマンス | RAC 環境の負荷状況を監視し情報をグラフィカルに表示（CPU 実行キュー、ブロック転送、待機イベント等） |
| 管理 | データベースとスキーマの管理を行う |
| メンテナンス | バックアップ・リカバリ・エクスポート・インポートなどを行う |
| トポロジ | RAC 環境のホスト、リスナー、インスタンスなどの配置構成表示 |

### [データベースインスタンス]ページ

| タブ | クラスターデータベースページとの差異 |
|---|---|
| ホーム | [一般]セクションから対象インスタンスが動作している[ホスト]、使用している[リスナー]<br>[ASM インスタンス]ページにアクセスすることが可能 |
| パフォーマンス | ブロック転送に関する情報がない |
| 管理 | DB 全体と、インスタンスレベルの管理両方が表示される<br>DB 全体の管理の場合はリンクの先頭にクラスタデータベースを示すアイコンが表示される |
| メンテナンス | [ソフトウェアデプロイ]セクションに構成詳細を知るためのリンクが追加されている |

## サーバー生成アラート

・メトリック(Oracle サーバーで発生する統計情報)が閾値を超過するとアラートを出力してくれる機能

・[アラート]セクションに表示されるメッセージは最新アラートのみ。[アラート履歴]リンクで過去のアラート確認可能

・メトリック閾値はデフォルトで設定されているものと、ないものがある

・新規に設定したい場合は[メトリックとポリシー設定]ページを利用する

## ブラックアウト

・アラートなどの通知を一時的に停止する機能

・[ホーム]タブにある[ブラックアウト]ボタンで構成する

・スケジュール設定することで 1 回だけ、毎週繰り返す等のブラックアウトが設定可能

## RAC データベースの起動と停止

・複数のインスタンスで同じ DB を同時に OPEN している

・全てのインスタンスが停止したら DB は停止する

・srvctl や EM を利用して DB 管理するにはクラスタリソースに登録されている必要がある

```
$ srvctl add database -d DB名 -o $ORACLE_HOME            データベースリソースの登録
$ srvctl add instance -d DB名 -i $ORACLE_SID -n ﾉｰﾄﾞ名    インスタンスリソースの登録
```

### 起動停止を行うツール

| ツール | インスタンス単位 | データベース単位 |
|---|---|---|
| SQL*Plus | ○ | × |
| SRVCTL | ○ | ○ |
| Enterprise Manager | ○ | ○ |

※DB 単位は、該当 DB を OPEN している全てのインスタンスが対象になる

## SQL*Plus による起動・停止

・環境変数 ORACLE_SID を設定し、sqlplus で インスタンスに接続し、STARTUP / SHUTDOWN コマンドを実行

・SHUTDOWN TRANSACTIONAL のみシングル環境と異なり、全インスタンスのトランザクション終了を待つ

コマンド実行するインスタンスのみのトランザクションを待つ場合は、TRANSACTIONAL LOCAL を指定する

## SRVCTL による起動・停止

srvctl { start | stop } { instance | database } -d DB名 [オプション]

| オプション | 説明 |
|---|---|
| -i ｲﾝｽﾀﾝｽﾘｽﾄ | インスタンス名（複数指定はカンマ区切り）※instance 時のみ指定 |
| -o [ｵﾌﾟｼｮﾝ] | start ｺﾏﾝﾄﾞ：**open** \| **mount** \| nomount<br>stop ｺﾏﾝﾄﾞ ：normal \| transactional \| **immediate** \| abort<br>太字は省略時のデフォルトだが、instance(**open**) と database(**mount**)で異なる |
| -c 接続文字列 \| -q | 接続文字列は、scott/tiger@orcl1 as sysdba など<br>-q は接続文字列を標準入力で指定可能。-c、-q を省略すると / as sysdba になる |

## Enterprise Manager による起動・停止

### > インスタンスの停止

[データベースインスタンス]ページ、または[クラスタデータベース]ページから対象となるインスタンスを選択し、

[ホーム]タブの一般セクションにて起動・停止を行う

### > データベースの停止

[クラスタデータベース]ページの[ホーム]タブの一般セクションから行う

## RAC 環境での初期化パラメータ

・クラスタデータベース固有のパラメータがある

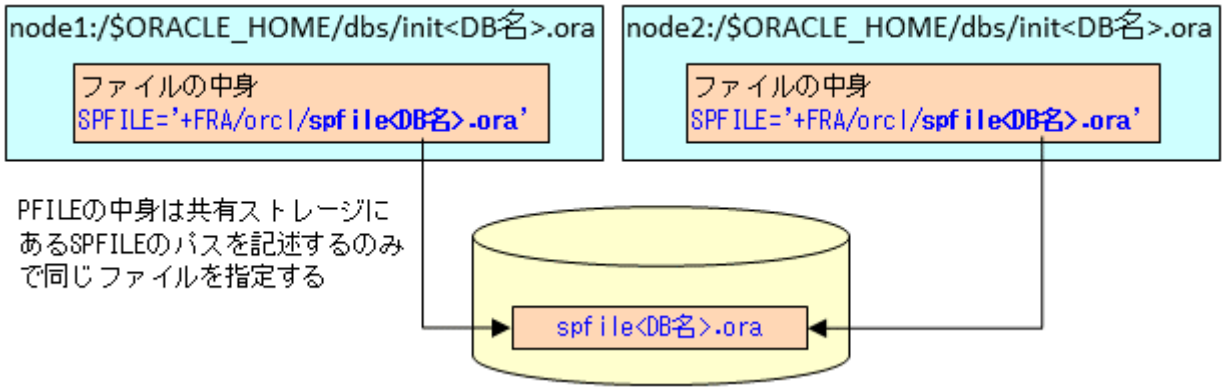
・複数インスタンスで異なる値にしなければいけないパラメータがある

## SPFILE の使用

・各インスタンスで使用する初期化パラメータファイルは共通で利用できるファイルを用意する

・テキストの PFILE を用意し、そのファイルには SPFILE の場所を記載するパラメータのみ記述することで同じ SPFILE を使う

| 利点 | 説明 |
|---|---|
| 管理の簡略化 | 1 つの SPFILE ですべてのインスタンスに関する初期化パラメータの確認が可能<br>RMAN では SPFILE を自動バックアップするように構成することが可能 |
| 整合性の維持 | 各インスタンスで同じ値にすべき初期化パラメータ変更時の設定ミスを防止可能 |




・DBCA を使用して DB を作成した場合は SPFILE 情報を OCR ファイルにも登録する為、

SQL*PLUS と SRVCTL でインスタンス起動した際の SPFILE 読み込み順序が異なる

| ツール | 説明 |
|---|---|
| SQL*PLUS | PFILE 内に記述されている初期化パラメータ（SPFILE）に従って SPFILE を読み込む |
| SRVCTL | OCR ファイル内に記述されている初期化パラメータ（SPFILE）に従って SPFILE を読み込む、情報がなければ SQL*PLUS と同様<br>srvctl config database -d <DB 名> -a で OCR ファイル内に SPFILE 情報があるか確認可能（null 以外であれば設定済）<br>srvctl config database -d <DB 名> -p 'SPFILE のパス' で OCR ファイル内に SPFILE 情報登録 |

## 初期化パラメータの変更

設定：ALTER SYSTEM SET 初期化ﾊﾟﾗﾒｰﾀ=値　[SCOPE={ MEMORY | SPFILE | BOTH }]　[SID='{* | ｲﾝｽﾀﾝｽ名}']

削除：ALTER SYSTEM RESET 初期化ﾊﾟﾗﾒｰﾀ　SCOPE=SPFILE SID='{* | ｲﾝｽﾀﾝｽ名}'

| オプション | 値 | 説明 | EM でのメニュー |
|---|---|---|---|
| SCOPE | MEMORY | インスタンスだけを変更し、SPFILE は変更しない | 現行 |
| | SPFILE | SPFILE だけを変更し、インスタンスに反映させない | SPFile |
| | BOTH | インスタンスと SPFILE 両方に反映 | 「～に適用する」のチェックを ON にする |
| SID | * | すべてのインスタンスを変更（デフォルト） | - |
| | ｲﾝｽﾀﾝｽ名 | 指定したインスタンスのみ変更 | - |

## RAC 環境で調整される初期化パラメータ

| パラメータ | 説明 |
|---|---|
| CLUSTER_DATABASE | TRUE で DB は共有モードとして複数インスタンスから使用できるようにする<br>FALSE は排他モードとして 1 つのインスタンスからのみ使用できるようにする |
| CLUSTER_DATABASE_INSTANCES | インスタンス数を設定。SGA 内に獲得されるインスタンス毎のメモリサイズの計算にも使用される |
| CLUSTER_INTERCONNECTS | インターコネクト用の NIC が複数ある場合や、同じノードの複数インスタンスに異なる NIC を割り当てる際に使用する。1 つのインターコネクトのみ使用している場合は設定不要 |
| THREAD | インスタンスで使用する REDO スレッド番号を設定。各インスタンスで異なる値にする必要あり |

## 同じ値にする必要のある初期化パラメータ

| パラメータ | 説明 |
|---|---|
| ACTIVE_INSTANCE_COUNT<br>※非推奨パラメータ | 2 つのインスタンスのみ存在するクラスタ DB で 1 を設定すると最初に起動したインスタンスがプライマリ（接続要求受付け）、2 番目に起動したインスタンスがセカンダリとして構成する<br>プライマリに障害が発生しない限りセカンダリのインスタンスに接続しない |
| ARCHIVE_LAG_TARGET | Data Guard 環境においてログスイッチを強制する間隔（秒）0 または 60～7200 の間を指定 |
| COMPATIBLE | DB が動作するリリース番号（10.2.0.1 など）以前のリリースとの下位互換を提供する |
| CONTROL_FILES | DB で使用する制御ファイル名のリスト |
| DB_BLOCK_SIZE | DB で使用する標準ブロックサイズ |
| DB_DOMAIN | ネットワーク上の DB を一意に識別するために使用できるドメイン名（分散 DB の際に設定） |
| DB_FILES | DB でオープンできる最大データファイル数 |
| DB_NAME | データベース識別名 |
| DB_RECOVERY_FILE_DEST | フラッシュリカバリ領域として使用する場所 |
| DB_RECOVERY_FILE_DEST_SIZE | フラッシュリカバリ領域としての最大サイズ |
| DB_UNIQUE_NAME | 一意な DB 名。DB_NAME と同じになるが Data Guard 環境ではプライマリと異なる名前を設定する |
| INSTANCE_TYPE | データベースインスタンス（RDBMS）か、ASM インスタンス（ASM）を識別する |
| LICENSE_MAX_USERS | DB で作成できる最大ユーザー数（同じ値にすることを推奨） |
| PARALLEL_EXECUTION_MESSAGE_SIZE | パラレル実行用のメッセージサイズ<br>値が大きいほど、パフォーマンスを向上させるために多くのメモリーが必要 |
| PARALLEL_MAX_SERVERS | パラレル実行プロセスの最大数 |
| REMOTE_LOGIN_PASSWORDFILE | 1 つ以上のデータベースがパスワードファイルを使用（SHARED）<br>1 つのデータベースがパスワードファイルを使用（EXCLUSIVE）、使用しない（NONE） |

| | |
|---|---|
| SPFILE | SPFILE（サーバーパラメータファイル）の場所を指定 |
| TRACE_ENABLED | Oracle の実行履歴とコードのトレースを制御（TRUE にするとエラー時にトレースを出力する） |
| UNDO_MANAGEMENT | 自動 UNDO 管理を使用（AUTO）使用しない（MANUAL）か識別する |
| UNDO_RETENTION | 自動 UNDO 管理における UNDO 保存期間の下限値を指定（同じ値にすることを推奨） |

### 異なる値にする必要のある初期化パラメータ

| パラメータ | 説明 |
|---|---|
| INSTANCE_NAME | インスタンスを識別する名前 |
| INSTANCE_NUMBER | インスタンスを識別する番号。空きリストグループにマップするインスタンスの識別に使用<br>※そのセグメントに割り当て済み且つデータを挿入することができるデータブロックの DB(順位付きのリスト) |
| THREAD | インスタンスで使用する REDO スレッド番号。INSTANCE_NUMBER と同じにすることを推奨 |
| UNDO_TABLESPACE | 自動 UNDO 管理におけるインスタンスで使用する UNDO 表領域名 |
| ROLLBACK_SEGMENTS | 手動 UNDO 管理におけるオンラインにするロールバックセグメント名。省略時はﾊﾟﾌﾞﾘｯｸﾛｰﾙﾊﾞｯｸｾｸﾞﾒﾝﾄを使用 |

### RAC 環境における REDO と UNDO

・インスタンス毎に REDO スレッドを 1 つ

・REDO スレッドに紐付く REDO ロググループを最低 2 つ用意する

・REDO ロググループのメンバー（ファイル）は 2 つ以上必要（REDO グループは 2 つ必須なので、4 ファイル必要になる）

・インスタンス毎に異なる UNDO 表領域を割り当てる

・REDO ログメンバー、UNDO 表領域のデータファイルは共有ストレージに配置する必要がある

### REDO ログファイルの定義

・REDO スレッドの変更は、

REDO ロググループの追加、スレッドの有効化、初期化パラメータ変更、インスタンス再起動　の手順が必要

・REDO ログに書き込まれるタイミングは以下の通り

① REDO ログバッファの使用部分が全体の３分の１に達した時
② タイムアウト発生時（３秒おき）
③ DBWR がデータベースバッファキャッシュ内の変更済みブロックをデータファイルに書き込む前
④ トランザクションのコミット時

### > REDO ロググループ

・V$LOG　　REDO ロググループの状態確認（status が current になっているのが現在書き込まれているグループ）

・V$LOGFILE　REDO グループとグループに紐付くメンバー（ファイル）が確認できる

| 操作内容 | 説明 |
|---|---|
| グループ追加 | ALTER DATABASE ADD LOGFILE { INSTANCE 'ｲﾝｽﾀﾝｽ名' \| THREAD ｽﾚｯﾄﾞ番号 } GROUP ｸﾞﾙｰﾌﾟ番号 'ﾌｧｲﾙ名' SIZE ｻｲｽﾞ<br>REDO ロググループの作成時に指定するスレッド番号によって対応付けられるインスタンスが決定する<br>ファイル名に ASM を利用する場合は'+DATA'のように指定する（ASM なので Oracle が自動でファイル名を決定する） |
| グループ削除 | ALTER DATABASE DROP LOGFILE GROUP ｸﾞﾙｰﾌﾟ番号<br>スレッドを無効化しておく必要がある。削除してもメンバーになっているファイルは残ったまま |
| メンバー追加 | ALTER DATABASE ADD LOGFILE MEMBER 'ﾌｧｲﾙ名' TO GROUP ｸﾞﾙｰﾌﾟ番号 |
| メンバー削除 | ALTER DATABASE DROP LOGFILE MEMBER 'ﾌｧｲﾙ名'<br>REDO ログが Current（v$log の status で確認）になっているメンバーは削除できない<br>Active になっている場合は、チェックポイントを実行してから削除する |

### > 制御ファイル

・新規のスレッド番号の場合、制御ファイルでも新しいスレッド番号を有効にする必要がある

・有効にするには 2 つ以上の REDO ロググループを作成しておく必要がある

・V$THREAD で有効化しているスレッド番号を確認（status が open になっていれば有効）

| 操作内容 | SQL 文 |
|---|---|
| 有効にする | ALTER DATABASE ENABLE { INSTANCE ｲﾝｽﾀﾝｽ名 \| THREAD ｽﾚｯﾄﾞ番号 } |
| 無効にする | ALTER DATABASE DISABLE { INSTANCE ｲﾝｽﾀﾝｽ名 \| THREAD ｽﾚｯﾄﾞ番号 } |

### > 初期化パラメータ：THREAD

・スレッド番号は各インスタンスで異なる値を指定する

・THREAD パラメータに指定できる最大値は create database 文、create controlfile 文の MAXINSTANCES 属性で指定（10gR2 以降は自動拡張される）

・静的パラメータの為、設定時はインスタンスの再起動が必要

| 操作内容 | SQL 文 |
|---|---|
| パラメータ設定 | ALTER SYSTEM SET THREAD=ｽﾚｯﾄﾞ番号 SCOPE=SPFILE SID='ｲﾝｽﾀﾝｽ名' |

## REDO ログファイルの管理

| アクション | 説明と強制的に実行する場合の SQL 文 |
|---|---|
| ログスイッチ | LGWR が、現行の REDO ロググループがいっぱいになった場合次のグループに書き込みを開始すること<br>ALTER SYSTEM SWITCH LOGFILE<br>ALTER SYSTEM ARCHIVE LOG CURRENT [ INSTANCE 'ｲﾝｽﾀﾝｽ名' ] CURRENT |
| チェックポイント | DBWR がデータベースバッファキャッシュ内の変更済みブロックをデータファイルに書き込む<br>ALTER SYSTEM CHECKPOINT [ LOCAL \| GLOBAL ]<br>チェックポイントは、以下のタイミングで実行される<br>・ログスイッチが実行された時<br>・ABORT 以外のオプションでインスタンスをシャットダウンした時<br>・LOG_CHECKPOINT_INTERVAL で設定した容量の値まで LGWR が REDO ログファイルに書き込んだ時<br>・LOG_CHECKPOINT_TIMEOUT で設定した最大経過時間（秒数で指定）に達した時 |

## UNDO 表領域の定義と管理

・インスタンス毎に異なる UNDO 表領域が必要

・インスタンス毎に指定できる UNDO 表領域は 1 つ（UNDO_TABLESPACE 初期化パラメータで指定）

・パラメータ省略時は使用されてない UNDO 表領域を使用する

何れも使用できない場合は SYSTEM ロールバックセグメント使用（SYSTEM 表以外の更新は行えなくなる）

・パラメータを同じにすると 2 番目に起動するインスタンスが起動できない

・新しい UNDO 表領域を指定した場合、新規トランザクションは新しい UNDO 表領域を使用し、

既存トランザクションは元の UNDO 表領域を使う

## RAC 環境におけるその他の構成

### > グローバル動的パフォーマンスビュー

・全てのインスタンス状況を確認する　GV$ビュー　がある

・GV$ビューを使用する場合、パラレル実行（1 つのインスタンスに対して 1 つのパラレル実行サーバー（PZ99）が起動）が使用される

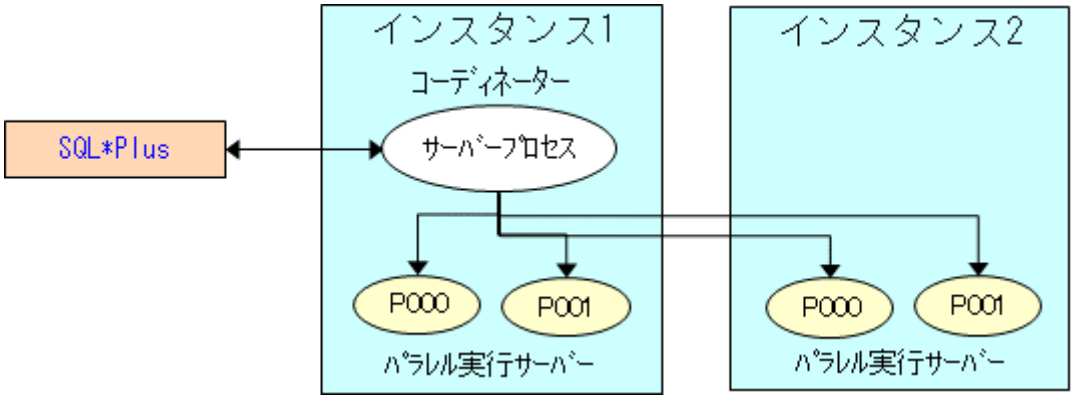
### > RAC 環境のパラレル実行

・パラレル実行はセッションに対応するｻｰﾊﾞｰﾌﾟﾛｾｽが SQL 文を実行するのではなく、複数のパラレル実行サーバーが実行する

・対応するサーバープロセスはコーディネータと呼ばれパラレル実行サーバーとの調整を行う

・コーディネータと同じノードでパラレル実行サーバーを起動するように調整するが、

アイドル状態のパラレル実行サーバーがない場合は別ノードで実行する




### > データベースの静止

・リソースマネージャを使用したセッション制限機能

・DB 静止を行うと SYS_GROUP グループに属するユーザー以外はセッションが確立できなくなる

・既存セッションには影響を与えず、新規セッションのみ接続不可

・DB 静止中は停止しているインスタンスの OPEN が出来ない（エラーとなる）

| 操作内容 | SQL 文 |
|---|---|
| 静止 | ALTER SYSTEM ENABLE RESTRICTED SESSION または DB 起動時に「STARTUP RESTRICTED」 |
| 解除 | ALTER SYSTEM UNQUIESCE |

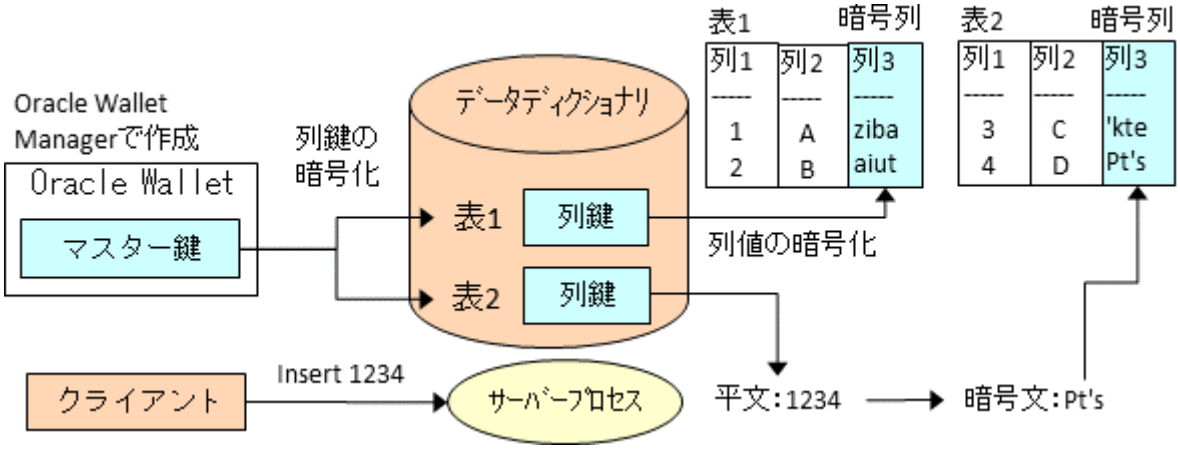
**› 透過的データ暗号化**

・データファイルと REDO ログファイル内のレコードを暗号化する

・クライアントからは意識する必要がない（透過的）暗号化機能

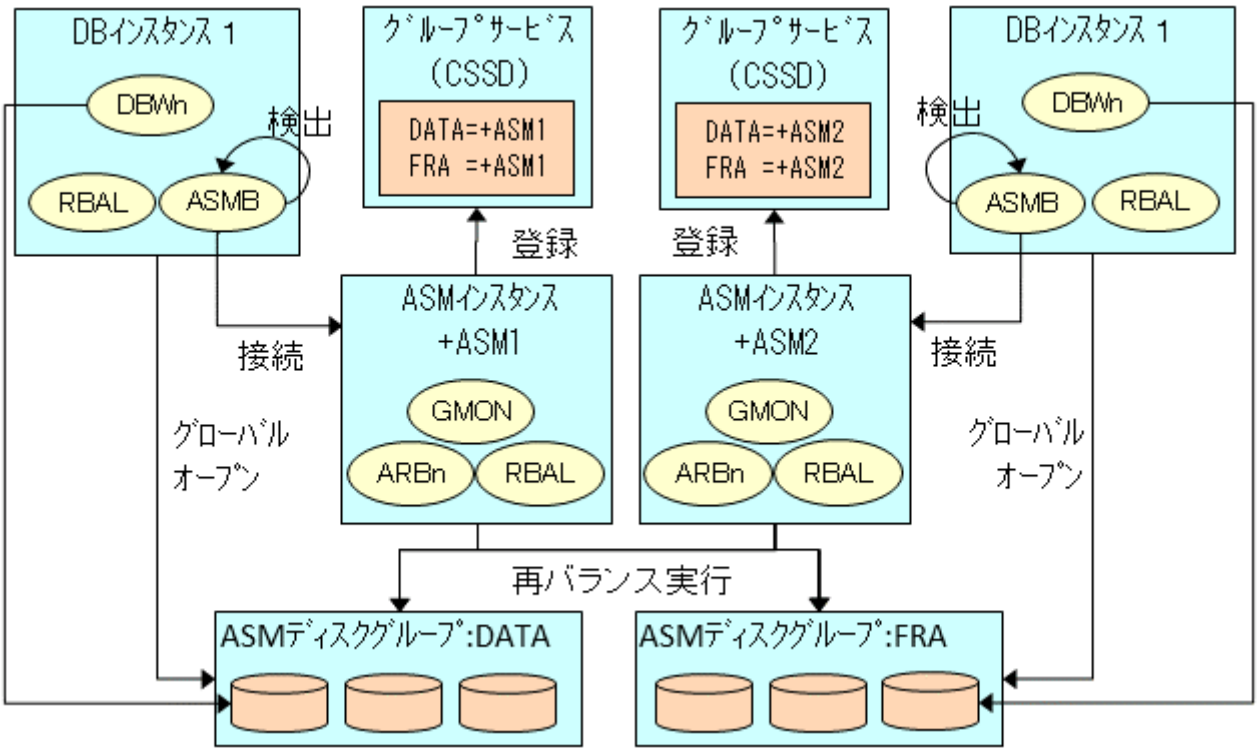
・インスタンス毎に「マスター鍵」、暗号化を行う列を含むテーブル毎に「列鍵」が必要

・RAC 環境では全てのインスタンスで同じ「マスター鍵」を使用する必要がある




**■自動ストレージ管理の構成**

**RAC 環境における ASM インスタンス**




| 名称 | 説明 | |
|---|---|---|
| ASM インスタンス | ASM ディスクグループをマウントし ASM ディスクグループの管理や再バランス（ASM ファイルの再配置）を実行する<br>ASM インスタンスは複数の DB インスタンスから接続することが可能。異なる DB で同じ ASM ディスクグループが使用可能 | |
| | GMON | ディスクグループモニター。ディスクグループに対するメンテナンス動作（メタデータの変更）を監査 |
| | RBAL | リバランスマスター。ディスクグループの再バランス動作を調整する |
| | ARBn | リバランサ。再バランスによるデータ移動を実行する。最大 11 プロセスまで起動可能 |

| | | |
|---|---|---|
| ﾃﾞｰﾀﾍﾞｰｽ ｲﾝｽﾀﾝｽ | DB ファイルを ASM ファイルとして作成した場合、DB インスタンスは ASM ファイルの配置に関する情報を取得する為に ASM インスタンスに接続し、ASM ファイルに直接アクセスできるようにする。ASMB と RBAL が追加起動される<br>DB インスタンスは 1 つの ASM インスタンスのみに接続し、常に同じノードの ASM インスタンスに接続する | |
| | ASMB | ASM バインド。グループサービスから ASM インスタンスと ASM ディスクグループ情報を取得し、ASM インスタンスに接続する。ASMB による接続を介して ASM インスタンスと定期的にメッセージ交換を行い統計が更新され、ASM、DB の両方が正常であることを確認する |
| | RBAL | リバランスマスター。ASM ディスクグループに属する ASM ディスクに対するグローバルオープンを実行する<br>オープンされた経路を使用して ASM ファイルに直接アクセスできるようになる |
| ｸﾞﾙｰﾌﾟｻｰﾋﾞｽ | DB インスタンスが ASM インスタンスを検出するために必要な接続情報を提供する。<br>ASM インスタンスが ASM ディスクグループをマウントすると、ASM ディスクグループへの接続に必要な情報をグループサービスに登録する。DB インスタンスはグループサービスを利用することで正しい ASM インスタンスへの接続情報を取得できる<br>グループサービスは CSSD（Cluster Synchronization Services デーモン）が提供する | |

### ＞ ASM インスタンスの初期化パラメータ

・CLUSTER_DATABASE が TRUE、INSTANCE_TYPE が ASM である（DB の場合は RDBMS）

・ASM インスタンス固有の初期化パラメータ（各ノードで異なる値に設定できるが通常は同じ値にする）

| パラメータ | 説明 |
|---|---|
| ASM_DISKGROUPS | ASM インスタンス起動時や、ALTER DISKGROUP ALL MOUNT コマンドでマウントさせる ASM ディスクグループリスト<br>パラメータなしの場合、ASM ディスクグループがマウントされると自動追加され、アンマウントすると削除される |
| ASM_DISKSTRING | ASM ディスクの検出文字列。省略時は NULL となり ASM インスタンスが読み書き可能な全てのディスクが対象となる |
| ASM_POWER_LIMIT | 再バランス速度。0～11 を指定。0 で再バランスしない。1 以上で ARBn のプロセス数に影響する<br>ALTER DISKGROUP コマンドの REBALANCE POWER 句で一時的な調整も可能 |

### ＞ ASM インスタンスの起動と停止

・ASM インスタンスを停止するには DB インスタンスが全て停止している必要がある

・ASM インスタンスを強制停止すると DB インスタンスも強制停止する

・ASM インスタンスが起動してない状態では DB インスタンスも起動できない

```
$ export ORACLE_HOME=/opt/app/oracle/product/10.1.0/asm
$ export ORACLE_SID=+ASM1
$ sqlplus / as sysdba
```

・srvctl コマンドによる ASM インスタンスの起動・停止

```
srvctl start asm -n ﾉｰﾄﾞ名 [-i ASM ｲﾝｽﾀﾝｽ名]  [-o { mount | nomount} ]  [-c 接続文字列 | -q]

srvctl stop  asm -n ﾉｰﾄﾞ名 [-i ASM ｲﾝｽﾀﾝｽ名]  [-o {normal | immediate | transactional | abort}]  [-c 接続文字列 | -q]
```

・srvctl を使用して ASM を管理するにはクラスタリソースとして登録され、DB と ASM の依存関係も必要

```
$ srvctl add asm -n orac1 -i +ASM1 -o /opt/app/oracle/product/10.1.0/asm
$ srvctl add asm -n orac2 -i +ASM2 -o /opt/app/oracle/product/10.1.0/asm
-- データベースインスタンスに依存関係を追加（-s で DB インスタンスに対する ASM の依存性）
$ srvctl modify instance -d orcl -i orcl1 -s +ASM1
$ srvctl modify instance -d orcl -i orcl2 -s +ASM2
```

依存関係が設定されていれば srvctl を使用して ASM を停止すれば DB インスタンスも自動で停止される、起動も同じ

### ＞ インスタンスリカバリとクラッシュリカバリ

・ASM インスタンスには ASM ディスクグループや ASM ファイルに関するメタデータを保存している

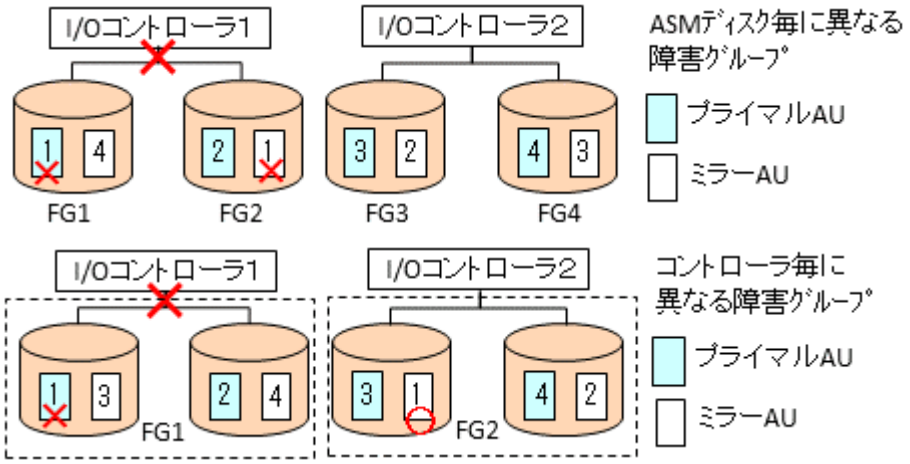
・ASM インスタンスに障害発生すると依存する DB インスタンスはクラッシュする

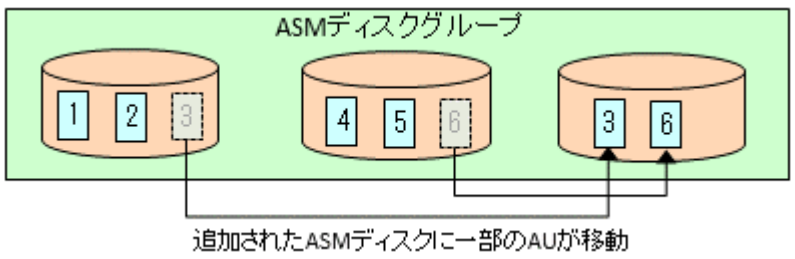
・DB インスタンスに障害が発生しても ASM インスタンスは影響なし

| リカバリタイプ | 障害インスタンス | リカバリタイミング |
|---|---|---|
| インスタンスリカバリ | 一部のインスタンス | 障害発生後、残存インスタンスによって実行される |
| クラッシュリカバリ | 全てのインスタンス | 次回何れかのインスタンスが起動したときに実行される |

## ASM ディスクグループ

### ＞ ASM ディスクグループの特徴

<table>
<tr><th>ASM3 つの特徴</th><th>説明</th></tr>
<tr><td>ストライプ化</td><td>データを 1MB 単位（Allocation Unit）で分割し ASM に属する複数の ASM ディスクに均等に配置する<br>REDO ログや、制御ファイルなどは AU 内を更に 128KB 単位で扱う事も可能</td></tr>
<tr><td>ミラー化</td><td>ASM は AU 単位でミラー化が行われプライマリ AU とミラーAU の配置方法が選択できる<br>障害グループはデータをミラー化したコピーを格納するために使用される<br> 

<table>
<tr><th>タイプ</th><th>ミラーレベル</th><th>障害グループ（FG：Fail Gropu）</th></tr>
<tr><td>外部（External）</td><td>非保護（ミラーなし・HW Raid の場合などに選択）</td><td>なし</td></tr>
<tr><td>通常（Normal）</td><td>2 方向（制御ファイルのみ 3 方向）</td><td>2 つ以上</td></tr>
<tr><td>高い（High）</td><td>3 方向</td><td>3 つ以上</td></tr>
</table>
</td></tr>
<tr><td>再バランス</td><td>ASM ディスクの追加や削除が行われると ASM ディスクに均等に AU が再配置される<br> 
</td></tr>
</table>

### ＞ ASM ディスクグループの作成

・ディスクパスは ASM として認識されている必要あり。V$ASM_DISK ビューの PATH 列で確認可能

・作成済みのディスクグループは、V$ASM_DISKGROUP で確認

・SQL 文で指定するディスクパスにワイルドカードを使用することも可能だが、候補ディスク以外が入ってるとエラーになる

| 構成内容 | SQL 文（ASM インスタンスで実行する） |
|---|---|
| 外部冗長性 | CREATE DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 EXTERNAL REDUNDANCY DISK 'ﾃﾞｨｽｸﾊﾟｽ' [,'ﾃﾞｨｽｸﾊﾟｽ'] |
| 通常(NORMAL)<br>または<br>高い(HIGH)冗長性 | CREATE DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 { NORMAL \| HIGH } REDUNDANCY<br>[ FILEGROUP 障害ｸﾞﾙｰﾌﾟ名 ] DISK 'ﾃﾞｨｽｸﾊﾟｽ' [,'ﾃﾞｨｽｸﾊﾟｽ']<br>[ FILEGROUP 障害ｸﾞﾙｰﾌﾟ名 ] DISK 'ﾃﾞｨｽｸﾊﾟｽ' [,'ﾃﾞｨｽｸﾊﾟｽ']<br>[ FILEGROUP 障害ｸﾞﾙｰﾌﾟ名 DISK 'ﾃﾞｨｽｸﾊﾟｽ' [,'ﾃﾞｨｽｸﾊﾟｽ'] ] ※HIGH の場合は FG を 3 つ以上作成する<br>※FALEGROUP 句を省略した場合、自動的にディスクグループの各ディスクを障害グループに追加する |

### ＞ ASM ディスクの管理

・ディスク名は、V$ASM_DISK の NAME 列で確認可能（デフォルトの名前は「ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名_連番」）

| 内容 | SQL 文 |
|---|---|
| マウント | ALTER DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 { MOUNT \| DISMOUNT [FORCE] } |
| ディスク追加 | ALTER DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 ADD DISK 'ﾃﾞｨｽｸﾊﾟｽ' [,'ﾃﾞｨｽｸﾊﾟｽ'] |
| 障害ｸﾞﾙｰﾌﾟ追加 | ALTER DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 ADD FAILGROUP 障害ｸﾞﾙｰﾌﾟ名 DISK 'ﾃﾞｨｽｸﾊﾟｽ' [,'ﾃﾞｨｽｸﾊﾟｽ'] |
| ディスクサイズ変更 | ALTER DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 RESIZE { ALL \| DISK ﾃﾞｨｽｸ名 } SIZE ｻｲｽﾞ[K\|M\|G] |
| ディスク削除 | ALTER DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 DROP DISK ﾃﾞｨｽｸ名 [,ﾃﾞｨｽｸ名] |
| ディスクグループ削除 | DROP DISKGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 INCLUDING CONTENTS<br>他の ASM インスタンスで当該ディスクグループがマウントしていると削除できない |

### ＞ ASM ディスクグループへの影響

・再バランス速度

デフォルトの再バランス速度は ASM_POWER_LIMIT 初期化パラメータで決定する

一時的にバランス速度を変更したい場合は、ALTER DISLGROUP ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名 REBALANCE POWER {0～11}

※再バランスが完了しているか確認するには V$ASM_OPERATION

・ASM ディスク検出時間

ASM ディスクとして検出されるディスクパスは、ASM_DISKSTRING 初期化パラメータで制限可能

ASM ディスク検出は以下の処理を行った時にも実行される

・ALTER DISKGROUP ADD DISK 文の実行

・ALTER DISLGROUP RESIZE DISK 文の実行

・V$ASM_DISKGROUP、V$ASM_DISK ビューの問い合わせ

正常に検出された ASM ディスクは V$ASM_DISK に表示される。HEADER_STATUS 列で用途を確認可能

| HEADER_STATUS | 説明 |
|---|---|
| FOREIGN | oracle オブジェクト（DB ファイル、投票ディスク、OCR）に使用されているディスク |
| MEMBER | ASM ディスクグループのメンバーとして初期化されている（使用済み） |
| CANDIDATE | 候補ディスク（ASM の DISKGROUP メンバーディスクとして追加可能なディスク） |

## ASM ファイル

### ＞ ASM ファイルの作成

・DB インスタンス側で ASM ファイルを作成するにはファイル名を指定する箇所で「+ﾃﾞｨｽｸｸﾞﾙｰﾌﾟ名」を指定する

・デフォルトでは OMF が適用されシステムが生成した階層構造上に ASM ファイルが作成される

（例）create tablespace TS01 datafile '**+DATA**' size 10M

### ＞ ASM で扱えないファイル

OCR ファイル、投票ディスク、バイナリファイル、トレースファイル、パスワードファイル

### ＞ ASM への移行

・CFS、RAW デバイスを使用したクラスタデータベースは ASM に移行することが可能

・RMAN で移行(制御ファイル、データファイル)と、SQL 文で再作成するファイル（一時ファイル、REDO ログ）がある

制御ファイルの移行

```
-- 期化パラメータの調整（制御ファイルを空にし、DB のデータファイルを ASM ディスクに指定）
SQL> ALTER SYSTEM SET CONTROL_FILES='' SCOPE=SPFILE;
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='+DATA' SCOPE=SPFILE;

-- DB 停止（すべてのインスタンス）
SQL> SHUTDOWN IMMEDIATE

-- 制御ファイルの移行（今まで使っていた制御ファイルからリストアを行う）
RMAN> STARTUP NOMOUNT
RMAN> RESTORE CONTROLEFILE FROM '/oradata/ct01.ctl';
```

制御ファイルの移行は NOMOUNT モードで行う。OMF を有効にし、CONTROL_FILES を空にしている為、OMF としてリストアされる

データファイルの移行

```
-- データベースのマウント
SQL> ALTER DATABASE MOUNT

-- イメージコピーを ASM ディスクグループに作成
RMAN> BACKUP AS COPY DATABASE FORMAT '+DATA';

-- データファイルを作成したイメージコピーへ切り替える
RMAN> SWITCH  DATABASE TO COPY;

-- 制御ファイルの為のリカバリ（DB 正常停止している場合は不要）
RMAN> RECOVER DATABASE;
```

データベースのバックアップは MOUNT モードで行う。ディスクグループにイメージコピーを取得し、SWITCH コマンドで切り替えを行えば、制御ファイルが認識しているデータファイル名が最新のイメージコピーのファイル名に切り替えられる

一時ファイルの移行

```
-- データベースをオープン
SQL> ALTER DATABASE OPEN

-- 一時表領域に一時ファイル追加（OMF を有効にしている為、ファイル名を指定してないが問題なく追加できる）
SQL> ALTER TABLESPACE TEMP ADD TEMPFILE;

-- 一時表領域からファイルを削除
SQL> ALTER DATABASE TEMPFILE '/oradata/temp01.dbf' DROP;
```

一時ファイルの再作成は OPEN モードで行う。OMF を有効にしている場合はファイル名を指定せずに一時ファイルを追加できる

REDO ログファイルを ASM へ移行

```
-- REDO ロググループの追加
SQL> ALTER DATABASE ADD LOGFILE INSTANCE 'ORCL' GROUP 5;
SQL> ALTER DATABASE ADD LOGFILE INSTANCE 'ORCL' GROUP 6;

-- ログスイッチとチェックポイント
SQL> ALTER SYSTEM SWITCH LOGFILE;
SQL> ALTER SYSTEM CHECKPOINT LOCAL;

-- 既存 REDO ロググループの削除
SQL> ALTER DATABASE DROP LOGFILE GROUP 1;
SQL> ALTER DATABASE DROP LOGFILE GROUP 2;
```

REDO ログファイルの再作成は OPEN モードで行う。OMF を有効にしている場合はファイル名を指定せず REDO ロググループを追加できる

## ■RAC データベースのバックアップリカバリ

### RAC 環境でのアーカイブログモード

select log_mode from v$database; または archive log list コマンドで現在のログモード確認

**＞ フラッシュリカバリ領域の構成**

・アーカイブログ、RMAN でバックアップしたファイル、フラッシュバックログを格納

・最大サイズ（DB_RECOVERY_FILE_DEST_SIZE）と保存場所（DB_RECOVERY_FILE）はインスタンスで全て同じ値にする

・保存場所に RAW デバイスは使用不可。NFS、CFS、ASM の何れかの場所を指定する

**＞ アーカイブログモードの構成**

・REDO ファイルのコピー（ローテーションのタイミングで作成）である為、各インスタンスのスレッドごとに生成される

・LOG_ARCHIVE_FORMAT の指定はログ順序 %s スレッド番号 %t RESETLOGS 識別子 %r の指定が必須

・保存場所は、LOG_ARCHIVE_DEST_n で n は 1～10 が対応し複数の保存場所を指定可能（ミラーされる）

・フラッシュリカバリ領域が構成されている場合、デフォルトで LOG_ARCHIVE_DEST_10 に
フラッシュリカバリ領域（DB_RECOVERY_FILE_DEST 初期化パラメータに指定されている場所）が設定される

・アーカイブログモードの変更は全てのインスタンスを停止し、何れかのインスタンスで ALTER 文を実行する

```
ローカルストレージに設定する場合（そのノードで作成されたアーカイブログファイルのみ保存される）
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/arch';

フラッシュリカバリ領域を使用する場合
ALTER SYSTEM SET LOG_ARCHIVE_DEST_10='LOCATION=USE_DB_RECOVERY_FILE_DEST';

アーカイブログモードを有効にする
-- すべてのインスタンスを停止し、有効にするインスタンスのみ MOUNT モードで起動する
$ srvctl stop database -d orcl
$ srvctl start inst -d orcl -i orcl1 -o mount

SQL> ALTER DATABASE ARCHIVELOG
-- DB を OPEN し、他インスタンスも起動する
SQL> ALTER DATABASE OPEN
$ srvctl start inst -d orac -i orcl2
```

### RAC 環境で RMAN 構成

・リポジトリ情報を制御ファイルまたはリポジトリカタログで保存している

・EM の「メンテナンス」-「リカバリ・カタログ設定」から保存先の変更が可能

**＞ スナップショット制御ファイル**

・リカバリカタログの再同期化と現行制御ファイルのバックアップ取得時にスナップショットが作成される

・デフォルトでは $ORACLE_HOME/dbs/snapcf_ｲﾝｽﾀﾝｽ名.f に作成される

・リカバリカタログ（RMAN リポジトリ情報を DB で保存）は Oracle データベースの表に作成される

・スナップショット制御ファイルの特徴

① 各ノードで同じディレクトリにする

② RAW デバイス、CFS、ASM に作成可能

③ バックアップを実行するノードで取得

```
RMAN に接続
$ rman target /
```

保存場所の確認と、保存場所の変更　※各ノードで同じパスでアクセスできるようにする必要がある

```
RMAN> SHOW SNAMPSHOT CONTROLFILE NAME;
RMAN> CONFIGURE SNAMPSHOT CONTROLFILE NAME TO '/dev/raw/raw10';
```

### ＞ 制御ファイルと SPFILE（サーバーパラメータファイル）の自動バックアップ

・リカバリカタログを利用してない場合（RMAN リポジトリの情報を制御ファイルに保存している場合）、
RMAN のバックアップを取得する時に制御ファイルと SPFILE を自動バックアップするに構成する

・自動バックアップするファイルは全てのノードから利用可能な場所に保存する

・フラッシュリカバリ領域でなくファイルシステムを利用する場合、ファイル作成場所のパラメータに %F が必要

現行設定確認

```
RMAN> SHOW CONTROLFILE AUTOBACKUP;              制御ファイル、SPFILE バックアップ有無
RMAN> SHOW CONTROLFILE AUTOBACKUP FORMAT;       バックアップの保存先（フラッシュリカバリ領域以外にしたい場合に指定）
```

制御ファイル、SPFILE バックアップ有効化

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;      BACKUP コマンドを発行後、制御ファイルと SPFILE もバックアップする
```

### ＞ 複数チャンネルの構成

・RMAN ではディスクまたはテープとアクセスを必要とするバックアップ、リストア、リカバリ時にチャネルが起動する

・各ノード毎にチャネルを張ることでバックアップ時間短縮が行える

現行設定確認

```
RMAN> SHOW DEVICE TYPE;                                  PARALLELISM の箇所がチャネル起動数
RMAN> SHOW CHANNEL;                                      各チャネルが使用する接続文字列確認
```

チャネル複数指定と接続文字列指定

```
RMAN> CONFIGURE DEVICE TYPE DISK PARALLELISM 2;          チャネルの数を 2 つにする
RMAN> CONFIGURE CHANNEL 1 DEVICE TYPE DISK CONNECT='sys/oracle@orac1';   10gR2 からは当該ｺﾏﾝﾄﾞはｵﾌﾟｼｮﾝとなり複数チャネ
RMAN> CONFIGURE CHANNEL 2 DEVICE TYPE DISK CONNECT='sys/oracle@orac2';   ル構成の場合、自動的に複数ノードに分散される
```

## RAC データベースのバックアップ

・非アーカイブログモードの場合、DB 停止した「オフラインバックアップ」が必要

・アーカイブログモードの場合、DB をオープンしたまま「オンラインバックアップ」が可能

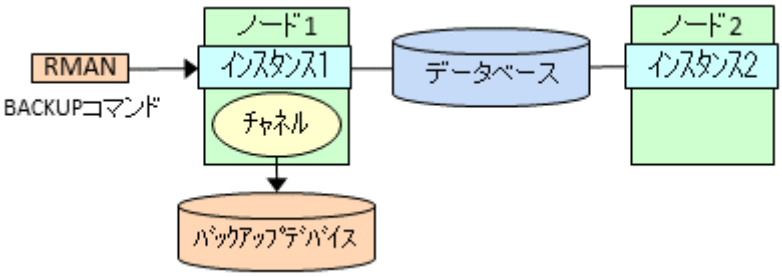
・RMAN によるバックアップはストレージを意識しない

・ユーザー管理のバックアップは、何れかのノードでバックアップモードへの変更が必要

### ＞ RMAN を使用したバックアップ

・バックアップファイルを格納するデバイスの配置がローカルか共有ストレージかによってチャンネル数は異なる

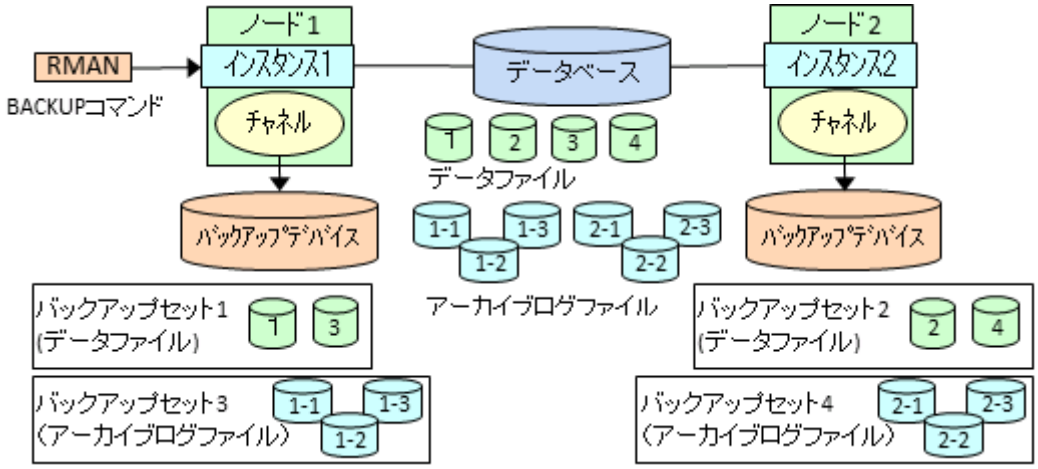
1 つのノードのみバックアップデバイスが接続されている場合




・ほかのノードからローカル接続されたバックアップデバイスにアクセスできるように構成するか、
バックアップデバイスが接続されたノードでバックアップ作業を完了できるようにする

・1 つのノードからすべての作業を完了させるにはアーカイブログを共有ストレージに配置していることが前提となる

### 複数のローカルデバイス




・それぞれのノードで独立したバックアップを作成する

・複数チャネルを使用してバックアップセットを作成するがバックアップセットに含まれるファイルは分散される

・複数チャネルで均等に分散されるが高速に対応できるノードに多くのファイルが配置される（ノードアンフィニティー）

・アーカイブログファイルはローカルストレージに配置されていてもバックアップすることが可能

### 共有ストレージ

・複数のローカルデバイスにバックアップを行う処理と同様に各ノードでチャネルを起動したパラレル処理が可能

フラッシュリカバリ領域にバックアップ取得

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE ALL INPUT
```

BACKUP DATABASE コマンドのオプション

PLUS ARHIVELOG　：ﾊﾞｯｸｱｯﾌﾟされてないｱｰｶｲﾌﾞﾛｸﾞもﾊﾞｯｸｱｯﾌﾟ

DELETE ALL INPUT：ﾊﾞｯｸｱｯﾌﾟ完了時にｱｰｶｲﾌﾞﾛｸﾞﾌｧｲﾙを削除する

### ＞ ユーザー管理のバックアップ

・OS コマンドを使用したバックアップは DB をバックアップモードに移行する

ALTER { DATABASE | TABLESPACE 表領域名 } { BEGIN | END } BACKUP;

BEGIN でバックアップモード開始、END でバックアップモード終了

・バックアップファイルの取得は共有ストレージタイプに依存

| タイプ | 説明 |
|---|---|
| CFS | ファイルをそのまま cp コマンドなどでバックアップすることが可能 |
| RAW | RAW デバイスにアクセスできる dd コマンドを使用する dd if=/dev/raw/raw1 of=/backup/ts1.bk bs=8k count=12801 |
| ASM | OS コマンドでのバックアップ不可。ASM ディスクに均等にファイル割付されている為、ディスク単位も意味がない<br>ファイル単位でバックアップする必要があり RMAN で行う |

### ＞ ストレージによるバックアップ

① バックアップモードに変更（ ALTER DATABASE BEGIN BACKUP ）

② ディスク操作の中断（ ALTER SYSTEM SUSPEND ）※クラスタ DB 全てに影響を及ぼす

バックアップモード中でもバックグラウンドプロセスによるディスク書き込みは行われるため、

当該コマンドを実施することでディスクを切り離し可能な状態にする

セッションは切断されないがディスク操作が必要な作業は全て一時停止状態になる

③ ディスクを切り離し、バックアップ

④ ディスク操作の再開（ ALTER SYSTEM RESUME ）

⑤ バックアップモードの終了（ ALTER DATABASE END BACKUP ）

## RAC データベースのリカバリ

・インスタンス障害は別のインスタンスでリカバリ可能

・ディスク障害（メディアリカバリ）はシングル環境と同様に手動でのリカバリが必要

### ＞ RAC 環境におけるリカバリ

・データファイル障害が発生した場合、アーカイブログモードであればリストアし、
アーカイブログファイルと REDO ログファイルを用いたリカバリを実行する

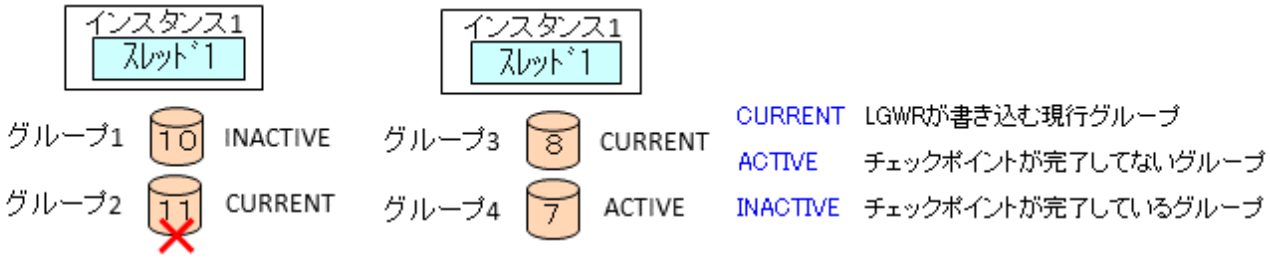
・メディアリカバリ時に各スレッドのアーカイブログファイルが必要

## データファイルのリカバリ

・SYSTEM 表領域や多くのデータファイルに障害が発生している場合は MOUNT モードでリカバリを行う
全てのインスタンスが OPEN されていないこと

・UNDO 表領域障害は、障害が発生した UNDO 表領域と対応させたいインスタンスのみ停止し、MOUNT モードでリカバリを行う
他のインスタンスは OPEN モードで良い

## REDO ログファイルのリカバリ




スレッド1＆ログ順序番号11までのリカバリ=SCN120までのリカバリ
→他のオンラインREDOログファイルも必要(内容がSCN120より前の情報が含まれているため)

| グループ | スレッド | ステータス | ログ順序 | 開始SCN | アーカイブ済 |
|---|---|---|---|---|---|
| 1 | 1 | INACTIVE | 10 | 105 | ○ |
| **2** | 1 | CURRENT | **11** | **120** | × |
| 3 | 2 | CURRENT | 8 | 115 | × |
| 4 | 2 | ACTIVE | 7 | 100 | ○ |

アーカイブ済の為、アーカイブログファイルを使用

アーカイブされていない為、オンラインREDOログファイルを直接使用

① 何れかのスレッドの CURRENT の REDO ロググループが全て破損すると DB が停止する

② 障害が発生した CURRENT グループのスレッド番号とログ順序番号までの不完全リカバリを実行する
DB を MOUNT モードで起動し、RECOVER [AUTOMATIC] DATABASE UNTIL SCN 120

③ ALTER DATABASE OPEN RESETLOGS で DB を OPEN する
※RESETLOGS：現行のオンライン REDO ログ・ファイル(または、破損が検出された場合は REDO 破損前の最後の REDO レコードまで)を
アーカイブし、オンライン REDO ログ・ファイルの内容を消去してオンライン REDO ログをログ順序 1 にリセット

## 制御ファイルのリカバリ

・全ての制御ファイルが破損した場合、バックアップ制御ファイルを使用するか、再作成を行う

・再作成は、CLUSTER_DATABASE 初期化パラメータを FALSE に設定したインスタンスのみを NOMOUNT モードで起動し
CREATE CONTROLFILE 文を実行する

### ＞ RMAN を使用したリカバリ

・必要なアーカイブログは RECOVER コマンド実行時に自動的にリストアも行われるが、

チャネル起動しているノードからバックアップファイルを読み込めるようにしておく必要がある

```
RMAN> SQL 'ALTER TABLESPACE USERS OFFLINE';          USERS 表領域を OFFLINE にする
RMAN> ALLOCATE CHANNEL c1 DEVICE TYPE DISK;          チャネルをローカルに割り当て
RMAN> SET ARCHIVELOG DESTINATION TO '/arck';         ローカルストレージにアーカイブログをリストア
RMAN> RESTORE TABLESPACE USERS;                      リストア対象を指定することで必要なデータファイルをバックアップフ
                                                     ァイルからリストアする
RMAN> RECOVER TABLESPACE USERS DELETE ARCHIVELOG;    メディアリカバリを実施する
RMAN> SQL 'ALTER TABLESPACE USERS ONLINE';           USERS 表領域を ONLINE にする
```

### ＞ ユーザー管理のリカバリ

・RECOVER コマンドの実行により必要なアーカイブログファイルがリクエストされるため、

リカバリ作業ノードから必要なアーカイブログファイルが読み込めるようにしておく

```
SQL> ALTER TABLESPACE USERS OFFLINE;                 USERS 表領域を OFFLINE にする
SQL> RECOVER TABLESPACE USERS;                       メディアリカバリを実施する
SQL> ALTER TABLESPACE USERS ONLINE;                  USERS 表領域を ONLINE にする
```

## ■RAC データベースの監視とチューニング

### RAC 固有のチューニング

基本はシングル環境と変わらない。加えてインターコネクトに使用するネットワークが問題になっている可能性がある

### ＞ キャッシュフュージョンと待機イベント

・キャッシュフュージョンは LMSn バックグラウンドプロセスによって処理され

「CURRENT または CR を要求する側」⇔「対象となるブロックを保持する側」⇔「リソースマスター」間で処理される

・グローバルキャッシュサービスはデータブロックのキャッシュ一貫性を提供する

・データブロックは DB バッファキャッシュ上で CURRENT（現行ブロック）と CR（読み取り一貫性ブロック）の状態がある

| 状態 | 説明 |
|---|---|
| CURRENT | 最近の変更すべてを含む最新コピー（DML 文は現行ブロックに対して行う） |
| CR | 現在のバッファより前のバージョンを含む特定時点（SCN）における CURRENT のコピー（読み取り専用） |

### ＞ ディスクからの読み込み（ GRANT 2-WAY ）

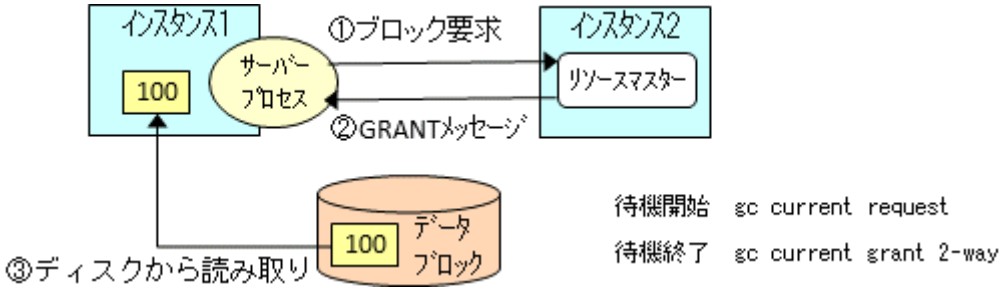
・要求したデータブロックをキャッシュしているインスタンスが存在しない場合、

リソースマスターから GRANT メッセージが送信される（メッセージ受信後、DISK からデータブロックを読み取る）

・同じインスタンスであれば即時実行、異なる場合（以下の図）はインターコネクトを使用し送受信する

・要求開始時に発生する待機イベント　gc { cr | current } [ multiblock ] request

・処理完了時に発生する待機イベント　gc { cr | current } grant 2-way



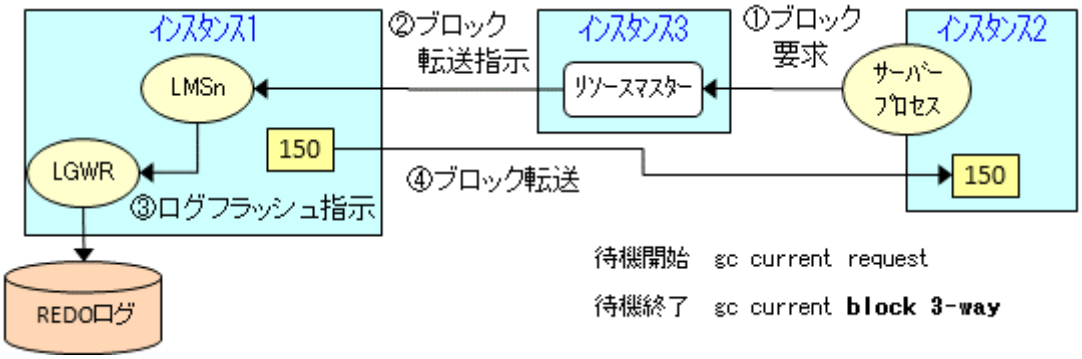
### ＞ データブロック転送（ BLOCK 2-WAY、BLOCK 3-WAY ）

・要求したデータブロックをキャッシュしているインスタンスが存在する場合、
　リソースマスターから保持側のインスタンスにブロック転送指示する

・保持側は変更内容が REDO ログに記録されていなければブロック転送前に LGWR にフラッシュ指示を行う

・リソースマスターが要求側、保持側どちらかにある場合、処理完了時に　gc { cr | current } block 2-way

・リソースマスターが要求側、保持側どちらにもない場合、処理完了時に　gc { cr | current } block 3-way




## RAC 固有の待機イベント

### 待機イベントを確認するビュー

| ビュー | 説明 |
|---|---|
| V$SYSTEM_WAIT_CLASS | インスタンスレベルの待機イベントクラス累積情報 |
| V$SYSTEM_EVENT | インスタンスレベルの待機イベント累積情報 |
| V$SESSION_WAIT_CLASS | セッションレベルの待機イベントクラス累積情報 |
| V$SESSION_EVENT | セッションレベルの待機イベント累積情報 |
| V$SESSION_WAIT | アクティブセッションの待機イベント |
| V$SESSION_WAIT_HISTORY | アクティブセッションの最新 10 個の待機イベント |
| V$ACTIVE_SESSION_HISTORY | アクティブセッションの 1 秒ごとにサンプリングされた待機イベント |

### グローバルキャッシュサービスの待機イベント

| 待機イベント | 説明 |
|---|---|
| gc buffer busy | ユーザーがバッファの確保解除を待機（バッファへのアクセス間隔がバッファ取得にかかる時間より短い）した回数<br>別のインスタンスで確保、保留されているブロックを受け取る場合に多く発生する |
| gc cr request | CR ブロック要求回数 |
| gc cr multi block request | CR ブロックを服う数ブロックで要求した回数 |
| gc cr block 2-way | 保持側または要求側がマスターノードの場合に、CR ブロック要求し即時にブロック取得が完了した回数 |
| gc cr block 3-way | 保持側・要求側以外がマスターノードの場合に、CR ブロック要求し即時にブロック取得が完了した回数 |
| gc cr block busy | CR ブロックの転送が即時に行われなかった回数。ログフラッシュによる遅延が原因となる場合が多い |
| gc cr block congested | CR ブロックの転送において内部キューが 1 マイクロ秒超える待機が発生した回数（CPU やメモリ不足など） |
| gc cr block lost | 要求した CR ブロックが失われたとみなされた回数。要求したブロックよりも先にチャネルメッセージが到着した場合などに発生する（インターコネクト障害や OS、ネットワーク構成などに起因） |
| gc cr grant 2-way | 全てのノードにブロックのキャッシュがない場合<br>CR ブロック要求後、即時にディスクからの読み取りが許可され、ブロック取得が完了した回数 |
| gc cr grant busy | CR ブロック要求後、即時にディスクからの読み取りが許可されなかった回数（先に処理すべき要求が存在する場合） |

| | |
|---|---|
| gc cr grant congested | CR ブロックの要求がマスターノードで許可されず、許可キューで 1 マイクロ秒を超える待機が発生した回数 |
| gc cr disk request | CR ブロックに対する許可メッセージを受け取り、ディスクからの読み取りを **要求** した回数 |
| gc cr disk read | CR ブロックに対する許可メッセージを受け取り、ディスクからの読み取りを **完了** した回数 |
| gc cr failure | CR ブロックを要求したが、障害ステータスが受信された回数（インターコネクトの負荷が高い場合） |

※ **cr** は全て **current**（現行ブロック）で読み替え可能

### グローバルエンキューサービスの待機イベント

| タイプ | 待機イベント | 説明 |
|---|---|---|
| TX | enq:TX-contention<br>enq:TX-index contention | トランザクションエンキュー<br>DML 操作時の対象レコードの排他制御、トランザクション範囲や追跡に使用 |
| TM | enq:TM-contention | 表やパーティションのエンキュー　DML 操作時の表定義保護に使用 |
| HW | enq:HW-contention | 最高水位標エンキュー　新しいブロック取得の制御に使用 |
| SQ | enq:SQ-contention | 順序エンキュー　順序番号の取得を直列化するために使用 |
| US | enq:US-contention | UNDO セグメントエンキュー　自動 UNDO 管理において UNDO セグメントの制御に使用 |
| TA | enq:TA-contention | トランザクションリカバリエンキュー<br>インスタンスリカバリ内のトランザクションリカバリに使用される |

個々の SQL で発生した Cluster 関連の待機イベントは以下のビューの CLUSTER_WAIT_TIME 列（マイクロ秒）で確認可能

| ビュー | 格納されているレコード |
|---|---|
| V$SQL | 子カーソル毎に 1 レコード |
| V$SQLAREA、V$SQLSTATS | 親カーソル毎に 1 レコード |

同じ SQL テキストを利用していても異なるスキーマにアクセスする場合もある

そのため、個々のハード解析による実行計画を「子カーソル」SQL テキストを「親カーソル」と呼ぶ

## RAC 固有のシステム統計情報

システム統計情報を取得するビューには、V$SYSSTAT（インスタンスレベル）、V$SESSTAT（セッションレベル）

V$MYSTAT（現行セッション）がある。V$STATNAME で統計名が確認できる

### グローバルキャッシュサービスのシステム統計

| システム統計 | 説明 |
|---|---|
| gc cr block build time | CR ブロックを作成するために要した時間 |
| gc **cr** block flush time | CR ブロック送信後のログフラッシュ合計時間<br>「gc cr block flush time」÷「gc cr blocks served」＝ CR ブロックあたりのログフラッシュ時間 |
| gc **cr** block receive time | 別インスタンスから CR ブロックを受け取るために要した合計時間<br>「gc cr block receive time」÷「gc cr blocks recived」＝ CR ブロックあたりの受信時間 |
| gc **cr** block send time | 別インスタンスに CR ブロックを送信するために要した合計時間<br>「gc cr block send time」÷「gc cr blocks served」＝ CR ブロックあたりの送信時間 |
| gc **cr** blocks received | 別インスタンスから受信した CR ブロック数 |
| gc **cr** blocks served | 別のインスタンスに送信した CR ブロック数 |
| gc current block pin time | 現行ブロックを確保していた時間 |
| gc blocks lost | ブロックが失われた為、再試行が必要とされたブロック数 |

| | |
|---|---|
| gc CPU used by this session | キャッシュのためにセッションで使用された CPU 量 |
| gcs messages sent | キャッシュのために送信されたメッセージ数 |

※太字 **cr** は **current**（現行ブロック）で読み替え可能

### グローバルエンキューサービスのシステム統計

| システム統計 | 説明 |
|---|---|
| global enqueue CPU used by this session | エンキューのためにセッションで使用された CPU 量 |
| global enqueue get time | 全ての同期、非同期のエンキュー取得の合計時間（10 ミリ秒） |
| global enqueue gets async | 非同期で取得されたエンキューの合計数 |
| global enqueue gets sync | 同期で取得されたエンキューの合計数 |
| global enqueue releases | 同期で取得されたエンキュー解除合計数 |

### ＞ パフォーマンス監視に使用されるビュー

| パフォーマンスビュー | 説明 |
|---|---|
| V$SEGMENT_STATISTICS | システム統計情報や、待機イベントをセグメントレベルで出力 |
| V$ENQUEUE_STATISTICS | 各ロック・タイプのエンキュー(ロック)要求数の統計情報 |
| V$INSTANCE_CACHE_TRANSFER | インスタンス間で転送されたブロッククラス別の統計情報 |

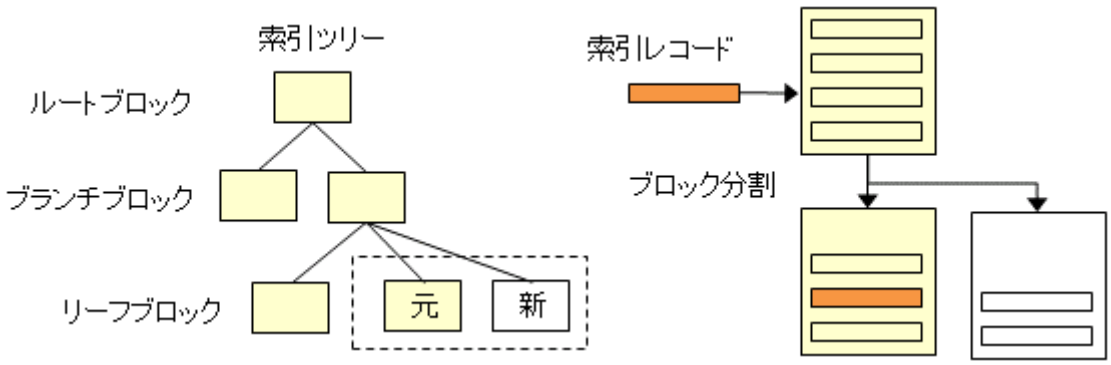
## RAC 環境のチューニング方法

基本的なチューニングはシングル環境と同じ

### ＞ 索引ブロックの競合

・「enq:TX - index contention」「gc buffer busy」「gc current block busy」「gc current split」待機イベントが多くなる

・索引ブロックが満杯の状態で、新しい索引エントリを格納する領域が不足すた場合、新しい索引ブロックを割り当てる
元のブロックの一部を新しいブロックに移動することで「ブロック分割」が行われる（パフォーマンス劣化）




・分割に伴うパフォーマンス劣化を最小限に抑えるためには 1 つの索引へのブロックアクセスが集中しないようにする

| 対策 | 説明 |
|---|---|
| ハッシュパーティション化したグローバル索引の使用 | グローバル索引をハッシュパーティション化する。ハッシュパーティションはハッシュ関数を使用して格納領域を決定する為、I/O 負荷を分散し競合を回避することが可能 |
| 逆キー索引の使用 | 反転させたデータ値を索引エントリの値として使用。順序を使用して増分する値を使用した例であっても常に右端のリーフブロックに格納されるとは限らない為、索引ブロックの競合を回避できる<br>ただし、検索時は等価検索（＝）しか使用できない |
| 順序を使用している場合は | 1 つのインスタンスで使用される順序番号を大きく確保しておくことにより､同じブロックに複数イン |

| | |
|---|---|
| 順序キャッシュを大きくする | スタンスからアクセス処理が不要になる |
| サロゲートキーではなく<br>ナチュラルキーを使用 | システム生成の一意キーではなく、自然発生のキーを使用することで（増分させていくだけの値と異なり）常に同じブロックに格納されるとは限らない為、索引ブロックの競合を防止できる |

### ＞ 順序の利用と索引ブロックの競合

・主キーや一意キーでは一意な値が必要とされることから、順序を使用することが多い（順序には索引が必要）

・RAC 環境では異なるインスタンスで同じ順序を使用しても異なる値が生成される

・索引はソートされた状態を維持する必要がある為、インスタンス間で頻繁に同じ索引ブロックが転送される

・頻繁な転送を防止するには順序のキャッシュサイズ（create sequence 文での chache 句）を大きくする（デフォルト 20）
デメリットとして障害などで共有プールからキャッシュが削除されると失う番号も大きくなる

| 順序 | 説明 |
|---|---|
| NOORDER | 各インスタンスでキャッシュされた順序値を独立して使用することができる（デフォルトで最もパフォーマンスが高い） |
| ORDER | インスタンス間で順序値を共有するが索引ブロックの転送が必要になる |

### ＞ 最高水位標の競合

・大量のデータ処理などで HWM を操作する HWM エンキューを取得する。HWM エンキュー競合時は次の待機イベントが発生する
「enq:HW - contention」「gc current grant busy」「gc current grant congested」

| 対策 | 説明 |
|---|---|
| ローカル管理表領域で均一な<br>大きなエクステントサイズを使用<br>（表領域の設定） | 新規エクステント獲得を減少される為、予め大きなエクステントにしておく<br>デフォルトでは AUTOALLOCATE 句（ディクショナリ管理）で最小 64KB の小さなエクステントで増分する<br>UNIFORM SIZE 句（ローカル管理）を使用し、最初から大きなエクステントを指定する<br>⇒UNIFORM SIZE 句で指定したサイズで常に拡張される |
| 自動セグメント領域管理（ASSM）<br>の検討<br>（データファイルの設定） | 更新を伴うデータブロックは ASSM を使用することで自動調整させることが可能<br>ASSM はデータブロックの使用率を調整、割り当てるデータブロックの分散のために空きリストを使用せず、各セグメント内に分散して格納するビットマップブロックを使用する |
| 空きリストを使用する場合（手動<br>セグメント領域管理）は空きリス<br>トグループに追加 | 各インスタンスが異なるブロックを使用するように空きリストグループの数をインスタンス数に合わせる。既存表の場合は、ALTER TABLE 文の STORAGE 句を用いて（FREELISTS n FREELIST GROUPS n）で変更する（デフォルトは 1、自動セグメント領域管理の場合、この値を自動的に調整する） |

### ＞ RAC 環境チューニングヒント

| 対策 | 説明 |
|---|---|
| フルスキャンの削減 | OLTP アプリケーションでは実行時間の長くかかるフルスキャンを避ける |
| パーティショニングによる<br>インスタンス間通信の削減 | 各インスタンスで異なるパーティションを利用することでインスタンス間の通信を削減可能<br>ハッシュパーティションを利用することで DB キャッシュのアクセスパターンを分析させバッファ競合（buffer busy waits）を減少させることが可能 |
| SQL 解析処理の削減 | 共有プール内のライブラリキャッシュとディクショナリキャッシュはインスタンス間で調整される<br>過度な解析処理はインターコネクトの通信量が増加する。PL/SQL やアドバンスドキューイングを使用している場合、パッケージやプロシージャの再コンパイル及びに排他モードでﾗｲﾌﾞﾗﾘｷｬｯｼｭﾛｯｸが取得される |
| SGA サイズの調整 | グローバルリソースの情報は共有プールに格納される<br>キャッシュフュージョンに関するパストイメージは DB バッファキャッシュに残される |

| | |
|---|---|
| | 上記のことからシングル環境に比べ共有プールは 15%、DB バッファキャッシュは 10%多く必要 ※<br>※全てのインスタンスで確保した SGA サイズの合計 |
| ロック処理の削減 | グローバルエンキューサービスにより、トランザクションロックもインスタンス間で調整される<br>順序を指定せず採番テーブルを使用した場合、位置否番号生成ではロック取得が必要なため、RAC 環境ではインスタンス間でロックの競合が発生する可能性がある |
| 不必要な索引の削除 | インスタンス間で索引ブロックも転送されることからインターコネクトの通信量増加の原因になる |
| トランザクションのコミット | コミットされてないトランザクションが存在する場合、キャッシュフュージョンによるブロック転送では UNDO データの送信も必要になる |
| インスタンスリカバリ時間 | RECOVERY_PARALLELISM 初期化パラメータにリカバリで使用するパラレル実行サーバープロセス数を設定<br>リカバリで実施される 1st pass log read は非同期 I/O が使用できる為、非同期 I/O 構成を検討する |

## RAC 環境のスナップショット

### > Statspack と自動ワークロードリポジトリ（AWR）

・各インスタンスのメモリ内統計を、それぞれ異なるスナップショットとして DB に保存する

・Statspack をインストールした後で、各インスタンスで Statspack スナップショットを作成する必要がある

・AWR の場合、何れかのインスタンスでスナップショットを作成すると
　全てのインスタンスで同時に AWR スナップショットが作成され、同じスナップショット ID が割り当てられる

RAC 固有の情報

| 情報 | 説明 | | | |
|---|---|---|---|---|
| Global Cache Load Profile | グローバルリソースに関する負荷情報。送受信したブロックとメッセージ数 | | | |
| Global Cache Efficiency Percentages | データブロックの取得方法（ディスク、ローカルキャッシュ、リモートキャッシュ） | | | |
| Global Cache and Enqueue Services<br>- Workload Characteristics | グローバルリソースの取得に必要とされた平均時間（下記の範囲に含まれることを推奨） | | | |
| | 統計名 | 下限 | 標準 | 上限 |
| | CR ブロックのリクエスト処理時間（ミリ秒）<br>Avg global cache cr block XX time　　XX build、send、flush の合計 | 0.1 | 1 | 10 |
| | CR ブロックの受信時間（ミリ秒）<br>Avg global cache cr block receive time | 0.3 | 4 | 12 |
| | 現行ブロックのリクエスト処理時間（ミリ秒）<br>Avg global cache current block XX time　　XX pin、send、flush の合計 | 0.1 | 3 | 23 |
| | 現行ブロックの受信時間（ミリ秒）<br>Avg global cache current block flush time | 0.3 | 8 | 30 |
| Global Cache and Enqueue Services<br>- Messaging Statistics | グローバルリソースのメッセージ統計 | | | |
| Segment Statistics | セグメント情報（V$SEGMENT_STATISTICS） | | | |
| Global Enqueue Statistics | グローバルエンキュー統計情報（V$GES_STATISTICS） | | | |
| Global CR Served Stats | CR ブロック統計情報（V$CR_BLOCK_SERVER） | | | |
| Global CURRENT Served Stats | 現行ブロック統計情報（V$CURRENT_BLOCK_SERVER） | | | |
| Global Cache Transfer Stats | インスタンス間で転送されたキャッシュブロック統計情報（V$_INSTANCE_CACHE_TRANSFER） | | | |

### > Automatic Database Diagnostic Monitor (ADDM)

・ADDM の DB 分析は AWR スナップショット作成時に実行される

RAC 固有の ADDM 検出結果

| 検出タイプ | 説明 |
|---|---|
| ホットオブジェクト | インスタンスとクラスタ全体で読み書きの競合が多いホットオブジェクト（セグメント） |
| ホットブロック | インスタンスとクラスタ全体で読み書きの競合が多いホットブロック詳細情報 |
| インスタンス競合 | 複数のインスタンスから実行されている更新によるブロック、エンキューの競合 |
| 上位 SQL | インスタンス間メッセージ交換が多く発生している SQL |
| LMS の輻輳 | LMSn プロセスでリクエスト処理に対応できないほど負荷が高い状況 |
| インターコネクト待機時間 | インターコネクトが問題となる待機時間 |

### > Enterprise Manager の[パフォーマンス]タブ

クラスタデータベースのパフォーマンスタブ

| パフォーマンス | 説明 |
|---|---|
| クラスタホストのロード平均 | 使用可能なホストで実行されたプロセス数の最小値、最大値、平均値を表示 |
| グローバルキャッシュブロックのアクセス待機時間 | 現行ブロックと CR ブロックの待機時間を表示 |

クラスタキャッシュー貫性ページ

| パフォーマンス | 説明 |
|---|---|
| グローバルキャッシュブロック転送率 | クラスタ内の全インスタンスがインターコネクト経由で受信した現行ブロックと CR ブロックの数 |
| ｸﾞﾛｰﾊﾞﾙｷｬｯｼｭﾌﾞﾛｯｸ転送と物理読み取り | ディスクから読み取った物理読み取りと、リモートインスタンスからのブロック転送の割合 |

## ■サービス

**RAC 環境におけるサービス**

・サービス名毎にインスタンスをグループ化することができる

・クラスタを構成するリソースを効率的に使用することができる

### > グリッドコンピューティング

・ハードウェアリソースをプール化し、必要な時に必要なリソースをワークロードに割り当てる

### > サービスの特徴

| 特徴 | 説明 |
|---|---|
| 高可用性 | クライアントがサービス接続するのであれば、サービスを構成する何れかのインスタンスに接続することになり可用性が向上する |
| パフォーマンス | 異なるサービスを異なるインスタンスで動作させれば、インスタンス間で発生するキャッシュフュージョンの使用が減少しブロック転送を最小限にすることが可能 |

### > 優先インスタンスと使用可能インスタンス

・サービスを提供するインスタンスには「優先インスタンス」と「使用可能インスタンス」の役割を与える

・サービス構成は CRSD によって監視されている（障害発生時に使用可能インスタンスに F/O する）

| インスタンス | 説明 |
|---|---|
| 優先 | サービスを提供するインスタンス（接続を受け付けるインスタンス） |
| 使用可能 | 優先インスタンスが停止したときにサービスを開始するインスタンス |

### › サービス属性

| 属性（シングルと共通） | 説明 |
|---|---|
| サービス名 | サービスを識別するための一意な名前。サービス名はリスナー登録され、Oracle Net ｸﾗｲｱﾝﾄ構内でも使用される |
| ネットサービス名 | Oracle Net クライアントがサービスに接続するために使用する |
| しきい値 | サービスメトリック（ﾚｽﾎﾟﾝｽ時間と CPU 使用時間）にしきい値を設定することで、超過時にアラート発生させる |
| 優先順位 | リソースマネージャのコンシューマグループとサービス名をマッピングできる。コンシューマグループに CPU 使用率を指定すると、インスタンス内の複数サービスに対して優先順位を設定したことになる |
| **属性（RAC 固有）** | **説明** |
| 高可用性構成 | 優先インスタンスと使用可能インスタンスによる複数インスタンスの構成 |
| フェイルオーバー特性 | 透過的アプリケーションフェイルオーバー（TAF）使用時のフェイルオーバー特性 |
| ロードバランシング方法 | サービスあたりのセッション数や、ロードバランシングアドバイザを使用したロードバランシング特性 |
| 分散ﾄﾗﾝｻﾞｸｼｮﾝﾌﾗｸﾞ | 有効にすると分散トランザクションは同じインスタンスを使用するように構成できる |
| ｱﾄﾞﾊﾞﾝｽﾄｷｭｰｲﾝｸﾞ特性 | 有効にすると OCI や ODP.NET クライアントに対して、RAC の高可用性イベントを送信することができる |

### › サービスタイプ

| 種類 | 説明 |
|---|---|
| アプリケーションサービス | 独自に作成したサービス。ワークロード単位、ワークロード内の機能別にサービス作成が可能<br>事前接続構成の TAF を使用した場合に作成されるシャドウサービスやアドバンストキューイングのキューごとに作成されるサービスも含まれる |
| 内部サービス | システムで生成されたサービス（以下の 2 つ）で変更削除はできない<br>SYS$BACKGROUND（バックグラウンドプロセスの接続に使用）<br>SYS$YSERS（アプリケーションサービスに関連付けられていないユーザーセッションで使用） |

作成済サービスは「DBA_SERVICES」セッションが使用しているサービスは「V$SESSION」で確認可能

## サービスの作成

・用途によって接続するサーバーを割り振りたい時、サービス名毎に振り分けることが可能になる

・CRSD によって管理される

・クラスタリソース登録と接続に使用する Oracle Net サービスの構成が必要

・DBCA、EM、SRVCTL で作成可能

| 登録内容 | DBCA | EM | SRVCTL |
|---|---|---|---|
| クラスタリソース登録 | ○ | ○ | ○ |
| tnasnames.ora（サーバー側）の修正 | ○ | × | × |
| サービス作成後の初期起動 | ○ | ○ | × |

・何れのツールを使用した場合でもクラスタリソースは作成される（crs_stat -t で確認）

リソース名の末尾の cs はサービス管理に使用するリソース、srv は優先インスタンスで指定したインスタンス毎に作成される（F/O すると crs_stat -t の一番右のホスト列のホスト名が変化する）

### > DBCA

操作のステップで「サービス管理」を選択することで、サービスの作成・削除する画面へ遷移できる

CRM というサービスを作成し、2 つのインスタンスを「優先インスタンス」とした例


### > Enterprise Manager

[クラスタデータベース]の[メンテナンス]タブに含まれる[クラスタ管理データベースサービス]から作成する

### > SRVCTL

| 操作 | 説明 |
|---|---|
| 作成 | srvctl add **service** -d DB 名 -s ｻｰﾋﾞｽ名 -r 優先ｲﾝｽﾀﾝｽﾘｽﾄ -a 使用可能ｲﾝｽﾀﾝｽﾘｽﾄ |
| 削除 | srvctl remove **service** -d DB 名 -s ｻｰﾋﾞｽ名 [-i ｲﾝｽﾀﾝｽ名] |
| 確認 | srvctl config **service** -d DB 名<br>PREF:の後に優先インスタンス、AVAIL の後に使用可能インスタンスが表示される |

### > サービスを使用するネットサービスエントリ

・DBCA を使用した場合、サーバー側の tnsnames.ora にネットサービスエントリが作成されるが

クライアント側の構成は各自追記が必要

```
CRM =                                                    作成したサービス名を指定
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = orac1v.com)(PORT = 1521))    優先インスタンスと使用可能インスタンスを全て列挙する
    (ADDRESS = (PROTOCOL = TCP)(HOST = orac2v.com)(PORT = 1521))
    (LOAD_BALANCE = yes)
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CRM)                               サービス名を記述(DB_DOMAIN 初期化パラメータが設定されている場合は、ドメイン名を含める 。CRM.orcl.com とか)
    )
  )
```

## サービスの管理

・永続的（サービス有効/無効）、一時的な変更が可能

・サービスを起動すると優先インスタンスの SERVICE_NAMES 初期化パラメータにサービス名が追加される

srvctl ｺﾏﾝﾄﾞ **service** -d DB 名 -s ｻｰﾋﾞｽ名 オプション

| コマンド | 内容 | コマンド例 |
|---|---|---|
| add | 作成 | srvctl add service -d orcl -s CRM -r orac1,orac2 -a orac3 |
| | ｲﾝｽﾀﾝｽﾘｽﾄ追加 | srvctl add service -d orcl -s CRM -u { -r orac1 \| -a orac2 } |
| remove | 削除 | srvctl remove service -d orcl -s CRM [ -i orac2 ]<br>最後のサービスは削除不可。また削除するにはサービスを事前に停止しておく必要がある |

| | | |
|---|---|---|
| config | 定義確認 | srvctl config service -d orcl -s CRM<br>PREF:の後に優先インスタンス、AVAIL の後に使用可能インスタンスが表示される |
| status | 起動状態確認 | srvctl status service -d orcl |
| start | 起動 | srvctl start service -d orcl [ -s CRM -i orcl1,orcl2 ] |
| stop | 停止 | srvctl stop service -d orcl [ -s CRM -i orcl1,orcl2 ] |
| enable | 有効化 | srvctl enable service -d orcl [ -s CRM -i orcl1,orcl2 ] |
| disable | 無効化 | srvctl disable service -d orcl [ -s CRM -i orcl1,orcl2 ] |
| relocate | サービス<br>再配置 | srvctl relocate service -d orcl -s CRM -i orcl1 -t orcl2 [-f]<br>未インスタンスには配置不可、優先インスタンスが停止すると使用可能インスタンスでサービス提供する |
| modify | 定義の変更 | サービス（サービスの配置状況を永続的に変更する）次回再起動時に定義した状態でサービス開始する<br>サービス提供ｲﾝｽﾀﾝｽ変更 srvctl modify service -d orcl -s CRM -i orcl1 -t orac2<br>使用可能ｲﾝｽﾀﾝｽを優先ｲﾝｽﾀﾝｽに変更 srvctl modify service -d orcl -s CRM -i orcl1 -r<br>サービス構成の再定義 srvctl modify service -d orcl -s CRM -n -i orcl1 -a orac2 |

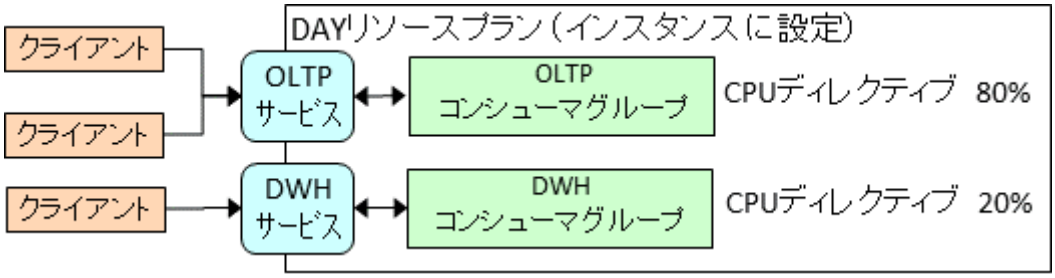
| オプション | 説明 |
|---|---|
| -s | サービス名を指定（サービス名を指定しない場合は DB で作成されている全てのサービスが対象になる） |
| -r | 優先インスタンス（複数指定する場合はカンマ区切り）modify コマンド時に優先インスタンスへ変更する場合にも使用 |
| -a | 使用可能インスタンス（複数指定する場合はカンマ区切り） |
| -i | インスタンス名（複数指定する場合はカンマ区切り） |
| -t | 移動先インスタンス名（relocate/modify コマンド時に使用する） |
| -n | modify コマンドで使用時、サービス配置を最初から再定義する場合に使用 |
| -u | インスタンスリストにインスタンスを追加する |
| -f | サービス操作の停止または再配置中のすべてのセッションの切断 |

## サービスと Oracle 機能の連携

### ＞ リソースマネージャでのサービス使用

・リソースマネージャはクライアントセッションをコンシューマグループとしてグループ化し、
　インスタンスに設定したリソースプランに紐づけてリソース制限を設定する機能

・コンシューマグループとサービスの紐づけが可能




| 用語 | 説明 |
|---|---|
| コンシューマグループ | スキーマをまとめたグループ（スキーマを登録しなくても良い） |
| リソースプラン | コンシューマグループ毎に CPU 使用率等を割り振ったプラン<br>上記図の例だと OLTP コンシューマグループに CPU80%、DWH コンシューマグループに CPU20%を割り振っている |
| コンシューマグループ<br>マッピング | 特定のユーザーや、特定のサービスをマッピングすることで、自動的にコンシューマグループに割り振る<br>上記図の例だと OLTP サービス⇒OLTP コンシューマグループにマッピングされており、<br>OLTP サービスで接続したセッションは自動的に OLTP コンシューマグループに割り振られる |

## リソースマネージャの設定

・EMの[クラスタデータベース] - [管理]タブの[コンシューマグループマッピング]ページ内［サービスマップ］項目

DBMS_RESORCE_MANAGER.SET_CONSUMER_GROUP_MAPPING プロシージャでも設定可能

・ユーザーに対するスイッチ権限を付与する必要がある（DBMS_RESORCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP）

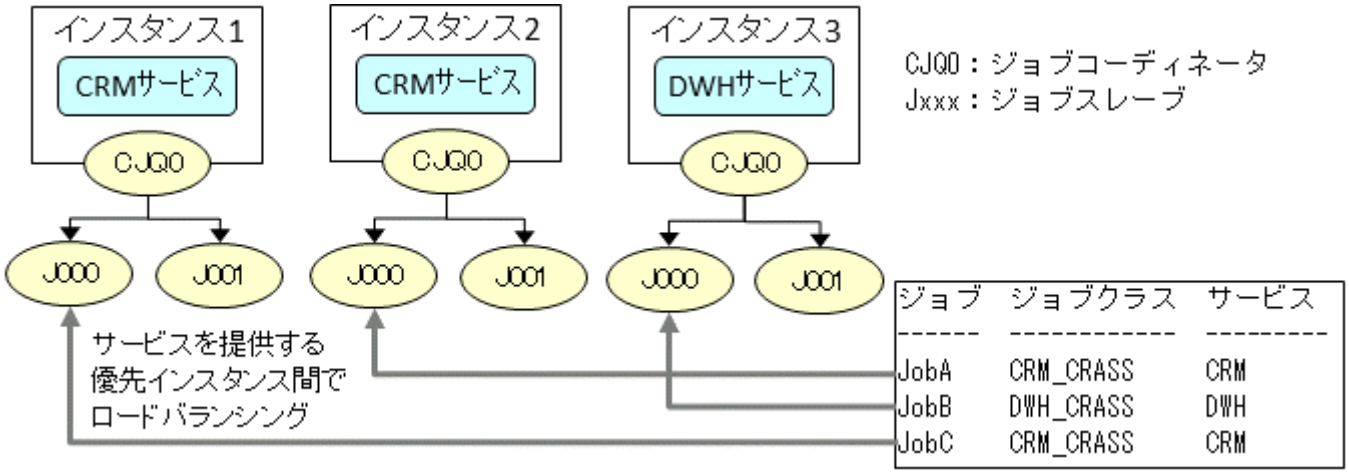
サービスに対するリソースコンシューマグループマッピングでは、スイッチ権限が付与されたセッションのみサービスに対応づけられたコンシューマグループへ切り替えることができる

接続ユーザーが特定できない場合はPUBLICにスイッチ権限を付与する

| ディクショナリ | 説明 |
|---|---|
| DBA_RSRC_GROUP_MAPPINGS | リソースマネージャと紐づけられているコンシューマグループ確認 |
| DBA_RSRC_CONSUMER_GROUP_PRIVS | すべてのリソース・コンシューマ・グループおよび<br>リソース・コンシューマ・グループに割り当てられたユーザーおよびロールに関する情報 |

### ＞ スケジューラでのサービス使用

| 用語 | 説明 |
|---|---|
| スケジュール | 実行するタイミングのパターン |
| プログラム | SQLやPL/SQLなど実行するコマンド |
| ジョブ | 所属するジョブクラス、実行するスケジュール、実行するプログラムを登録 |
| ジョブクラス | グループ化されたジョブで、サービスや、コンシューマグループと関連付けが行える<br>サービスと関連付けることで可用性とパフォーマンス向上が見込める<br>高可用性：優先インスタンスが停止しても使用可能インスタンスがサービスを提供する<br>パフォーマンスの向上：1つのジョブで一番負荷の低いインスタンスで実行される（デフォルトの動作） |




## ジョブクラスの作成

・EMの[クラスタデータベース] - [管理]タブにある[ジョブクラス]ページから[ジョブクラスの作成]

DBMS_SCHEDULER.CREATE_JOB_CLASS プロシージャを使用し作成

既存のジョブクラスをサービスと対応付ける場合は、DBMS_SCHEDULER.SET_ATTRIBUTE を使用

## ジョブの作成

・EMの[クラスタデータベース] -[管理]タブにある[ジョブ]ページ

ジョブクラスをジョブに対応付け実行するコマンドを定義する

・DBMS_SCHEDULER.CREATE_JOB プロシージャでジョブ作成、DBMS_SCHEDULER.SET_ATTRIBUTE で

インスタンスを振り分ける attribute => 'INSTANCE_STICKINESS' を設定する

### ＞ サービスに対するメトリックしきい値の設定

・V$SERVICEMETRIC（直近 60 秒）、V$SERVICE_METRIC_HISTORY（過去 1 時間）でメトリック値の結果確認

| カラム | 説明 |
|---|---|
| INTSIZE_CSEC | 何秒毎の計算結果か判断。5 秒または 60 秒 |
| ELAPSEDPERCALL | １コール辺りのレスポンス時間（マイクロ秒）（閾値の設定が可能） |
| CPUPERCALL | １コールあたりの CPU 使用時間（マイクロ秒）（閾値の設定が可能） |

| メトリックしきい値 | 説明 |
|---|---|
| ELAPSED_TIME_PER_CALL | 1 処理あたりのレスポンス時間に対するしきい値（経過時間として許容できる最大値を設定する） |
| CPU_TIME_PER_CALL | 1 処理あたりの CPU 時間に対するしきい値（待機時間を除く実際に処理に割り当てられた CPU 時間） |

#### メトリックしきい値設定

・EM の[データベースインスタンス] - [メトリックとポリシー設定]

・DBMS_SERVER_ALERT.SET_THRESHOLD プロシージャ

| パラメータ | 説明 |
|---|---|
| WARNING_VALUE | 警告アラートが発生するしきい値 |
| CRITICAL_VALUE | クリティカルアラートが発生するしきい値 |
| OBSERVATION_PERIOD | 観測期間 |
| CONSECUTIVE_OCCURRENCES | 発生回数（観測期間中にここで指定した回数のしきい値超過が発生したら値に応じたアラートが発生） |

### ＞ サービス集計とトレースの構成

・サービス単位（サービス名 - モジュール名 - アクション名）でシステム統計の収集、SQL トレースの収集が可能

・サービス名はログインする時に使用したサービス名が自動的に認識される

・モジュール名とアクション名は DBMS_APPLICATION_INFO パッケージや OCI コールで設定できる

・モジュール名には実行中のプログラムなどを設定する

・アクション名には特定の処理（データ登録）などを設定する

#### サービスパフォーマンスに関するビュー

| 区分 | ビュー | 説明 |
|---|---|---|
| システム統計 | V$SERVICES_STATS | サービス単位の集計結果 |
| | V$SERV_MOD_ACT_STATS | サービス・モジュール・アクション単位の集計結果 |
| メトリック | V$SERVICEMETRIC | サービスメトリックの結果（直近 60 秒） |
| | V$SERVICEMETRIC_HISTORY | サービスメトリックの結果（過去 1 時間） |
| 待機イベント | V$SERVICE_WAIT_CLASS | サービス単位の待機イベントクラス |
| | V$SERVICE_EVENT | サービス単位の待機イベント |
| 設定確認 | DBA_ENABLED_AGGREGATIONS | システム統計の集計設定確認 |
| | DBA_ENABLED_TRACES | SQL トレースの設定確認 |

#### システム統計の集計

【有効】DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE('ｻｰﾋﾞｽ名', 'ﾓｼﾞｭｰﾙ名' [,'ｱｸｼｮﾝ名'])

【無効】DBMS_MONITOR.SERV_MOD_ACT_STAT_DISABLE('ｻｰﾋﾞｽ名', 'ﾓｼﾞｭｰﾙ名' [,'ｱｸｼｮﾝ名'])

## SQL トレース

・USER_DUMP_DEST 初期化パラメータで指定した場所にトレースファイルが出力される

```
DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE(              トレースを有効にする
 service_name     IN VARCHAR2,                       サービス名
 module_name      IN VARCHAR2 DEFAULT ANY_MODULE,    モジュール名
 action_name      IN VARCHAR2 DEFAULT ANY_ACTION,    アクション名
 waits            IN BOOLEAN DEFAULT TRUE,           待機イベントをトレースに含める
 binds            IN BOOLEAN DEFAULT FALSE,          バインド変数の値をトレースに含める
 instance_name    IN VARCHAR2 DEFAULT NULL);         トレース取得を行うインスタンスを指定

DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE(             トレースを無効にする
   service_name     IN VARCHAR2,                     サービス名
   module_name      IN VARCHAR2,                     モジュール名
   action_name      IN VARCHAR2 DEFAULT ALL_ACTIONS, アクション名
   instance_name    IN VARCHAR2 DEFAULT NULL);       トレース取得を行うインスタンスを指定
```

・DB 単位（DBMS_MONITOR.DATABASE_TRACE_ENABLE）や、
SID 単位（DBMS_MONITOR.SESSION_TRACE_ENABLE）の取得も可能
※参考 http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc_cd/appdev.102/B19245-02/d_monitor.htm

### ＞ パラレル実行とサービス

・サービスを利用したパラレル処理ではサービスを提供する優先インスタンス以外のインスタンスでも
パラレル実行サーバーが動作する可能性がある

### ＞ セッション制限とサービス

・ALTER SYSTEM ENABLE RESTRICTED SESSION などによるセッション制限はサービスと連携している
・インスタンスでセッション制限が行われると優先インスタンスとしてサービス提供できなくなる
・別のインスタンスが使用可能であればサービスの F/O が実施される
・セッション制限を行ったインスタンスへはローカル接続のみ可能

## ■接続の高可用性

### 接続時ロードバランシングの構成

#### › クライアント側での設定

| パラメータ | 説明 |
|---|---|
| なし | ADDLESS_LIST に記述されている先頭アドレスのみ使用する |
| FAILOVER = ON | フェイルオーバー　先頭のアドレスから順次接続を試みる<br>ホストに VIP を指定することで、停止時に別ノードへ F/O するため、<br>該当 VIP に接続する再、TCP/IP タイムアウトを待つことなく即時に次のリスナー接続へ切り替わる |
| LOAD_BALANCE = ON | ロードバランシング　ランダムに接続を試みる（接続数が少ないとランダムになるとは限らない） |

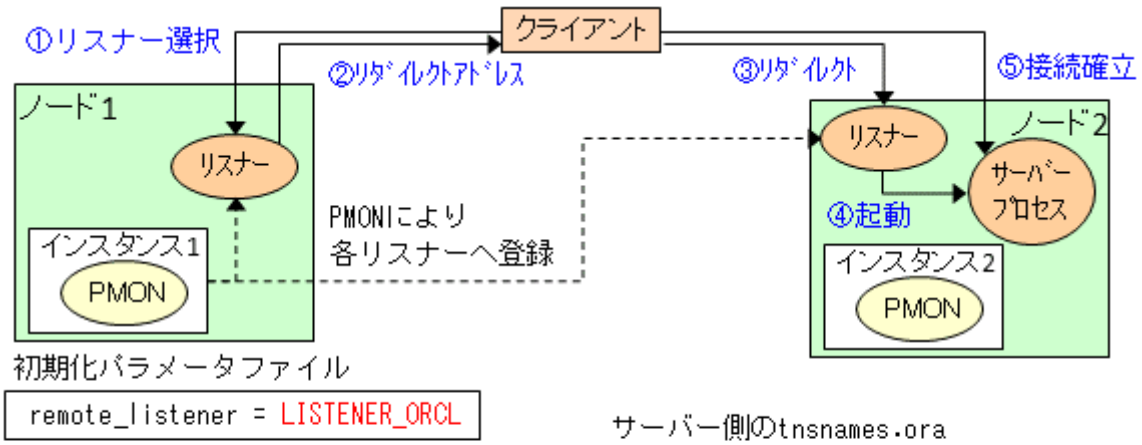
```
CRM =                                                    接続識別名（sqlplus で@の後の文字列）
  (DESCRIPTION =
    (ADDRESS_LIST =
      (パラメータ)                                        パラメータを指定
      (ADDRESS = (PROTOCOL = TCP)(HOST = orac1v.com)(PORT = 1521))   ホストは VIP を指定
      (ADDRESS = (PROTOCOL = TCP)(HOST = orac2v.com)(PORT = 1521))   ホストは VIP を指定
      (CONNECT_DATA = (SERVICE_NAME = CRM)
    )
  )
```

・パラメータで F/O の場合、サーバー側の listener.ora に他ノードから移動してきた VIP をリスニングしない設定を追加する

```
LISTENERS_ORCL =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = orac1v.com)(PORT = 1521)      IP=FIRST を指定することで、
      (IP=FIRST))                                                    ADDRESS で指定した IP アドレスのみ
    (ADDRESS = (PROTOCOL = TCP)(HOST = orac2v.com)(PORT = 1521)      リスナーを対応付ける
      (IP=FIRST))
  )
```

#### › サーバー側の接続時ロードバランシング

・接続要求を受けたリスナーが同じノードのインスタンスだけでなく別ノードのリスナーにリダイレクトする




初期化パラメータファイル

```
remote_listener = LISTENER_ORCL
```

サーバー側のtnsnames.ora

```
LISTENERS_ORCL =                                  PMONが登録するリスナー
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP) (HOST = sv1-vip) (PROT = 1521))
    (ADDRESS = (PROTOCOL = TCP) (HOST = sv2-vip) (PROT = 1521))
  )
```

・インスタンスが起動すると、PMON により LOCAL_LISTENER、REMOTE_LISTENER の各初期化パラメータに指定した
リスナーにインスタンス情報が登録される

・各インスタンから登録先のリスナーアドレスを認識できるように REMOTE_LISTENR で指定したネットサービス名を tnsnames.ora にネットサービス名と対応する接続記述子で登録先リスナーアドレスを指定する

・クライアントからの接続は最適なパフォーマンスが実現されているノードのリスナーにリダイレクトされる

### ＞ 接続ロードバランシングとロードバランシングアドバイザ

・サービスに対する接続は「接続ロードバランシング（Connection Load Balancing：CLB)」が実行される

次の 3 つのロードバランシング目標の設定ができる

| パターン | clb_goal | goal | 備考 |
|---|---|---|---|
| セッション数 | CLB_GOAL_LONG | - | セッション数が均等になるように分散（デフォルト） |
| CPU 実行キュー | CLB_GOAL_SHORT | GOAL_NONE | 実行キューサイズを調べもっとも負荷の低いノードへ割り振る |
| サービス適合度※ | | GOAL_SERVICE_TIME | サービス（レスポンス）時間を目標とする |
| | | GOAL_THROUGHPUT | スループットを目標とする |

※ ロードバランシングアドバイザ（Load Balancinfg Adviser：LBA）によるサービス品質

・DBMS_SERVICE パッケージによる設定

```
-- DBMS_SERVICE パッケージによる設定
DBMS_SERVICE.MODIFY_SERVICE (
  service_name => 'ｻｰﾋﾞｽ名',
  clb_goal     => 接続ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ目標,
  goal         => ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞｱﾄﾞﾊﾞｲｻﾞ
);
```

```
-- サービス時間を目標とした LBA を設定例
DBMS_SERVICE.MODIFY_SERVICE (
  service_name => 'CRM',
  clb_goal     => DBMS_SERVICE.CLB_GOAL_SHORT,
  goal         => DBMS_SERVICE.GOAL_SERVICE_TIME
);
```

・EM による設定は[クラスタデータベース] - [メンテナンス] - [クラスタ管理データベースサービス] - [サービス作成]

・設定状況は、DBA_SERVICES（clb_goal,goal 列）で確認する

・LBA によるインスタンスへの割り当て比率の変化は SYS$SERVICE_METRICS_TAB で確認

user_data 列にインスタンス毎に percent=XX が表示され、XX が割り当て比率

・接続キャッシュをサポートした中間層を使用している場合、実行時ロードバランシング(RLB)の使用が可能

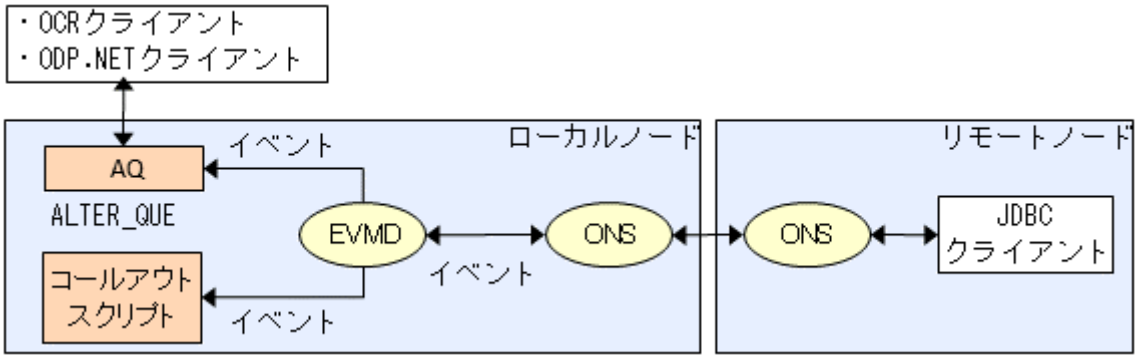
中間層では DB 接続処理に必要なコネクションをコネクションプールに事前作成しており、コネクションを取得する際にどのインスタンスに接続されているコネクションを使用するか決定する機能

・RLB は JDBC、OCI セッションプール、.NET 用の ODP 接続プールなど RLB に対応しているクライアントが必要

## 高速アプリケーション通知（FAN：Fast Application Notification）

・simplefan.jar を DL しクラスパスを通し java の API としてイベントを通知

・FAN を使用することでノードや DB、サービスなどの情報を設定したサブスクライバ（受信者）に通知できる




・FAN イベントは EVMD（Event Manager デーモン）が管理する

・ローカルノードに配置した「コールアウトスクリプト」や「アドバンストキューイング（AQ)」を使用したり、

別のリモートノードに ONS（Oracle Notification Services）を使用して通知させることが可能

・FAN イベントの書式（通知されてくる文字列の内容）

**ｲﾍﾞﾝﾄﾀｲﾌﾟ** VERSION=ﾊﾞｰｼﾞｮﾝ番号 [service=ｻｰﾋﾞｽ名] [database=DB 名] [instance=ｲﾝｽﾀﾝｽ名] [host=ﾎｽﾄ名]

**status=ｲﾍﾞﾝﾄｽﾃｰﾀｽ reason=ｲﾍﾞﾝﾄ理由** [cardinality=ｱｸﾃｨﾌﾞなｻｰﾋﾞｽﾒﾝﾊﾞｰ数] timestamp=日付

| イベントタイプ | 説明 | イベントタイプ | 説明 |
|---|---|---|---|
| DATABASE | データベース | SERVICE | サービス |
| INSTANCE | インスタンス | SERVICE_MEMBER | 特定インスタンス上のサービス |
| ASM | ASM インスタンス | SRV_PRECONNECT | シャドウサービス |
| NODE | クラスタノード | SERVICE_METRICS | ロードバランシングアドバイザ |

| ステータス | 説明 | ステータス | 説明 |
|---|---|---|---|
| UP | 起動 | PRECONN_UP | シャドウサービス起動 |
| DOWN | 停止 | PRECONN_DOWN | シャドウサービス停止 |
| NOT_RESTARTING | リモートノードへの F/O 失敗 | NODEDOWN | ノードダウン |
| RESTART_FAILED | 起動の失敗が再試行回数最大値に到達 | UNKNWON | ステータス不明 |

| 理由 | 説明 |
|---|---|
| USER | SRVCTL、SQL*Plus などのコマンドをユーザーが使用 |
| FAILURE | リソースチェック（CRSD など）による障害の検出 |
| DEPENDENCY | 障害状態にある別のリソースに依存している |
| AUTOSTART | CRS スタック起動に伴うリソース起動（CRS スタック停止前にリソースが**停止**していた場合） |
| BOOT | CRS スタック起動に伴うリソース起動（CRS スタック停止前にリソースが**起動**していた場合） |
| UNKNOWN | 内部アプリケーションの状態不明 |

### ＞ サーバー側コールアウト

・FAN イベント発生時に独自にプログラムしたシェルを呼び出す（コールアウト）させることが可能

・ローカルノードで発生したイベントによって自動起動される

・$ORA_CRS_HOME/racg/usrco ディレクトリに配置する

・複数のスクリプトを配置することで全てのスクリプトが実行されるが実行順序は不特定

### ＞ ONS の使用

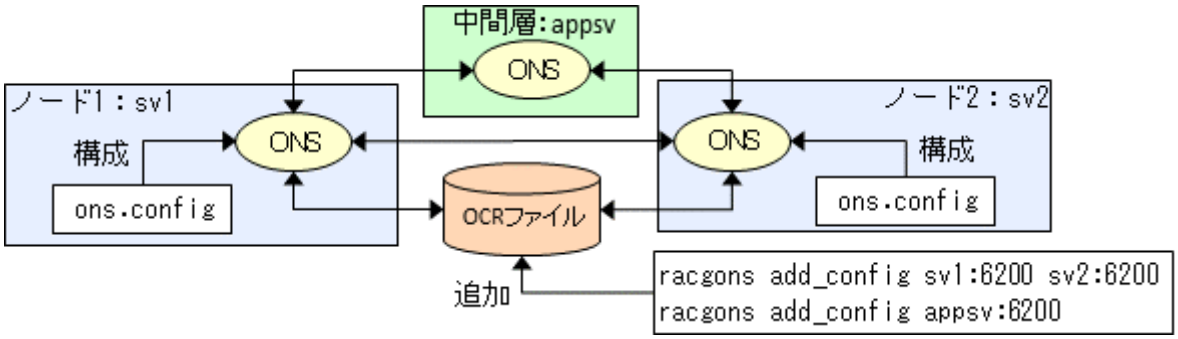
・リモートノードに FAN イベントを送信する場合に使用

・RAC サーバー側の ONS 構成は$ORA_CRS_HOME/opmn/conf/ons.config ファイルで構成する

```
localport=6100      ローカルノード⇔クライアント間の通信で使用するポート番号
remoteport=6200     リモートノード⇔ONS 間通信で使用するポート番号
loglevel=3
useocr=on           通信するノード情報が OCR ファイルに格納される（1 つのノードで racgons コマンドを行うだけでよくする）
```

・OCR ファイルにノード情報を追加（削除）する

$ORA_CRS_HOME/bin/racgons { add_config | remove_config } ﾉｰﾄﾞ名:ONS ﾘﾓｰﾄﾎﾟｰﾄ




**＞ 高速接続時フェイルオーバー（Fast Connection Failover：FCF）**

・中間層でコネクションプールを使用している場合、各コネクションが正常に使用できるか確認する必要がある

・サーバー側から高可用性イベントと通知させ、コネクションを自動的にメンテナンスすることが可能

・暗黙的接続キャッシュ（Implicit Connection Cache：ICC）と連携し、DOWN イベントが発生すると、
接続キャッシュ内のコネクションに停止マーク付けし接続のクリーンアップを行う

・UP イベントが発生すると再接続を行う

・ICC と FCF を有効化すると Java は自動的に ONS を利用することが可能になる為、FAN イベントの直接管理が不要になる

**透過的アプリケーションフェイルオーバー（Transparent Application Failover：TAF）**

・サービスを使用して接続したインスタンスが停止した時に自動的に再接続を行う機能

・TAF で接続確立しているかは、V$SESSION の FAILOVER_TYPE, FAILOVER_METHOD, FAILED_OVER 列で確認

・SELECT 中に F/O した場合はフェッチ済みのレコード以降のレコードから続きを行う

・トランザクション中の場合は、ORA エラーが出力する

| TAF の種類 | 説明 |
|---|---|
| TAF 基本接続 | BASIC　再接続が必要になった時にリスナーを使用して再接続先となるインスタンスへの接続確立が行われる<br>クライアント側で設定する方法と、サーバー側で設定する方法の 2 種類がある |
| TAF 事前接続 | PRECONNECT　最初の接続時に別インスタンスに「シャドウ接続」を確立しておき、再接続が必要になった時、<br>シャドウ接続を使って再接続する。クライアント側でしか設定できない |

**＞ クライアント側で TAF を構成する方法**

| ツール | |
|---|---|
| DBCA | サービス管理で作成したサービスを選択し、右ペイン下段にある TAF ポリシー構成から選択する |
| EM | [クラスタ管理データベース・サービス] の [サービスプロパティ]セクションで設定 |
| SRVCTL | 【作成】srvctl　add　service　-d DB 名　-s ｻｰﾋﾞｽ名　-r ｲﾝｽﾀﾝｽ [,ｲﾝｽﾀﾝｽ...]　-P { BASIC \| PRECONNECT }<br>【確認】srvctl　config service　-d DB 名 -a |

・DBCA 以外で TAF 構成した場合は、手動で tnsnames.ora の修正が必要

TAF 基本接続（**サーバー**側の tnsnames.ora 修正）

```
CRM =
-- 省略 --
      (CONNECT_DATA = (SERVICE_NAME = orcl)
      (FAILOVER_MODE=(TYPE=SELECT)
                     (METHOD=BASIC)
                     (RETRIES=10)
                     (DELY=5) )
  )
```

TAF 事前接続（**クライアント**側の tnsnames.ora 修正）

```
CRM =
-- 省略 --
      (CONNECT_DATA = (SERVICE_NAME = orcl)
      (FAILOVER_MODE=(BACKUP=CRM_PRECONNECT)
                     以降は基本接続と同じパラメータ )
  )
CRM_PRECONNECT =
-- 省略 --
```

```
(CONNECT_DATA = (SERVICE_NAME = orcl)
(FAILOVER_MODE=(BACKUP=CRM)
                以降は基本接続と同じパラメータ )
```

| パラメータ | 説明 |
|---|---|
| BACKUP | シャドウ接続に用いるネットサービス名を記述。2 つのサービス名をお互いに記述すれば良い |
| TYPE | SELECT（再接続後、SELECT 文を継続）または SESSION（再接続のみ実行） |
| METHOD | BASIC（基本接続）または　PRECONNECT（事前接続） |
| RETRIES | F/O 後に再接続を試みる回数 |
| DELY | 再接続を待機する間隔（秒） |

### ＞ サーバー側で TAF（基本接続）を構成する方法

・Oracle 10g R2 からサービス内に TAF パラメータを設定する方法がサポートされた

・サーバー側で設定した場合、Oracle Net クライアントの接続記述子に FAILOVER_MODE は記述しない

・DBMS_SERVICE パッケージを使用してサービスの TAF パラメータを設定する

```
DBMS__SERVICE_MODIFY_SERVICE(
  aq_ha_notifications => TRUE,                    ｱﾄﾞﾊﾞﾝｽﾄ･ｷｭｰｲﾝｸﾞ（AQ）を介して可用性の高い通知を送信できるようにする
  service_name        => 'CRM'                    サービス名を指定
  failover_type   => DBMS_SERVICE.FAILOVER_TYPE_SESSION,  再接続のみ行う（SELECT の継続は行わない）
  failover_method => DBMS_SERVICE.FAILOVER_METHOD_BASIC,  TAF の種類で基本接続を設定
  failover_retries    => 10,                      再試行を行う回数
  failover_delay      => 5);                      再試行間隔
```

### ＞ フェイルオーバー使用時の考慮事項

・TAF と FCF は何れもサービスを使用した接続ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞを使用できるが、両方同時に使用することは非推奨

・FCF はｺﾈｸｼｮﾝﾌﾟｰﾙが作成されている為、TAF 事前接続と同様に障害発生時に接続を確立する時間が不要になる

・TCP タイムアウトに依存するフェイルオーバーと依存しないフェイルーバーがある

| 機能 | タイムアウト依存 |
|---|---|
| VIP フェイルオーバー | |
| クライアント側の接続時フェイルオーバー | ○ |
| 透過的アプリケーションフェイルオーバー（TAF） | ○ |
| 高速接続フェイルオーバー（FCF） | |

## ■高可用性の設計

### 環境に応じた最大可用性アーキテクチャ

計画停止時間を短縮する解決策

| 解決策 | 説明 |
|---|---|
| ローリングアップグレード | Oracle に対するパッチ適用時に 1 ノードずつ順番にアップグレードすることで全停止しない |
| オンライン再定義 | 既存テーブルの定義変更時、DBMS_REDEFINITION パッケージを使用するオンライン再定義を使用することで、クライアントからの問い合わせやデータ変更を中断させない |
| オンライン再構築 | テーブルや索引などの断片化解消や、表領域の移動時に ONLINE 句を使用することで、クライアントからの問い合わせやデータ変更を中断させない |

計画外停止時間を短縮する解決策

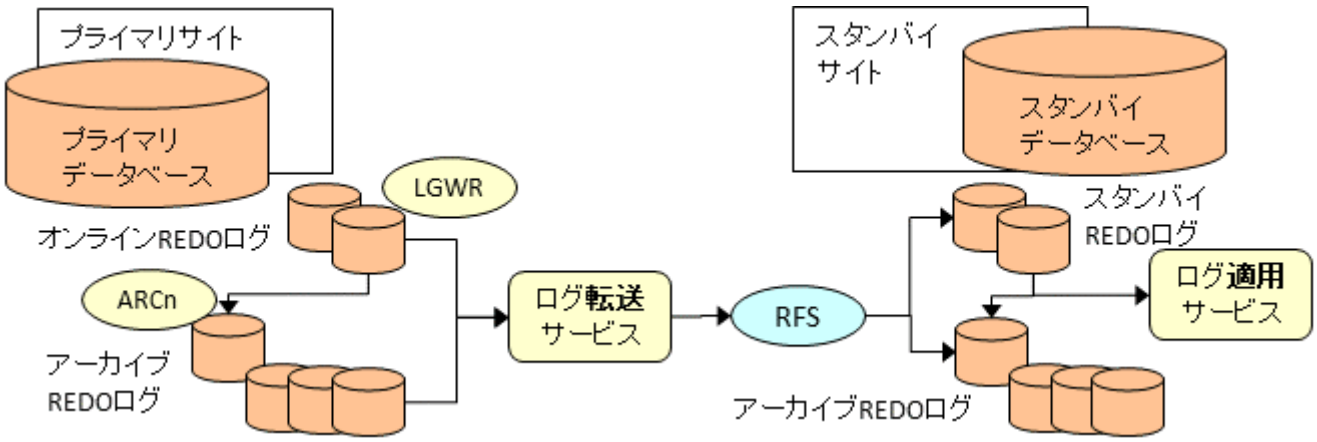
| 解決策 | 説明 |
|---|---|
| ファストスタートリカバリ | 増分チェックポイントを定期的に行うことでインスタンス/クラッシュリカバリに必要な時間を短縮 |
| RAC | 全インスタンス停止しない限りサービス提供可能。複数インスタンスへのリソース分散 |
| Data Guard | プライマリ DB の他にスタンバイ DB を用意することでシステム障害に備える |
| Oracle Streams | Data Guard と同様に別 DB を用意しプライマリ DB に対する変更を送信適用できる<br>Data Guard と異なり DB リリース間や異なるプラットフォーム間でも環境構築可能 |
| RMAN | データ障害、メディア障害に備えたバックアップの取得とリカバリ |
| ASM | ASM ディスクの冗長構成を行うことでディスク障害に備えることができる |
| フラッシュリカバリ | 人的エラー（誤ったトランザクションやテーブル削除、切り捨て）に対するリカバリ機能 |
| HARD | Hardware Assisted Resilinet Data は I/O パスでのデータ破損を防止する為に設計された飽和的なプログラム。DB_BLOCK_CHECKSUM が TRUE の時に有効 |

### Data Guard

・サイト障害（環境障害）やデータ障害（人的エラー）に備えてメインの DB のバックアップをコピーして作成した別の DB を準備しておく技術

・プライマリ DB からスタンバイ DB に対して REDO ログを転送し、REDO ログを自動的に適用することで、プライマリ DB と同じデータを保持できるようにする

**＞ スタンバイデータベース**



| サービス名 | 説明 |
|---|---|
| ログ転送<br>サービス | プライマリ DB からスタンバイ DB に REDO ログを転送し RFS（Remote File Server）で受信する<br>オンライン REDO ログであれば転送先のスタンバイ REDO ログに、<br>アーカイブ REDO ログであれば転送先のアーカイブ REDO ログに格納する |
| ログ適用<br>サービス | スタンバイ DB に REDO ログを適用することでプライマリ DB の変更内容が反映される<br>リアルタイム適用機能が有効化されていれば満杯になったスタンバイ REDO ログから適用し、<br>無効であればアーカイブログから適用する |

REDO ログ適用方法

| 適用方法 | 説明 |
|---|---|
| フィジカル<br>スタンバイ | DB 構造がプライマリ DB と全く同じ DB で、ログ適用サービスは「Redo Apply」と呼ばれる<br>MRP（Management Recovery Process）がリカバリのメカニズムを使用してスタンバイ DB に REDO ログを適用<br>ログ適用中は DB を OPEN することができないが、リカバリモードを解除すれば読み取り専用で OPEN したり<br>フラッシュバック DB 機能を用いて一時的に読み書き可能で OPEN することが可能 |
| ロジカル<br>スタンバイ | 最初は DB 構造がプライマリと同じだが後から異なる構造に変更できる DB で、ログ適用サービスは「SQL Appay」<br>LSP（Logical Standby Process）が LogMiner を使用して REDO ログから SQL 文を抽出し、SQL 文をスタンバイ DB で<br>実行する。ログ適用中も DB の OPEN が可能で、索引や MV を追加するなどのチューニングが可能 |

### ＞ スイッチオーバーとフェイルオーバー

・フィジカルスタンバイ、ロジカルスタンバイの何れも、プライマリ DB とスタンバイ DB の役割を切り替えることが可能

| 切り替え方法 | 説明 |
|---|---|
| スイッチオーバー | プライマリ DB の計画停止時にプライマリとスタンバイを相互に切り替える |
| フェイルオーバー | プライマリ DB の計画外停止に、何れかのスタンバイ DB をプライマリ DB として OPEN する |

### ＞ Data Guard Broker

・Data Guard 環境におけるプライマリ DB とスタンバイ DB の管理や監視を集中的に行うためのフレームワーク

・DG_BROKER_START 初期化パラメータを TRUE にすることで DMON（Data Guard Broker Moniter Process）が起動する

・DMON により DB の状態を集中管理され、管理者からの管理コマンドを DB に適用する

・管理情報は Data Guard Broker 構成ファイル（バイナリ）を使用して DB の役割や状態を記録する

・EM Grid Control や、DGMGL コマンドで管理を行う

### ＞ ファストスタートフェイルオーバー

・Oracle10g R2 以降であればプライマリ DB 停止の障害検出時に自動的にスタンバイ DB へ F/O する

・Oracle Client Administrator ソフトウェアからオブザーバをインストールされる

・オブザーバは個別にサーバーに配置したコンポーネントで、プライマリ DB とスタンバイ DB を監視する

・障害検知すると可能な限り短時間で F/O され、元のプライマリ DB はリカバリ完了後、自動的にスタンバイ DB になる

## RAC と Data Guard の連携

・スタンバイ DB が RAC 環境の場合、スタンバイサイトの各ノードで REDO ログを受信できる

・REDO ログを適用できるインスタンスは 1 つのみ

### ＞ RAC と Data Guard Broker

・ファイストスタート F/O を利用しない場合、プライマリサイトの障害が検出されると Clusterware から DB 管理者に NOT_RESTARTING（再起動できない）イベントが渡される

・DB 管理者はプライマリサイトを手動で修復するか、Data Guard Broker を使用して F/O を行う

・F/O が実行される場合、DMON プロセスが元のプライマリサイトを停止し、使用不可を clusterware に通知する

・役割の変更が完了すると DMON が新しいプライマリサイトを使用可能にし、起動するように clusterware に通知する

### ＞ RAC の Data Guard Broker 構成ファイル

・2 つの初期化パラメータで、構成情報を格納したファイル（バイナリ）のパスを指定する（同じ値にする必要がある）
DG_BROKER_CONFIG_FILE1　と　DG_BROKER_CONFIG_FILE2

・DMON プロセスが起動していない時のみ上記パラメータは変更できる

・スタンバイサイトを RAC 構成で Data Guard Broker を使用する場合、構成ファイルは共有ストレージに保存する

・CFS（クラスタファイルシステム）、ASM、RAW デバイスの何れかに格納する

## RAC 環境へのパッチ適用

### ＞ Oracle Universal Installer（OUI）の使用

・PSR（Patch Set Release）には OUI が同梱されている

・RAC 環境で PSR を適用するには何れかのノードで OUI を起動し PSR のインストールを開始することで全ノード適用される

・Clusterware は RAC と同じバージョンか、新しいバージョンである必要がある為、Clusterware の PSR から適用する

### ＞ Opatch の使用　（ $ORACLE_HOME/OPatch/opatch ）

・個別パッチの適用時に使用する（適用方法は、入手した個別パッチ内の README.txt に記載）

| 実施内容 | コマンド例（$ORACLE_HOME/OPatch） |
|---|---|
| 適用状況確認 | opatch　lsinventory　-detail |
| パッチ適用 | opatch　apply　[オプション]　/patches/133469（/patches/133469 に存在するパッチを、$ORACLE_HOME に適用）<br>実行するノードのみでパッチ適用する場合は、-local オプションを付与する |
| パッチ削除 | opatch　rollback -id 133469 -ph /patches/133469（/patches/133469 に存在するパッチを削除） |
| モード確認 | opatch　query　/patches/133469（該当パッチがローリングモードに対応しているか確認）<br>パッチ解答したディレクトリの/etc/config/inventory ファイル内<online_rac_installable>要素が true なら対応してる |

| apply オプション | 説明 |
|---|---|
| デフォルトは<br>ローリングモード<br>⇒できない場合は<br>-all_nodes になる | ローカルノードにパッチ適用後、残りのノードは 1 ノードずつインスタンス停止、パッチ適用、起動を繰り返す<br>パッチ適用中のインスタンスは停止するが他のインスタンスは起動している為、停止時間は発生しない<br>適用するパッチがローリングモードがサポートされている必要がある<br>共有ストレージに Oracle ホームを配置している場合はローリングモードは使用できない |
| -all_nodes<br>全ノードモード | パッチ適用中は全てのインスタンス停止が必要（ローリングモードが使用できな場合）<br>ローカルノードにパッチ適用すると、残り全てのノードに伝播され最後にインベントリを更新する |
| -minimize_downtime<br>最小停止時間モード | パッチ適用ノードのみ停止した状態で、パッチ適用を行う<br>最初のノードのパッチ適用完了後、残り全てのノードに伝播され、最後にインベントリを更新する<br>最初のノードパッチ適用後、次のインスタンスを停止し、最初のノードが起動完了するまでが停止時間となる |
| -local | パッチ適用を各ノードで個別に行う |

### > Enterprise Manager の使用

[設定] - [パッチ適用設定]で必要な設定を行う

### > ロジカルスタンバイを使用したローリングアップグレード手順

A：プライマリサイト　A'：新スタンバイサイト（プライマリ⇒スタンバイへスイッチ）

B：スタンバイサイト　B'：新プライマリサイト（スタンバイ⇒プライマリへスイッチ）

**① スタンバイサイト（B）のアップグレード**

「A」からログ転送サービス（SQL Apply）を停止し、「B」をアップグレードする

「A」の REDO ログはログキューに蓄積される

**② SQL Apply の再開**

「A」からアップグレードした「B」へ REDO ログの適用が行われる

**③ スイッチオーバーの実行**

「A」を新スタンバイ DB（A'）へ。「B」を新プライマリ DB（B'）にする

**④ 新スタンバイ DB（A'）のアップグレード**

「B'」からのログ転送サービスを停止し、「A'」をアップグレードする（「B'」の REDO ログはログキューに蓄積される）

**⑤ SQL Apply の再開**

「B'」からアップグレードした「A'」へ REDO ログの適用が行われる

**⑥ スイッチオーバーの実行**

「A'」をプライマリ DB（A）へ。「B'」をスタンバイ DB（B）にする

※ロジカルスタンバイによる SQL Applay はサポートされない型がある為、システムよっては使用できない

**最適なストレージ構成**

・共有ストレージのストライプ化またはミラー化に RAID か ASM を選択できる

| ミラー化 | 説明 |
|---|---|
| ASM によるミラー | 通常の冗長性（2 つのミラー）または高いミラー（3 つのミラー）を選択 |
| ASM によるストライプ | ASM ディスクグループ内のメンバーディスクを増やすことで、グループ内ディスク間でストライプする |
| RAID1＋ASM ミラー | 容量が 2 倍になる。RAID5 はパリティ容量のみだが書き込みパフォーマンスは低下する<br>RAID1 + ASM ミラーはパフォーマンスの観点から非推奨。ASM によるミラーの方がパフォーマンスが良い |
| **ストライプ化** | **説明** |
| RAID によるストライプ | 複数のディスクを 1 つのセット（ディスクアレイ）にし、論理的に分割（LNU）して使用<br>LNU 内は同じスピードを持つディスクを使用する |
| ASM＋RAID ストライプ | RAID のストライプサイズは ASM の割り当てユニット（AU）サイズである 1MB 以下の 2 の累乗（128KB、256KB など）にすることが推奨されている |

### > ASM ディスクグループの配置

・データ用とフラッシュリカバリ領域の ASM ディスクグループを用意するのを推奨

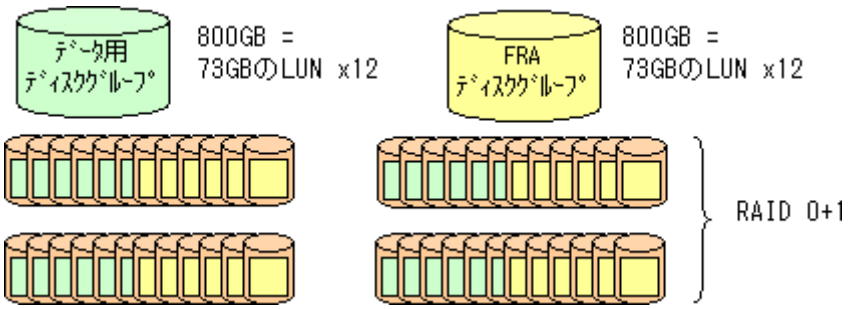
・ディスクの外周の方が高速な為、外周を使用した LUN をデータ用、内周を使用した LUN をフラッシュリカバリ用にする

・容量やパフォーマンス特定が異なるディスクを使用する場合、同じ LUN に含めないこと

・可用性が高いのは異なるアレイにデータ用ディスクグループとフラッシュリカバリ領域用ディスクグループを配置した時

### > 73GB のディスクを 12 台で 1 つのディスクアレイを構成し、4 つのアレイを持つ環境での例

#### ASM によるストライプ化のみ




メリット

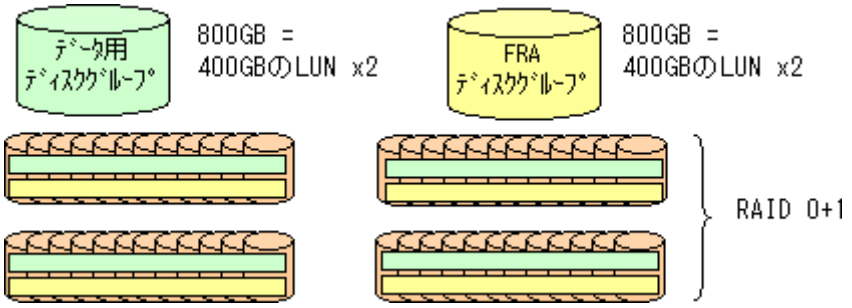
・少ない単位（73GBx4 アレイ）でディスク増設可能

デメリット

・異なる ASM ディスクグループには分散されない為、
全てのディスク間で十分な分散ができない

・管理する LUN が多くなる

#### RAID によるストライプ化：アレイ辺り複数グループ




メリット

・データ一用領域に高速な領域（ディスクの外周）
を割り当て可能

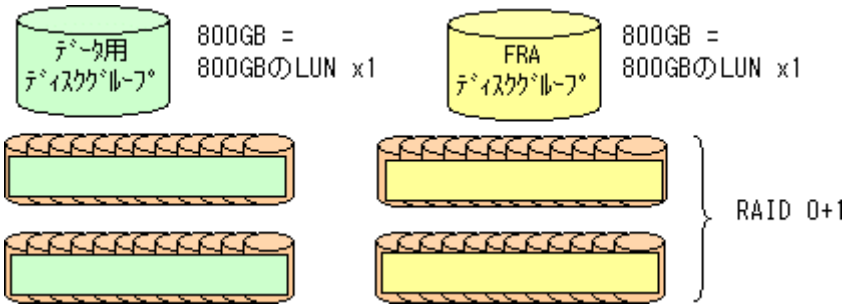
・データ分散が十分に行われる

デメリット

・大きい単位でディスク増設が必要

・データ用と FRA のディスクグループで競合が発生

#### RAID によるストライプ化：アレイ辺り 1 グループ




メリット

・データ分散が十分に行われる

・データ用と FRA のディスクグループで競合しない

・可用性が向上する（ディスクコントローラ障害時
異なる ASM ディスクグループへの影響が最小限）

デメリット

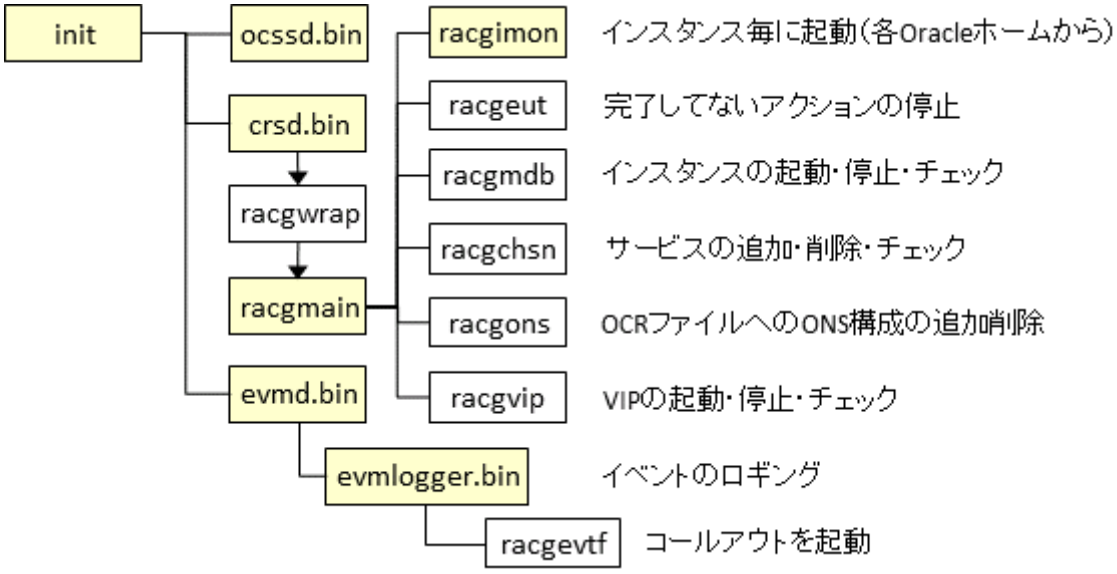
・大きい単位でディスク増設が必要

・領域が無駄になる可能性

## ■Oracle Clusterware の管理

### Oracle　Clusterware スタック

Oracle Clusterware の機能を提供する CRS スタックは、機能実行のために子プロセスを起動する




| プロセス | 説明 |
|---|---|
| ocssd.bin | **CSSD**（Cluster Synchronization Services デーモン）は投票ディスクを使用してノード間の状態を監視する<br>Oracle ユーザーにて起動され、CSSD が異常終了した場合、ノードが再起動する<br>ベンダー製クラスタウェアと併用する場合はクラスタウェアと統合される<br>併用しない場合は、root によって起動された OPRCD(プロセスモニタデーモン)がノードを停止する<br>Windows/Linux 環境では hangcheck-timer などのカーネルドライバが同様の機能を提供する為、起動しない |
| crsd.bin | **CRSD**（Cluster Ready Services デーモン）は OCR ファイルを使用して Oracle Clusterware に登録<br>root ユーザーによって起動され、各アプリケーションで定義したアクションスクリプトを使用し起動、停止、F/O 等を行う<br>ASM と DB に関連するプロセスとして RACGIMON（インスタンスモニター）を起動し、状態監視や FAN などの機能を提供する<br>CRSD は racgwrap スクリプトから racgmain プログラムを起動し、更に各アクション実行の為、子プロセスを一時的に起動する |
| evmd.bin | **EVMD**（Event Manager デーモン）は、クラスタイベントの受信、転送を行う<br>Oracle ユーザーによって起動され、イベントを受信すると、イベントをログ化し、子プロセスである EVMLOGGER（イベントマネージャロガー）にイベント処理を要求する。EVMLOGGER は ONS（Oracle Notification Services）にイベントを送信したり、RACGEVTF を起動しコールアウトを実行する |

### > Oracle Clusterware スタックの手動管理

・root ユーザーにて、$ORA_CRS_HOME/bin/crsctl コマンドを事項

| コマンド | 説明 |
|---|---|
| crsctl start crs | CRS スタック（ocssd.bin、crsd.bin、evmd.bin）の起動 |
| crsctl stop crs | CRS スタックの停止 |
| crsctl check crs | CRS スタックの起動状態確認 |
| crsctl enable crs | CRS スタックの自動起動の有効化（次回起動時に CRS スタックは起動される） |
| crsctl disable crs | CRS スタックの自動起動の無効化（次回起動時に CRS スタックは起動しない） |

## クラスタリソースの管理

・クラスタリソースの状態は crs_stat コマンドで確認する

・クラスタリソースは OCR ファイルに登録されﾘｿｰｽﾌﾟﾛﾌｧｲﾙとｱｸｼｮﾝｽｸﾘﾌﾟﾄの内容に従って CRSD で管理される

・リソースプロファイルは　crs_stat -p リソース名　で確認する

| ファイル | 説明 |
|---|---|
| リソースプロファイル | チェック間隔、障害ポリシー、アクションスクリプトの名前、アプリケーション管理に使用される権限、<br>リソース依存性などの情報が含まれている |
| アクションスクリプト | アプリケーションの起動、停止、チェックの実行を行うための方法を記述したスクリプト<br>通常は「$ORACLE_HOME/bin/racgwrap」が使用される |

### クラスタリソースの属性

| 属性 | 説明 | |
|---|---|---|
| NAME | クラスタリソース名 | |
| TYPE | application 固定 | |
| ACTION_SCRIPT | アクションスクリプト名と保存場所 | |
| ACTIVE_PLACEMENT | デフォルトは「0」で、「1」を指定するとノードの追加や起動時に、リソースの配置が再評価される | |
| AUTO_START | CRS スタックが再起動する前にリソースが実行中だったかに依存せず、リソースを自動起動するか | |
| | 0 (restore) | 前回停止前に起動していたら、起動する |
| | 1 (always) | 常に起動する |
| | 2 (never) | 常に起動しない |
| CHECK_INTERVAL | アプリケーションのチェックを実行する間隔（秒） | |
| FAILOVER_DELAY | リソースの再起動や F/O を試行する前に待機する時間 | |
| FAILURE_INTERVAL | 障害しきい値（FAILURE_THRESHOLD）が適用される間隔（0 の場合は障害追跡が無効になる） | |
| FAILURE_THRESHOLD | FAILOVER_DELAY 時間内に検出される障害しきい値（障害最大数：最大 20 まで設定可能）<br>値を超過するとリソースが使用不可マークとなり監視を停止する（0 で障害追跡は無効になる） | |
| HOSTING_MEMBERS | リソースの配置可能なノードリスト（複数ノードは空白で区切る） | |
| PLACEMENT | リソース起動するノードを選択する配置ポリシー | |
| | balanced | 実行中リソースが最も少ないノードを優先する。決定できなければ使用可能な任意のノード |
| | favored | HOSTING_MEMBERS で指定された順。何れも不可なら使用可能な任意のノード |
| | restric | HOSTING_MEMBERS で指定された順。何れも不可であればリソースを起動しない |
| REQUIRED_RESOURCES | 当該リソースが起動する条件となるリソースを定義（複数ある場合は空白で列挙）<br>リストに定義したリソースが起動しない場合は、当該リソースは再配置や停止が行われる | |
| RESTART_ATTEMPTS | リソースの再配置が試行される場合に 1 つのノードで再起動が試行される最大回数<br>再起動すると RESTART_COUNT が 1 つカウントされ、RESTART_ATTEMPTS まで上がる（超えるとリソース監視停止）<br>RESTART_COUNT をリセットするには手動で再起動するか、UPTIME_THRESHOLD 期間が経過するのを待つ | |
| SCRIPT_TIMEOUT | アクションスクリプト内のプログラムを実行する最大時間（秒）<br>この時間内に完了しない場合は、エラーを返却し EVMD にイベントを通知する | |
| START_TIMEOUT | アクションスクリプトで起動完了とみなすまでの時間（秒） | |
| STOP_TIMEOUT | アクションスクリプトで停止官僚とみなすまでの時間（秒） | |
| UPTIME_THRESHOLD | RESTART_COUNT が RESTART_ATTEMPTS に到達した時、最後に起動してからの監視再開するまでの間隔<br>単位は週(w)、日(d)、時間(h)、分(m)、秒(s) | |

| | |
|---|---|
| USR_XXX | ユーザー定義のアプリケーションで使用する属性（デバッグの有無指定する USR_ORA_DEBUG など） |

### ＞ クラスタリソース変更

crs_register リソース名 -update -o オプション=値

| ｵﾌﾟｼｮﾝ | 変更するパラメータ | ｵﾌﾟｼｮﾝ | 変更するパラメータ | ｵﾌﾟｼｮﾝ | 変更するパラメータ | ｵﾌﾟｼｮﾝ | 変更するパラメータ |
|---|---|---|---|---|---|---|---|
| ap | ACTIVE_PLACEMENT | fd | FAILOVER_DELAY | ra | RESTART_ATTEMPTS | pt | STOP_TIMEOUT |
| as | AUTO_START | fi | FAILURE_INTERVAL | st | SCRIPT_TIMEOUT | ut | UPTIME_THRESHOLD |
| ci | CHECK_INTERVAL | ft | FAILURE_THRESHOLD | rt | START_TIMEOUT | | |

### ＞ インスタンス（inst、asm）の自動起動を停止する場合

・.db　.cs　.srv も調整する必要がある

・AUTO_START=2（常に起動しない）にする場合、RESTART_ATTEMPTS の値を 0 または 1 にすることが推奨される

```
-- DB/ASM インスタンスの変更（手動起動、最大試行回数=1）
$ crs_register ora.sv1.ASM1.asm -update -o as=2,ra=1
$ crs_register ora.sv1.ASM2.asm -update -o as=2,ra=1
$ crs_register ora.sv1.ASM1.inst -update -o as=2,ra=1
$ crs_register ora.sv1.ASM2.inst -update -o as=2,ra=1
--DB、サービス、サービスメンバーの変更
$ crs_register ora.sv1.RACDB.db -update -o as=2,ra=1
$ crs_register ora.sv1.RACDB.HR.cs -update -o as=2,ra=0
$ crs_register ora.sv1.RACDB.HR.srv -update -o as=2,ra=0
$ crs_register ora.sv1.RACDB.HR.srv -update -o as=2,ra=0
```

・データベースとインスタンスは、srvctl modify database コマンドでも変更可能

```
-- 現行設定の確認
$ srvctl config database -d RACDB -a
-- 手動起動（自動起動）に変更
$ srvctl modify database -d RACDB -y MANUAL（AUTOMATIC）
```

## Oracle Clusterware によるアプリケーションの保護

10g R2 よりサードパーティ製のアプリを Oracle Clusterware で保護することができる

### ＞ ユーザー定義のアプリケーション

・Oracle Clusterware に登録したサードパーティ製のアプリケーション

・クラスタリソースと同様にアクションスクリプト内のチェックアクションに基づいてプロセスを監視し、
　障害が発生した場合は再起動や F/O が実行される

ユーザー定義のアプリケーション管理コマンド

| コマンド | 説明 | コマンド | 説明 |
|---|---|---|---|
| crs_profile | リソースプロファイルの作成、検証、削除、変更 | crs_relocate | リソースを別のノードに移動 |
| crs_register | リソースの登録、既存リソースプロファイルの一部変更 | crs_stop | リソースの停止 |
| crs_start | リソースの起動 | crs_unregister | リソースの削除 |
| crs_stat | リソースの現行ステータス確認 | | |

### ＞ アプリケーション VIP

・ユーザー定義のアプリケーションが NW を介してアクセスする場合、アプリケーション VIP を作成することができる

・アプリケーション VIP はアプリケーションが停止した場合に使用される（RAC の VIP はノード停止の場合に使用される）

RAC VIP とアプリケーション VIP の違い

| 状況 | RAC VIP | アプリケーション VIP |
|---|---|---|
| F/O タイミング | ノード停止、パブリック NW の停止 | アプリケーション停止 |
| F/O 先 | 障害が発生してないノード | 障害が発生してないノード |
| F/O されるコンポーネント | VIP のみ | VIP とアプリケーション |
| F/O 後の VIP 接続 | 受け付けない | 受け付ける |
| VIP の構成 | 複数コンポーネント（DB など）で共有 | 1 つのアプリケーションに 1 つの VIP |

### CRS フレームワークの利用

WEB サーバーを Oracle Clusterware とアプリケーション VIP で保護する方法を以下に示す

WEB サーバーなので、VIP とアプリケーションの 2 つ定義する必要がある

### ＞ アプリケーション VIP の構成

① リソースプロファイルの作成

crs_profile コマンドにて作成する

| オプション | 説明 | オプション | 説明 | |
|---|---|---|---|---|
| -create | アプリケーション VIP 名を指定 | -o | アプリケーション VIP の情報を指定 | |
| -t | アプリケーションタイプとして application を指定 | | oi | 使用する NIC 名称 |
| | | | ov | アプリケーション VIP で使用する IP アドレス |
| -a | ACTION_SCRIPT 属性を指定 | | on | アプリケーション VIP で使用するサブネット |

```
$ crs_profile -create APP-VIP -t application -a $ORA_CRS_HOME/bin/usrvip -o oi=eth1,ov=192.168.11.61,on=255.255.255.0
# crs_profile コマンドを root ユーザーで実行した場合は、 $ORA_CRS_HOME/crs/profile/APP-VIP.cap に作成される
# crs_profile コマンドを oracle ユーザーで実行した場合は、$ORA_CRS_HOME/crs/public/APP-VIP.cap に作成される
```

② Oracle Clusterware への登録

```
$ crs_register APP-VIP [-u dir リソースプロファイルの格納されているディレクトリ名]
※ 省略時はでデフォルトのディレクトリ（ root で実行した場合は、$ORA_CRS_HOME/crs/profile/ ）になる
```

③ 権限の設定

```
-- アプリケーション VIP は root ユーザーにて実行
$ crs_setperm APP-VIP -o root
-- 管理作業は oracle ユーザーに権限を付与
$ crs_setperm APP-VIP -u user:oracle:r-x
```

④ リソースプロファイルを配置

リソースプロファイルは関連する全てのノードに必要なので、他ノードへコピーして配置する

⑤ アプリケーション VIP の起動

```
-- 管理権限が付与されたユーザー（oracle）で VIP 起動
$ crs_start APP-VIP
```

### ＞ アプリケーションの構成

① アクションスクリプトの作成（srvctl コマンドの引数によって処理を分岐させる）

アプリケーションの起動（start）、停止（stop）、チェック（check）を行うコマンドを記述した

スクリプトファイル（apache なら start の時、apachectl start など）を$ORA_CRS_HOME/crs/script に配置する

② リソースプロファイル作成

crs_profile コマンドにて作成する

| ｵﾌﾟｼｮﾝ | 説明 | ｵﾌﾟｼｮﾝ | 説明 |
|---|---|---|---|
| -create | アプリケーション名を指定 | -r | REQUIRED_RESOURCES 属性（ｱﾌﾟﾘｹｰｼｮﾝ VIP を指定） |
| -t | アプリケーションタイプを指定（application で固定） | -o | アプリケーション情報を指定 |
| -a | ACTION_SCRIPT 属性を指定<br>$ORA_CRS_HOME/crs/script に存在しなければﾃﾝﾌﾟﾚｰﾄ作成 | | ci チェック間隔<br>ov 再起動試行の最大回数 |

```
$ crs_profile -create WAS -t application -r APP-VIP -a WAS.scr -o ci=30,ra=5
#ファイルが作成される  $ORA_CRS_HOME/crs/profile/WAS.cap
```

③ Oracle Clusterware への登録

```
$ crs_register WAS [-u dir リソースプロファイルの格納されているディレクトリ名]
```

④ 権限の設定

```
-- アプリケーションは root ユーザーにて実行
$ crs_setperm WAS -o root

-- 管理作業は oracle ユーザーに権限を付与
$ crs_setperm WAS -u user:oracle:r-x
```

⑤ リソースプロファイルとアクションスクリプトの配置

関連する全てのノードにリソースプロファイルとアクションスクリプトを配置する

⑥ アプリケーション起動

```
-- 管理権限が付与されたユーザー（oracle）で VIP 起動
$ crs_start WAS
```

### ＞ リソースの手動再配置

crs_relocate　リソース名　[ｵﾌﾟｼｮﾝ]

| オプション | 説明 |
|---|---|
| -c crs_node | crs_node で指定したノードへリソースを再配置する |
| -f | リソース及び依存リソースを強制的に再配置 |
| -s source_node | 移動元のノードを指定。-c で移動先を指定しない場合はポリシーに従って再配置される |
| -q | メッセージを表示しない（サイレントモードで実行） |

### ＞ リソースの削除

crs_stop 実行後、crs_unregister コマンドで削除

## ネットワーク構成の変更

RAC 環境の NW はインストール時に OCR ファイルに登録される為、IP アドレスや VIP アドレスを変更する場合、

ノードの物理 NIC の IP アドレスを変更後、$ORA_CRS_HOME/bin/oifcfg コマンドにて OCR ファイルの内容を更新する

### > パブリックネットワークとインターコネクトの変更

```
-- 現行ネットワーク構成の確認
$ oifcfg getif

-- パブリックネットワークアドレスの削除と追加
$ oifcfg delif -global eth0
$ oifcfg setif -global eth0/192.168.12.0:public

-- インターコネクトネットワークアドレスの削除と追加
$ oifcfg delif -global eth1
$ oifcfg setif -global eth1/192.168.102.0:cluster_interconnect
```

### > VIP アドレスの変更

・1 ノード毎にクラスターリソースを停止後、IP アドレスを変更し、リソースを再開させる

```
-- VIP に依存している全てのクラスタリソース停止
$ srvctl stop instance -d orcl -i orcl1
$ srvctl stop asm -n orac1
$ srvctl stop nodeapps -n orac1

-- VIP インターフェースが実行されてないことを確認
$ ifconfig -a | grep eth0

-- OCR ファイル内の IP アドレスを変更（-A VIP アドレス/ネットマスク/IF）
$ su -
# srvctl modify nodeapps -n orcl1 -A 192.168.10.111/255.255.255.0/eth0

-- VIP に依存している全てのクラスタリソース起動
$ srvctl start nodeapps -n orac1
$ srvctl start asm -n orac1
$ srvctl start instance -d orcl -i orcl1
```

・上記作業を全てのノードで繰り返す

## ■投票ディスクと OCR ファイルの調整

### 投票ディスク構成

・クラスタノード間で正常に通信できているかどうか監視するハートビートの結果を格納

・Oracle10g R2 からは多重化（3 多重以上）が可能

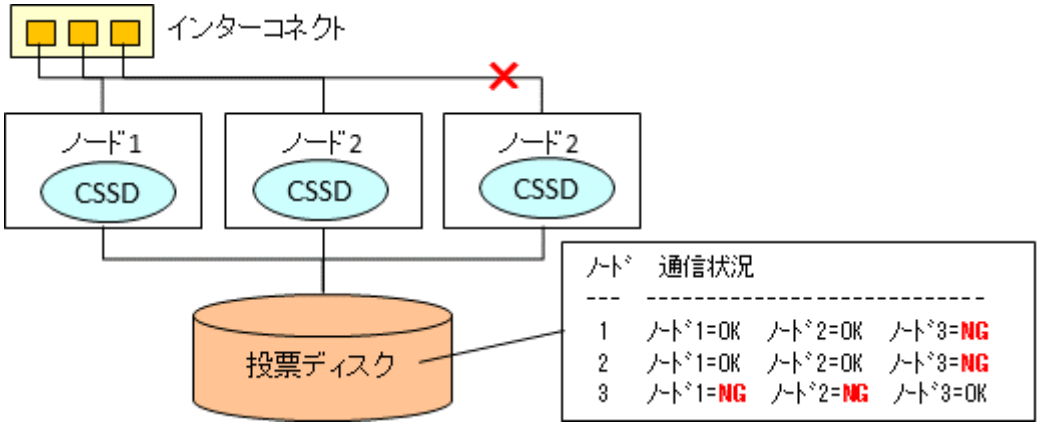
#### ＞ 投票ディスクの機能

・CSSD はインターコネクトを介して他ノードと通信し、通信状況を投票ディスクに格納

・自ノードから見て他のノードとの正常通信が多いノードを残存ノードとする（少ないノードを削除する）

・正常通信のノード数が同じ場合は、大きいノード番号（olsnodes -n で確認）が割り当てられてる方を削除する

・削除されたノードは全ての I/O を停止する為、ノードが再起動される




#### ＞ 投票ディスクの動作に影響を与えるパラメータ

・インターコネクトを使用した NW のハートビートと投票ディスクの I/O は 1 秒間隔

・ハートビートが認識できない、または、I/O がタイムアウトすると該当ノードは CSSD が停止し再起動する

・動作状況は、$ORA_CRS_HOME/log/<ノード名>/cssd/ocssd.log

タイムアウトの設定パラメータ

| パラメータ | 説明 |
|---|---|
| MISSCOUNT | インターコネクト全体でハートビートが認識できないと判定されるまでの最大許容範囲（秒）<br>ベンダー製クラスタウェア使用時のデフォルトは 600 秒（未使用時は 30 秒、Linux は 60 秒）<br>ex）crsctl set css misscount 120　現在の値を確認するには、crsctl get css misscount |
| DISKTIMEOUT<br>※10.2.0.2 以降 | 投票ディスクへの I/O が実行できないと判定されるまでの最大許容範囲（秒）デフォルトは 200 秒<br>ex）crsctl set css disktimeout 100 |

#### ＞ 投票ディスクの多重化

・一貫性のある読み取りデータが過半数を満たしていれば動作を続ける
　満たさない場合は、クラスタ構成する全ノードで再起動が発生する

・1 つの投票ディスク障害に耐えるには 3 多重、2 つの投票ディスク障害に耐えるには 5 多重にする必要がある

・投票ディスクはシンボリックリンクの使用が推奨されている
　投票ディスクの場所は OCR ファイルに格納されるが、OCR ファイルは直接編集できない為

```
-- 現在の設定確認
$ crsctl query css votedisk

-- 投票ディスクの追加
$ crsctl add css votedisk 投票ﾃﾞｨｽｸﾊﾟｽ [-force]

-- 投票ディスクの削除
$ crsctl delete css votedisk 投票ﾃﾞｨｽｸﾊﾟｽ [-force]
-- force オプションは CRS スタックが停止している時に必要なオプション
-- 10g まで全ノードの CRS スタック停止が必要だったが、11g からは起動したまま追加削除が可能
```

## 投票ディスクのバックアップ/リカバリ

・バックアップは、Clusterware インストール後、ノードの追加または削除後に行う

・dd if=投票ﾃﾞｨｽｸﾊﾟｽ of=ﾊﾞｯｸｱｯﾌﾟﾊﾟｽ bs=4k　でバックアップする（オンラインでの取得が可能）

・投票ディスクに障害が発生すると、alert.log にオフラインになっているメッセージが出力される

・リカバリは全てのノードの CRS スタックを停止後、dd if=ﾊﾞｯｸｱｯﾌﾟﾊﾟｽ of=投票ﾃﾞｨｽｸﾊﾟｽ bs=4k　で復旧させる

・全ての投票ディスクが破損した場合は Clusterware の再インストールが必要

## Oracle Cluster Registry (OCR) の構成

・クラスタリソース情報や、投票ディスクの場所などが格納されている

・10g R2 からは多重化も可能

### > OCR の機能

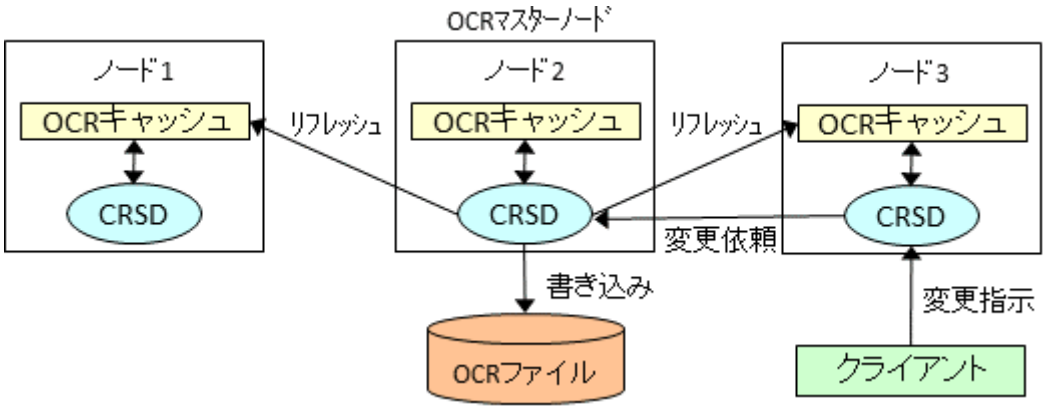
・各ノードには OCR ファイルの内容をキャッシュしたメモリが用意されている

・CRSD（Cluster Ready Service デーモン）によって OCR キャッシュがリフレッシュされる

・OCR ファイルの内容を読み書きできるのは何れかのノードの CRSD（OCR ファイルマスターノードの CRSD）のみ

・OCR ファイルマスターノードは最初に CRSD が起動したノード

$ORA_CRS_HOME/log/ﾉｰﾄﾞ名/crsd/crsd.log | grep MASTER で現在のマスターノードが確認できる




・OCR ファイルに格納されている情報はキーと値のペアとしてツリー上の構成

| キー | 説明 | |
|---|---|---|
| SYSTEM | CRS スタックに関するデータ | |
| | css | ネットワーク、投票ディスク、ノード情報 |
| | version | ソフトウェアバージョン情報 |
| | ORA_CRS_HOME | CRS ホームの場所 |
| | evm | イベントの通知に使用する情報 |

| | | |
|---|---|---|
| | OCR | OCR ファイルのバックアップ情報 |
| DATABASE | データベースに関するデータ | |
| | NODEAPPS | ノードアプリケーション情報 |
| | ASM | ASM インスタンス情報 |
| | DATABASES | クラスタデータベース、インスタンス情報 |
| | ONS_HOSTS | OCR ファイルに登録済みの ONS |
| CRS | ユーザー定義のアプリケーションに関するデータ | |

# ocrdump [{ファイル名 | -stdout}] [-xml] [-noheader] [-keyname キー名]

| パラメータ | 説明 |
|---|---|
| ファイル名 | ダンプ結果のファイル名（省略時は OCRDUMPFILE） |
| -stdout | ダンプ結果を標準出力する |
| -xml | ダンプ結果を XML 形式にする |
| -noheader | ダンプ結果にヘッダー情報（日時、ユーティリティ名）を含めない |
| -keyname キー名 | 指定したキー名のツリー配下のみをダンプ結果に出力する |
| -backupfile ファイル | OCR のバックアップファイルを識別する（内容確認する）10.2.0.3 から使用可能 |

### ＞ OCR 構成ファイル

・OCR プライマリファイルと、OCR ミラーファイルの 2 つに多重化して保管可能

・現在の OCR ファイル構成は、/etc/oracle/ocr.loc で確認可能（ocrmirrorconfig_loc の文字列があればミラー）

・CRS スタックが起動すると OCR 構成ファイルが読み込まれ、
OCR ファイルに格納された投票ディスクの場所から投票ディスクをオンラインにする

・OCR ファイルの内容の整合が取れているかは、ocrcheck コマンドで確認する

### ＞ OCR ファイルの多重化

・root ユーザーにて ocrconfig コマンドにてディレクトリの変更やミラーファイル追加を行う

・OCR プライマリファイルを削除すると、ミラーファイルがプライマリになる

・CRS スタックは全てのノードで起動したままオンラインで作業を実施する

・変更時に CRS スタックが停止しているノードが存在した場合は、-repair オプションで修復させる

```
-- OCR プライマリファイルのディレクトリ変更（パスを指定しない場合は削除される）
# ocrconfig -replace ocr [OCR ファイルパス]

-- OCR ミラーファイルのディレクトリ変更（存在しない場合は新規にミラーが追加される）
# ocrconfig -replace ocrmirror [OCR ファイルパス]

-- OCR ファイルの不整合を修正する（CRS スタックを停止した後に実行）
# ocrconfig -repair ocr OCR ファイルパス
# ocrconfig -repair ocrmirror OCR ファイルパス
```

## OCR ファイルのバックアップ/リカバリ

・全ての OCR ファイルが破損すると CRSD が停止し、クラスタリソースの監視ができなくなる

・OCR ファイルは CRSD によって自動的にバックアップされている

### > OCR ファイルの物理バックアップ

・OCR ファイルマスターノードの $ORA_CRS_HOME/cdata/ｸﾗｽﾀ名 に保存される

・4 時間ごとに 3 世代バックアップ（backup00～02.ocr）

・日次バックアップ（毎日最初のバックアップ）として最新 2 ファイル（day_.ocr、day.ocr）

・週次バックアップ（CRSD が最初に起動したあとの最初のバックアップ）として最新 2 ファイル（week_.ocr、week.ocr）

### ocrconfig コマンド

| オプション | 説明 |
|---|---|
| -showbackup | OCR ファイルの物理バックアップ保存ノードと、保存場所の確認 |
| -backuploc dir | バックアップ保存場所の変更（自動バックアップされたファイルのみ確認可能） |
| -export file | OCR ファイルをバックアップする（論理バックアップ） |
| -import file | export オプションのバックアップ OCR ファイルからリストア |
| -restore file | 物理バックアップ（自動でバックアップ）の OCR ファイルをリストアする |

### > OCR のリカバリ

```
-- 物理バックアップを特定する
# ocrconfig -showbackup

-- バックアップファイル内容を確認する（10.2.0.3 以降で確認可能）
# ocrconfig -backupfile OCR ﾊﾞｯｸｱｯﾌﾟ ﾌｧｲﾙ

-- 全てのノードで CRS スタックを停止する
# crsctl stop crs

-- 物理バックアップ（自動バックアップ）をリストアする
# ocrconfig -restore OCR ﾊﾞｯｸｱｯﾌﾟ ﾌｧｲﾙ
-- 論理バックアップ（export コマンドで取得）からリストアする場合は以下のコマンド
# ocrconfig -import OCR ﾊﾞｯｸｱｯﾌﾟ ﾌｧｲﾙ

-- 全てのノードで CRS スタックを起動する
# crsctl start crs

-- OCR ファイルの整合性を確認する（クラスタ検証ユーティリティの使用）
$ cluvfy comp ocr -n all
```

## ■Oracle Clusterware コンポーネント診断

### クラスタ検証ユーティリティ

・RAC 環境のためにクラスタが適切に構成されているかを検証するツール

・<Clusterware メディア>/cluvfy/runcluvfy.sh　で実行

・Clusterware インストール後は、$ORA_CRS_HOME/bin/cluvfy コマンドも可能

### > クラスタ検証ステージ

システム設定内容、インストール、DB 作成や構成変更のステップを正常実行するための準備状況をテストする

cluvfy　stage　{ -pre | -post }　ｽﾃｰｼﾞ名　ｵﾌﾟｼｮﾝ　[-verbose]

| ｽﾃｰｼﾞ名 | チェック項目 | 事前 | 事後 |
|---|---|---|---|
| hwos | HW と OS | - | ○ |
| cfs | CFS（クラスタファイルシステム）設定 | ○ | ○ |
| crsinst | CRS インストール | ○ | ○ |
| dbinst | Clusterware が正常にインストールされているか | ○ | - |
| dbcfg | データベース作成準備が整っているか | ○ | - |

cluvfy stage -list で使用可能なステージ名

cluvfy stage -help で使用可能なオプション

（例）cluvfy stage -post crsinst -n orac1,orac2

### > クラスタ検証コンポーネント

各クラスタのコンポーネントをチェックし、その状態を判断する

cluvfy　comp　ｺﾝﾎﾟｰﾈﾝﾄ名　ｵﾌﾟｼｮﾝ　[-verbose]

| ｺﾝﾎﾟｰﾈﾝﾄ | チェック項目 | 使用例 |
|---|---|---|
| nodereach | 特定ノードから別ノードへの到達可能性 | cluvfy comp nodereach -n sv1 -srcnode sv2 |
| nodecon | 全 IF や特定 IF(-i IF 名)を使用したノード間の接続性 | cluvfy comp nodecon -n sv1,sv2 -i eth0 |
| cfs | OCFS でフォーマット済の CFS 整合性 | cluvfy comp cfs -f /ocfs -n all |
| ssa | 共有記憶域アクセス | cluvfy comp ssa -n all -s /dev/sda |
| space | 空き領域確認 | cluvfy comp space -n all -l /u01 -z 5G |
| sys | 最小システム要件 | cluvfy comp sys -n all -p { crs \| database } |
| clu | クラスタ整合性（クラスタ構成ビューが同じか） | cluvfy comp clu -n all |
| clumgr | クラスタマネージャに含まれる個々のコンポーネント | cluvfy comp clumgr -n all |
| ocr | OCR ファイル整合性（リストアが可能か） | cluvfy comp ocr -n all |
| crs | CRS スタック（CSSD、EVMD、CRSD）起動状況 | cluvfy comp crs -n all |
| nodeapp | ノードアプリケーション（VIP、ONS、GSD）存在 | cluvfy comp nodeapp -n all |
| admprv | 管理権限<br>-o user_equiv　ユーザー等価<br>-o crs_inst　Cluster インストール権限<br>-o db_inst　RAC ソフトウェアインストール権限<br>-o db_config　DB 構成用の権限 | cluvfy comp admprv -n all -o user_equiv -sshonly |
| peer | プロパティ（メモリ領域、ｶｰﾈﾙﾊﾞｰｼﾞｮﾝ等）をピアと比較 | cluvfy comp peer -refnode sv2 -n sv1 |

## Oracle Clusterware の診断

・ログは、$ORA_CRS_HOME/log/ホスト名　と　$ORACLE_HOME/log/ホスト名　に格納（DB は /racg と/client のみ）

| ﾃﾞｨﾚｸﾄﾘ | ファイル | 説明 | CRS | DB |
|---|---|---|---|---|
| . | alert ﾎｽﾄ名.log | アラートログ | ○ | - |
| /crsd/ | crsd.log | CRSD のログ。10MB でローテーション（crsd.l01～10 で、10 世代保存） | ○ | - |
| /cssd/ | ocssd.log | CSSD のログ。20MB でローテーション（ocssd.01～05 で、5 世代保存） | ○ | - |
| /evmd/ | evmd.log | EVMD のログ。500KB でローテーション（<ホスト名>_evmlog_<日付>_1、2 の名前で保存）<br>$ORA_CRS_HOME/evm/logn にもあるが、バイナリの為、emvshow コマンドを使用<br>時間ごとに確認：emvshow -t "[@timestamp]@@" ログ名<br>イベント属性毎：emvshow -D ログ名 | ○ | - |
| /racg/ | vip.log など | VIP、ONS などのリソースログ | ○ | ○ |
| /client/ | css.log など | サーバー制御ユーティリティ(srvctl)や OCR 管理(ocrdump、ocrconfig、ocrcheck) | ○ | ○ |

### ＞ 診断情報収集スクリプト

・root ユーザーにて実行（$ORA_CRS_HOME、$ORA_BASE、$ORACLE_HOME の環境変数を設定する）

・$ORA_CRS_HOME/bin/**diagcollection.pl** スクリプトを実行することで各種ログを一括収集する

| オプション | 説明 | |
|---|---|---|
| --collect | ログ情報の収集 | |
| | --all | デフォルト。すべてのログを収集 |
| | --crs | CRS ホームの Oracle Clusterware ログ、OCR、CORE ファイルを収集 |
| | --oh | RAC ホームの Oracle Clusterware ログを収集 |
| | --nocore | コアファイルを含めない |
| --clean | 現行ディレクトリに存在するファイルのクリーンアップを行う<br>現行ディレクトリに同名ファイルが存在すると上書き確認を聞いてくるので事前に削除する | |
| --coreanalyze | 生成されたファイルで検出されたコアファイルのみテキストファイルに抽出 | |
| --help | 使用方法を表示 | |

## 各種デバッグ

### ＞ Oracle Clusterware のデバッグ有効化

・CRS スタックは実行時に内部モジュールを使用しており、内部モジュールに対してデバッグを有効化することが可能

・root ユーザーにて実行（$ORA_CRS_HOME、$ORA_BASE、$ORACLE_HOME の環境変数を設定する）

```
-- デバッグ可能なモジュール一覧確認
$ crsctl lsmodules { css | crs | evm }

-- デバッグの有効化 レベルは 0(無効)～5(最大)　※次回起動時も有効
# crsctl debug log { css | crs | evm } ﾓｼﾞｭｰﾙ名:ﾚﾍﾞﾙ [,ﾓｼﾞｭｰﾙ名:ﾚﾍﾞﾙ]

-- CRS スタックの状態情報をダンプ（各 CRS スタックのログファイル（ocssd.log、crsd.log、evmd.log）に出力される）
# crsctl debug statedump { css | crs | evm }

-- CRS スタックのメモリ内トレースキャッシュをダンプ
# crsctl debug trace { css | crs | evm }
```

CRS スタックモジュール一覧

| サービス | モジュール | 説明 | | |
|---|---|---|---|---|
| CSS | CSSD | CSS デーモン | | |
| CRS | CRSUI | ユーザーインターフェース | CRSRES | リソースモジュール |
| | CRSCOMM | 通信モジュール | CRSOCR | OCR インターフェース |
| | CRSRTI | リソース管理モジュール | CRSTIMER | CRS 関連タイマー |
| | CRSMAIN | メインモジュール | CRSEVT | イベントインターフェース |
| | CRSPLACE | 配置モジュール | CRSD | CRS デーモン |
| | CRSAPP | アプリケーションモジュール | CLUCLS | クラスタ（CSS）情報 |
| EVM | EVMD | EVM デーモン | EVMAPP | アプリケーションモジュール |
| | EVMMAIN | メインモジュール | EVEMAGENT | エージェントモジュール |
| | EVMCOMM | 通信モジュール | CRSOCR | OCR インターフェース |
| | EVMEVT | イベントインターフェース | CLUCLS | クラスタ（CSS）情報 |

### > リソースデバッグの有効化

・Clusterware によって管理されるクラスタリソース(crs_stat で確認)のデバッグを有効化することが可能

・デバッグを有効化するには、crsctl debug log res "ﾘｿｰｽ名:1" ※無効化は 0 を指定

これは指定したリソースプロファイルの「USR_ORA_DEBUG」属性を「1」に変更するのと同じ

・全てのリソースを有効化するには、$ORA_CRS_HOME/bin/racgwrap に _USR_ORA_DEBUG=1 を設定する

```
-- VIP リソースのデバッグ有効化
# crsctl debug log res "ora.orac1.vip:1"

-- ログの確認
# cat $ORA_CRS_HOME/log/orac1/racg/ora.orac1.vip.log
```

### > OCR 関連ツール用のデバッグフラグの設定

・$ORA_CRS_HOME/srvm/admin/ocrlog.ini の mesg_logging_level でロギングされる情報を変更できる

・デフォルトは「0」でエラーのみ記録。「3」または「5」に設定することで詳細なログ取得が可能

・comploglvl（ロギングレベル）、comptrclvl（トレースレベル）パラメータではコンポーネントレベルの変更が可能

・OCR 操作ツール（ocrdump、ocrconfig、ocrcheck、srvctl）のログは$ORA_CRS_HOME/log/ﾎｽﾄ/client にある

### > Java ベースのツールに対するトレースの有効化

・DBCA と DBUA はデフォルトでトレースが有効化されている

・SRVM_TRACE 環境変数を TRUE にすることで、以下のツールのトレースが有効になる

| ツール | ログディレクトリ | ツール | ログディレクトリ |
|---|---|---|---|
| DBCA | $ORACLE_HOME/cfgtoollogs/dbca | NETCA | $ORACLE_HOME/cfgtoollogs/netca |
| DBUA | $ORACLE_HOME/cfgtoollogs/dbua | VIPCA | $ORACLE_HOME/cfgtoollogs/vipca |
| SRVCTL | 標準出力 | CVU | $ORA_CRS_HOME/cv/log |

## ■ノード追加と削除

### 新規ノードの追加

**› ノード追加に伴う共有ストレージ領域**

| 必要な領域 | 説明 |
|---|---|
| REDO ロググループ | インスタンス毎に最低２つの REDO ロググループが必要 |
| UNDO 表領域 | インスタンス毎に１つ必要 |
| SYSAUX 表領域 | 合計サイズが、300MB + ( 250MB x インスタンス数)になるように領域を拡張する |

**› ノード追加ステップ**

・クローニング、Enterprise Manager Grid Control、OUI 直接使用の 3 パターンがある

### クローニング

① 新規ノードで既存と同じ Oracle ユーザーを作成する

② 既存ノードの RAC ホーム、CRS ホームのファイル全てを、新規ノードにコピーする

③ RAC と CRS を OUI のクローンモードで実行する（clone.pl）

④ NETCA を使用し新規ノードでリスナーを作成

⑤ DBCA を使用して新規ノードにインスタンスを作成する

### Enterprise Manager Grid Control

① 新規ノードにエージェントをインストールする（エージェントのインストールウィザードの使用）

② RAC と CRS のクローニング実行（Oracle ホームのクローニングウィザードの使用）

※Grid Control の「デプロイ」タブに各ウィザードがある

### OUI 直接使用

**① 新規ノードの準備**

OS、ハードウェア、OS ユーザーの準備を行う

**② CRS ホーム追加**

既存ノードで $ORA_CRS_HOME/oui/bin/addNode.sh を実行すると OUI が起動し新規ノードを指定することが可能

OUI によって新規ノードへのコピー完了後、構成スクリプトの実行を root ユーザーにて行う

| スクリプト | 実行 | 説明 |
|---|---|---|
| rootaddnode.sh | 既存 | 新規ノードに OCR 構成ファイルが送信され新規ノードにノードアプリケーションが追加される |
| root.sh | 新規 | 新規ノード側で CRS スタックが起動され、VIPCA によってノードアプリケーション構成が行われる<br>※パブリックネットワークにプライベートクラスを指定しているとエラーになるが、<br>ノードアプリケーションの追加は完了している為、手動で VIPCA を実行する必要はない |

**③ 新しいノードで ONS の構成**

$ORA_CRS_HOME/opmn/conf/ons.config で リモートポート番号を確認する

$ORA_CRS_HOME/bin/racgons add_config ｻｰﾊﾞｰ名:ﾎﾟｰﾄ番号 で ONS 追加

ocrdump -stdout -keyname DATABASE.ONS_HOSTS で追加された ONS 構成情報を確認する

**④ ASM ホームの追加**（ASM ホームが存在する場合に実行するステップ）

既存ノード側にて、$ORACLE_HOME 環境変数に ASM ホームのディレクトリを指定する

既存ノード側にて、$ORACLE_HOME/oui/bin/addNode.sh を実行する（OUI が起動し新規ノードの指定が可能になる）

OUI によって新規ノードのコピーが完了後、新規ノードにて root.sh を実行する

※この時点では ASM インスタンスは追加されない（DB インスタンス追加時に新規ノードに ASM インスタンスも追加される）

**⑤ RAC ホーム（データベースソフトウェア）の追加**

$ORACLE_HOME 環境変数を DB ホームのディレクトリに指定後、④ASM ホームの追加と同じことを行う

※DB インスタンスの追加は DBCA で行う

**⑥ リスナー追加**

既存ノードが ASM ホームからリスナーを構成している場合は、新規ノードも ASM ホームからリスナーを構成(※)する

※ORACLE_HOME 環境変数を ASM ホームにする

新規ノード側で $ORACLE_HOME/bin/netca を実行する

リスナー名は「LISTENER_ホスト名」

netca を使用してリスナー構成することで、クラスタリソースとしてのリスナーも追加される

**⑦ DB インスタンスの追加**

既存ノードで DBCA を実行し「インスタンスの管理」にて追加する

[インスタンスの名前付けとノードの選択]画面で追加するノードを選択し、インスタンス名を指定する

[サマリー]画面では ASM を記憶域に使用している場合、新規ノードに ASM インスタンスを追加する必要があるという

ポップアップが出力されるので「はい」応答する

## ノードの削除

**① DB インスタンスの削除**

残存ノードで DBCA を実行し「インスタンス管理」にて削除を行う

インスタンスに対応付けた REDO ログ、UNDO 表領域、クラスタリソースが削除される

**② ASM インスタンスの削除**

```
-- ASM インスタンスの削除
$ srvctl stop asm -n ﾉｰﾄﾞ名

-- ASM インスタンスリソースの削除
$ srvctl remove asm -n ﾉｰﾄﾞ名

-- ASM 関連ファイルの削除
$ export ORACLE_HOME=$ORACLE_BASE/product/10.2.0/asm_1
$ rm -rf $ORACLE_HOME/dbs/*ASM*
$ rm -rf $ORACLE_BASE/admin/+ASM

-- /etc/oratab ファイルから次のようなエントリを削除
+ASM3:/u01/app/oracle/product/10.2.0/asm_1:N
```

**③ リスナーの削除**

残存ノードで NETCA を実行する（ASM ホームからリスナーを構成している場合は ASM ホームからリスナーを削除する）

NETCA で削除することでクラスタリソースも削除される

```
$ export ORACLE_HOME=$ORACLE_BASE/product/10.2.0/asm_1
$ $ORACLE_HOME/bin/netca
```

### ④ RAC ホームの削除

削除ノードのインベントリを更新後、OUI にて RAC ホームを削除する

```
-- 削除ノードでインベントリの更新（削除ノード以外のノードに影響を与えずソフトウェアの削除を可能にする）
$ /RAC ﾎｰﾑ/oui/bin/runInstaller -updateNodeList ORACLE_HOME=RAC ﾎｰﾑ "CLUSTER_NODES=削除ﾉｰﾄﾞ名" -local

-- 削除ノードで RAC ホームの削除（引数なしで実行）GUI で削除対象に RAC ホームを選択する
$ /RAC ﾎｰﾑ/oui/bin/runInstaller

-- 残存ノードでインベントリ更新（残存ノードリストはカンマ区切りで複数指定）
$ /RAC ﾎｰﾑ/oui/bin/runInstaller -updateNodeList ORACLE_HOME=RAC ﾎｰﾑ "CLUSTER_NODES=残存ﾉｰﾄﾞﾘｽﾄ"
```

### ⑤ ASM ホームの削除（ASM ホームが存在する場合に実行するステップ）

④ RAC ホームの削除と同様の手順（RAC ホームが ASM ホームになる）で GUI の削除対象で ASM ホームを選択する

### ⑥ ONS 構成からノード削除

```
-- 使用するリモートポートの確認（remoteport=XXXX）
$ cat $ORA_CRS_HOME/opmn/conf/ons.config

-- ONS 構成情報削除
$ $ORA_CRS_HOME/bin/racgones remove_config ﾉｰﾄﾞ名:remoteport 番号
```

### ⑦ CRS ホームの削除

```
-- 削除ノードで OCR から情報を削除する
# $ORA_CRS_HOME/install/rootdelete.sh

-- 残存ノードで OCR から情報を削除する（ノード番号は$ORA_CRS_HOME/bin/olsnodes -n コマンドで確認）
# $ORA_CRS_HOME/install/rootdelete.sh 削除ﾉｰﾄﾞ名,ﾉｰﾄﾞ番号

-- 削除ノードでインベントリ更新
$ /CRS ﾎｰﾑ/oui/bin/runInstaller -updateNodeList ORACLE_HOME=CRS ﾎｰﾑ "CLUSTER_NODE=削除ﾉｰﾄﾞ" CRS=TRUE -local

-- 削除ノードで CRS ホームの削除（製品削除で CRS ホームにチェックを入れる）
$ /CRS ﾎｰﾑ/oui/bin/runInstaller

-- 残存ノードでインベントリ更新
$ /CRS ﾎｰﾑ/oui/bin/runInstaller -updateNodeList ORACLE_HOME=CRS ﾎｰﾑ "CLUSTER_NODE=残存ﾉｰﾄﾘｽﾄﾞ" CRS=TRUE
```