Oracle® Real Application Clusters Real Application Clusters 管理およびデプロイメント・ガイド 19c

F16187-08(原本部品番号:E95728-09)

2023年3月

タイトルおよび著作権情報

Oracle Real Application Clusters Real Application Clusters管理およびデプロイメント・ガイド, 19c

F16187-08

原本著者: Subhash Chandra

原本協力著者: Janet Stern

原本協力者: Troy Anthony、Lance Ashdown、Ram Avudaiappan、Prasad Bagal、Mark Bauer、Anand Beldalker、Eric Belden、Gajanan Bhat、David Brower、George Claborn、Maria Colgan、Carol Colrain、Jonathan Creighton、Rajesh Dasari、Mark Dilman、Richard Frank、GP Prabhaker Gongloor、Wei Hu、Yong Hu、Dominique Jeunot、Sameer Joshi、Raj K.Kammend、Ankita Khandelwal、Sana Karam、Roland Knapp、Karen Li、Barb Lundhild、Venkat Maddali、Bill Manry、John McHugh、Saar Maoz、Matthew Mckerley、Markus Michalewicz、Anil Nair、Philip Newlan、Michael Nowak、Muthu Olagappan、Bharat Paliwal、Hanlin Qian、Hairong Qin、Mark Ramacher、Sampath Ravindhran、Kevin Reardon、Kathy Rich、Dipak Saggi、Daniel Semler、Ara Shakian、Cathy Shea、Khethavath P.Singh、Kesavan Srinivasan、Leo Tominna、Peter Wahl、Tak Wang、Richard Wessman、Douglas Williams、Mike Zampiceni、Michael Zoll

目次

- srvctl disable listener
- srvctl enable listener
- srvctl getenv listener
- srvctl modify listener
- srvctl predict listener
- srvctl remove listener
- srvctl setenv listener
- srvctl start listener
- srvctl status listener
- srvctl stop listener
- srvctl unsetenv listener
- srvctl update listener
- srvctl add network
- srvctl config network
- srvctl modify network
- srvctl predict network
- srvctl remove network
- srvctl add nodeapps
- srvctl config nodeapps
- srvctl disable nodeapps
- srvctl enable nodeapps
- srvctl getenv nodeapps
- srvctl modify nodeapps
- srvctl remove nodeapps
- srvctl setenv nodeapps
- srvctl start nodeapps
- srvctl status nodeapps
- srvctl stop nodeapps
- srvctl unsetenv nodeapps
- srvctl add ons
- srvctl config ons
- srvctl disable ons
- srvctl enable ons
- srvctl modify ons
- srvctl remove ons
- srvctl start ons
- srvctl status ons
- srvctl stop ons
- srvctl add scan
- srvctl config scan
- srvctl disable scan
- srvctl enable scan
- srvctl modify scan

- srvctl predict scan
- srvctl relocate scan
- srvctl remove scan
- srvctl start scan
- srvctl status scan
- srvctl stop scan
- srvctl add scan_listener
- srvctl config scan_listener
- srvctl disable scan_listener
- srvctl enable scan_listener
- srvctl modify scan_listener
- srvctl predict scan_listener
- srvctl relocate scan_listener
- srvctl remove scan_listener
- srvctl start scan_listener
- srvctl status scan_listener
- srvctl stop scan_listener
- srvctl update scan_listener
- srvctl relocate server
- srvctl status server
- srvctl add service
- srvctl config service
- srvctl disable service
- srvctl enable service
- srvctl modify service
- srvctl predict service
- srvctl relocate service
- srvctl remove service
- srvctl start service
- srvctl status service
- srvctl stop service
- srvctl add srvpool
- srvctl config srvpool
- srvctl modify srvpool
- srvctl remove srvpool
- srvctl status srvpool
- srvctl add vip
- srvctl config vip
- srvctl disable vip
- srvctl enable vip
- srvctl getenv vip
- srvctl modify vip
- srvctl predict vip

表一覧

はじめに

*Oracle Real Application Clusters*管理およびデプロイメント・ガイドでは、Oracle Real Application Clusters (Oracle RAC)のアーキテクチャについて説明します。

Oracle Real Application Clusters One Node (Oracle RAC One Node)を含め、製品の概要を示します。また、Oracle RACの管理およびデプロイメントについても説明します。

このマニュアルの情報は、特に指定がないかぎり、すべてのプラットフォーム上で動作するOracle RACに適用されます。また、このマニュアルの内容は、他のOracleドキュメントで説明している非クラスタOracle Databaseの管理およびデプロイメントに関する項目の説明を補足するものです。必要に応じて、プラットフォーム固有のドキュメントを参照しています。

対象読者

このマニュアルは、次のタスクを実行するデータベース管理者、ネットワーク管理者およびシステム管理者を対象としています。

- Oracle RACデータベースのインストールおよび構成
- Oracle RACデータベースの管理
- Oracle RACを使用するクラスタおよびネットワークの管理およびトラブルシューティング

ドキュメントのアクセシビリティ

Oracleのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートを購入したオラクル社のお客様は、My Oracle Supportを介して電子的なサポートにアクセスできます。詳細情報は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

関連ドキュメント

このマニュアル『Oracle Real Application Clusters管理およびデプロイメント・ガイド』では、Oracle RAC固有の管理およびアプリケーションのデプロイメントについて説明します。このマニュアルでは、Oracle Clusterwareについて理解していることを前提としています。

詳細は、次のOracleドキュメントを参照してください。

- Oracle Clusterware管理およびデプロイメント・ガイド

  このマニュアルでは、投票ディスクなどのOracle ClusterwareコンポーネントおよびOracle Cluster Registry(OCR)について説明します。
- プラットフォーム固有のOracle ClusterwareおよびOracle RACのインストレーション・ガイド

Oracle Databaseのプラットフォーム固有の各インストール・メディアには、プラットフォーム固有のOracle ClusterwareおよびOracle RACのインストールおよび構成ガイド(HTML形式およびPDF形式)が含まれています。このマニュアルには、Oracle ClusterwareおよびOracle RACが動作するUNIX、LinuxおよびWindowsベースの様々なプラットフォームに対するインストール前、インストールおよびインストール後の情報が記載されています。

- Oracle Database 2日でデータベース管理者
- Oracle Database管理者ガイド
- Oracle Database Net Services管理者ガイド
- Oracle Databaseプラットフォーム・ガイドfor Microsoft Windows
- Oracle Database管理者リファレンス for Linux and UNIX-Based Operating Systems
- 『Oracle Database管理者リファレンス11gリリース1 (11.1) for Linux and UNIX-Based Operating Systems』(AIX Systems、HP-UX、LinuxおよびSolarisオペレーティング・システム(SPARC))

ノート:

今回のリリースの追加情報は、Oracle Database 12c の README またはリリース・ノートを参照してください。今回のリリース用のこれらのドキュメントが存在する場合、Oracle 製品のインストール・メディアに収録されています。

データベース・エラー・メッセージの説明は、オンラインまたはTahitiドキュメント検索で参照できます。

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ドキュメントのタイトル、強調またはユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

『Oracle Real Application Clusters管理およびデプロイメント・ガイド』のこのリリースの変更内容

内容は次のとおりです。

- Oracle Real Application Clusters リリース19cでの変更内容
- Oracle Real Application Clusters リリース18c (バージョン18.1)での変更内容
- Oracle Real Application Clusters 12cリリース2 (12.2)での変更内容
- Oracle Real Application Clusters 12cリリース1 (12.1)での変更内容

Oracle Real Application Clusters リリース19cでの変更内容

次のトピックでは、Oracle Database 19cの*Oracle Real Application Clusters*管理およびデプロイメント・ガイドにおける新機能、非推奨となった機能、またはサポート対象外となった機能を示します。

新機能

このリリースの新機能は次のとおりです。

- クライアントをルーティングするための関連付けタグ

  Oracle Databaseリリース19.3以降では、colocation_tagを使用して接続文字列のCONNECT_DATAの一部として文字列値を指定できます。可能な場合、同じコロケーション・タグを持つクライアントは、所与のサービスを提供する同じインスタンスにルーティングされます。接続時に関連付けを使用できない場合、タグは無視され、接続はタグがない場合と同様に使用可能なインスタンスにルーティングされます。

  同じインスタンスのセッションを関連付けると、インスタンス間通信が減少するため、同じインスタンスで実行することでメリットがあるワークロードのパフォーマンスが向上します。

  サービスの関連付けを参照してください。

- 動的サービス・フォールバック・オプション

  Oracle Databaseリリース19.3以降では、preferredおよびavailable設定を使用して配置された動的データベース・サービスの場合、使用可能なインスタンスにフェイルオーバーした後、優先インスタンスが使用可能になったときに、このサービスがそのインスタンスに戻るように指定できるようになりました。使用可能な優先インスタンスがない場合、サービスは使用可能インスタンスにフェイルオーバーします。「動的サービス・フォールバック・オプション」を使用すると、動的なデータベース・サービスの配置をより細かく制御し、特定のサービスを可能なかぎり優先インスタンスで使用可能にすることができます。

  インスタンスのプリファレンスを参照してください。

- RACリソース・ランタイム管理

  Oracle Database SGAランタイム管理では、実行時にSGAを拡張できます。Oracle Database 19.3以降では、Oracle RACのリソース・ランタイム管理を使用すると、データベースの起動時にのみ割り当てられていたリソースの自動的な調整および実行時の調整が可能になります。これにより、より効率的なリソース割当てが可能になります。

自動メモリー管理の詳細は、*Oracle Database*管理者ガイドの自動メモリー管理についてを参照してください。

- グリッド・インフラストラクチャ管理リポジトリのオプションのインストール

  Oracle Grid Infrastructure 19c以上では、グリッド・インフラストラクチャ管理リポジトリ(GIMR)は、Oracleスタンドアロン・クラスタの新規インストールでオプションです。Oracleドメイン・サービス・クラスタでは、GIMRをサービス・コンポーネントとしてインストールする必要があります。

  GIMRに含まれるデータは、適用された機械学習に基づく予防診断の基礎となり、Oracle Real Application Clusters (Oracle RAC)データベースの可用性の向上に役立ちます。GIMRをオプションでインストールすると、特にテスト・システムや開発システムのインストール時に、ストレージ領域の管理および高速デプロイメントをより柔軟に行うことができます。

  *Oracle Grid Infrastructure*インストレーションおよびアップグレード・ガイド *for Linux*のグリッド・インフラストラクチャ管理リポジトリについてを参照してください。

OCRおよび投票ディスクの直接ファイル配置の再サポート

Oracle Grid Infrastructure 19c以上では、共有ファイル・システム上のOCRおよび投票ファイルの直接配置のサポート終了は、Oracleスタンドアロン・クラスタに対して廃棄されます。

Oracle Grid Infrastructure 12cリリース2 (12.2)では、直接、共有ファイル・システム上でOracle Grid InfrastructureのOracle Cluster Registry (OCR)および投票ファイルの配置がサポートされなくなることが発表されました。このサポート終了は現在破棄されました。Oracle Grid Infrastructure 19c (19.3)以上では、Oracleスタンドアロン・クラスタとともに、OCRおよび投票ディスク・ファイルを共有ファイル・システムに直接配置できます。ただし、Oracleドメイン・サービス・クラスタの場合、Oracle Automatic Storage Management (Oracle ASM)で管理される定数障害グループにOCRおよび投票ファイルを引き続き配置する必要があります。

Oracle Real Application Clusters 19cで非推奨となった機能

次の機能は、Oracle Real Application Clusters 19cでは非推奨であり、将来のリリースではサポートされなくなる可能性があります。

Addnodeスクリプトの非推奨

addnodeスクリプトは、Oracle Grid Infrastructure 19cでは非推奨です。クラスタにノードを追加する機能は、インストーラ・ウィザードで使用できます。

addnodeスクリプトは、今後のリリースで削除できます。addnodeスクリプト(addnode. shまたはaddnode. bat)を使用するかわりに、インストーラ・ウィザードを使用してノードを追加します。インストーラ・ウィザードでは、addnodeスクリプトを超える機能が多数追加されています。インストーラ・ウィザードを使用すると、すべてのソフトウェア・ライフサイクル操作を1つのツールに統合して管理が簡素化されます。

clone.plスクリプトの非推奨

clone. plスクリプトは、今後のリリースで削除できます。clone. plスクリプトを使用するかわりに、インストーラ・ウィザードを使用して、抽出したゴールド・イメージをホームとしてインストールすることをお薦めします。

ベンダー・クラスタウェアのOracle Clusterwareとの統合の非推奨

Oracle Clusterware 19c (19.5)以降、ベンダーまたはサード・パーティのクラスタウェアとOracle Clusterwareの統合は非推奨になっています。

Oracle Clusterwareとベンダー・クラスタウェアの統合は非推奨であり、将来のリリースでサポートされなくなる可能性があります。特定のクラスタ機能を非推奨にし、制限付きの採用とすることにより、オラクル社では、すべての機能の中核となるスケーリング、可用性および管理性の向上に注力できます。異なるクラスタ・ソリューション間の統合がない場合、システムではクラスタ・ソリューションの競合問題が発生します。独立したクラスタ・ソリューションの場合は、特定の障害が発生したときに、どの修正処置が必要かについて個別に判断できます。競合を回避するには、常に1つのクラスタ・ソリューションのみをアクティブにする必要があります。そのため、次回のソフトウェアまたはハードウェアのアップグレードは、ベンダーのクラスタ・ソリューションからの移行に合わせることをお薦めします。

Oracle Real Application Clusters 19cでサポート対象外となった機能

関連項目:

詳細は、『Oracle Databaseアップグレード・ガイド』を参照してください。

- Standard Edition 2 (SE2)データベース・エディションでのOracle Real Application Clustersのサポート終了

  Oracle Database 19c以上では、Oracle Real Application Clusters (Oracle RAC)はOracle Database Standard Edition 2 (SE2)でサポートされなくなりました。

- Oracle Streamsのサポート終了

  Oracle Database 19c (19.1)からは、Oracle Streamsがサポート対象外となります。Oracle GoldenGateがOracle Database用の代替ソリューションです。

  Oracle Database Advanced Queuingは非推奨ではなく、Oracle Database 19cでも完全にサポートされます。Oracle Streamsでは、Oracle Database 12c (12.1)以降で追加された機能、たとえばマルチテナント・アーキテクチャ、LONG VARCHAR、その他の新機能などがサポートされていませんでした。Oracle Streamsレプリケーション機能はGoldenGateによって置き換えられます。

Oracle Real Application Clusters リリース18c (バージョン18.1)での変更内容

Oracle Real Application Clusters 18c (18.1)の『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』における新しい機能は、次のとおりです。

連続したアプリケーションの可用性

継続的なアプリケーション可用性が計画メンテナンスおよび透過的アプリケーション・コンティニュイティの組合せによって実現されています。データベースの計画メンテナンスには、アプリケーションがアクティビティを認識することなく、計画メンテナンスの開始前のデータベース・セッションの排出および移行が含まれています。透過的データベース・セッションの排出が発生するのは、アプリケーション・コンティニュイティが有効化されていて、安全なトランザクション内のポイントにある別のデータベース・インスタンスにセッショ

ンをフェイルオーバーさせる場合(アプリケーションが接続テストを発行するときまたはトランザクション内のリクエスト境界など)です。

透過的アプリケーション・コンティニュイティでは、リカバリ可能な停止の後にデータベース・セッションがリカバリできるように、透過的にセッションおよびトランザクションの状態を追跡および記録します。アプリケーションの知識やコードの変更が必要なくても安全にこれが実行され、アプリケーション・コンティニュイティがアプリケーションの標準になります。透過性は、アプリケーションがユーザー・コールを発行するときにセッション状態の使用状況を分類する新しい状態追跡インフラストラクチャを使用して実現されます。

透過的アプリケーション・コンティニュイティを参照してください。

Oracle RACシャーディング

Oracle RACシャーディングでは、表パーティションをOracle RACインスタンスにまとめ、パーティション化キーを指定するデータベース・リクエストを、対応するパーティションを論理的に保持するインスタンスにルーティングします。これによって、より適切にキャッシュが利用され、インスタンス全体でブロックのpingが大幅に削減されます。パーティション化キーは、パフォーマンスが最も重要なリクエストにのみ追加できます。キーを指定しないリクエストは透過的に動作し、どのインスタンスにもルーティング可能です。この機能を有効化する際に、データベース・スキーマへの変更は必要ありません。Oracle RACシャーディングでは、最小限のアプリケーションの変更でパフォーマンスおよびスケーラビリティのメリットを提供します。

Oracle RACシャーディングを参照してください。

拡張性の高い順序

キーとして順序値を使用する表にデータをロードする際に、拡張性の高い順序により索引リーフ・ブロックの競合が緩和されます。

関連項目:

順序をスケーラブルにする方法

Oracle Real Application Clusters 12cリリース2 (12.2)での変更内容

Oracle Real Application Clusters 12cリリース2 (12.2)の『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』における新しい機能は、次のとおりです。

SCANリスナーによるHTTPプロトコルのサポート

現在、SCANリスナーは、接続要求を受信すると、同じノードに配置されているハンドラ間でロード・バランシングを行います。次に、ノード上で最も負荷の低いハンドラにその接続を渡します。このリリースでは、SCANリスナーはHTTPプロトコルを認識するため、HTTPクライアントを適切なハンドラ(クラスタ内のSCANリスナーが存在するノードとは別のノードに存在する可能性がある)にリダイレクトできます。

プライベート・ネットワークでのOracle Real Application Clustersに対するIPv6のサポート

プライベート・ネットワーク上でIPv4またはIPv6ベースのIPアドレスを使用するようにクラスタ・ノードを構成でき、1つのクラスタに対して複数のプライベート・ネットワークを使用できます。

関連項目:

Oracle Clusterware管理およびデプロイメント・ガイド

管理者管理データベースの完全なサポートのためのOracle Database QoS Managementの拡張

このリリースでは、管理モードもサポートすることで、Oracle Database Quality of Service Management (Oracle Database QoS Management)の完全なサポートを利用できます。Oracleは、データベース内で実行するパフォーマンス・クラスのCPU共有を調整することで、管理者管理Oracle RACデータベース内のスキーマ統合をサポートしています。さらに、同じ物理サーバーでホストされているデータベースごとのCPU数を調整することで、データベース統合もサポートされています。

管理者管理データベースはサーバー・プールで実行しないため、サーバー・プールのサイズを変更してインスタンスの数を増減する機能は、ポリシー管理データベースのデプロイメントでサポートではサポートされていますが、管理者管理データベースでは利用できません。このデプロイメントのサポートは、Oracle Enterprise Manager Cloud ControlのOracle Database QoS Managementのページに統合されています。

関連項目:

Oracle Database Quality of Service Managementユーザーズ・ガイド

Oracle Real Application Clustersリーダー・ノード

Oracle RACリーダー・ノードにより、オンライン・トランザクション処理(OLTP)ワークロードを実行する読取り/書込みインスタンスのセットと、読取り専用データベース・インスタンスのセットをクラスタ内のハブ・ノードとリーフ・ノードに割り当てることで、Oracle Flex Clusterのアーキテクチャが促進されます。このアーキテクチャでは、読取り/書込みインスタンスに対する更新は、オンライン・レポートの作成や即座の問合せに使用できるリーフ・ノード上の読取り専用インスタンスにすぐに伝播されます。

関連項目:

Oracle Clusterware管理およびデプロイメント・ガイド

サーバーの重みベースのノード削除

サーバーの重みベースのノード削除は、すべてのノードが削除に対して同じ選択を表すクラスタから、Oracle Clusterwareが特定のノードまたはノードのグループを削除する必要がある状況で、タイブレークのメカニズムとして機能します。サーバーの重みベースのノード削除メカニズムは、これらのサーバーの負荷に関する追加情報に基づいて、削除するノードまたはノードのグループを特定する際に役立ちます。2つの原則メカニズム(システム固有の自動メカニズムおよびユーザー入力ベースのメカニズム)が、それぞれのガイダンスを提供するために存在します。

関連項目:

Oracle Clusterware管理およびデプロイメント・ガイド

Oracle Real Application Clustersを管理するための義務の分離

Oracle Database 12c リリース2 (12.2)から、Oracle Databaseは、クラスタウェア・エージェントのSYSRAC管理権限を導入することにより、Oracle Real Application Clusters (Oracle RAC)を管理する際の義務のベスト・プラクティスの分離サポートを提供するようになりました。この機能により、Oracle RACに対して強力なSYSDBA管理権限を使用する必要がなくなりました。

SYSRACは、SYSDG、SYSBACKUPおよびSYSKMと同様、義務の分離の施行、および本番システム上のSYSDBAの使用への依存の軽減に役立ちます。この管理権限は、SRVCTLなどのOracle RACユーティリティのかわりに、クラスタウェア・エージェントによってデータベースに接続するためのデフォルト・モードです。

インメモリー・ファスト・スタート

インメモリー・ファスト・スタートは、インメモリー圧縮ユニットをディスクに直接格納することで、インメモリー列ストアのデータベース・オブジェクトの移入を最適化します。

関連項目:

『Oracle Database In-Memoryガイド』

Oracle Real Application Clusters 12cリリース1 (12.1)での変更内容

次に、Oracle Real Application Clusters (Oracle RAC) 12cの『Oracle Real Application Clusters管理およびデプロイメント・ガイド』での変更内容を示します。

- Oracle Real Application Clusters 12cリリース1 (12.1.0.2)での変更内容
- Oracle Real Application Clusters 12cリリース1 (12.1.0.1)での変更内容
- 非推奨となった機能
- サポート対象外となった機能

Oracle Real Application Clusters 12cリリース1 (12.1.0.2)での変更内容

このリリースの新機能は次のとおりです。

- インメモリー列ストア

  インメモリー列ストアは、表全体、表パーティション、および圧縮列形式の個々の列を格納する、SGAのオプション領域です。データベースは、SIMDベクトル処理を含む特別な手法を使用して、列データを非常に高速にスキャンします。インメモリー列ストアは、データベース・バッファ・キャッシュを置き換えるものではなく、補完するものです。

  関連項目:

  詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください

- インメモリー・トランザクション・マネージャ

  インメモリー・トランザクション・マネージャは独立したエンジンで、インメモリー列ストアへの変更を適用するトランザクションの読取り一貫性を自動的に提供します。インメモリー列ストアに存在する表およびパーティションは列形式でメモリーに格納され、行メジャー形式でデータ・ファイルおよびデータベース・バッファ・キャッシュに格納されるので、このエンジンが必要です。

  関連項目:

詳細は、*Oracle Database*管理者ガイドのインメモリー列ストアを使用した問合せパフォーマンスの向上を参照してください

- フリート・パッチ適用およびプロビジョニング

  フリート・パッチ適用およびプロビジョニングでは、事前作成済のソフトウェア・ホームのカタログに格納されたイメージに基づいてOracleホームをデプロイできます。

  関連項目:

  詳細は、『Oracle Clusterware管理およびデプロイメント・ガイド』を参照してください。

- 全データベース・インメモリー・キャッシング

  このリリースでは、データベース全体をメモリーにキャッシングできます。各インスタンスのバッファ・キャッシュ・サイズがデータベース全体のサイズよりも大きいときにこの機能を使用します。Oracle RACシステムで、正しくパーティション化されたアプリケーションでは、すべてのデータベース・インスタンスの結合バッファ・キャッシュ(インスタンス間で重複するキャッシュされたブロックを処理する追加の領域を含む)がデータベース・サイズよりも大きい場合に、この機能を使用できます。

  関連項目:

  詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください

- Oracle Database QoS管理をアクティブにする必要がないメモリー・ガード

  このリリースでは、Oracle Databaseサービスのクオリティ管理(Oracle Database QoS管理)を使用するかどうかに関係なく、メモリー・ガードがデフォルトで有効化されています。メモリー・ガードはノードのメモリー不足を検出し、既存のワークロードが縮小してメモリーが解放されるまで、新しいセッションを他のインスタンスに送ります。ノードの空きメモリーが増えると、自動的にサービスが再び新しい接続を受け入れられるようになります。

Oracle Real Application Clusters 12cリリース1 (12.1.0.1)での変更内容

このリリースの新機能は次のとおりです。

- アプリケーション・コンティニュイティ

  このリリースより前は、エンド・ユーザーから停止をマスクする場合、アプリケーション開発者は基礎となるソフトウェア、ハードウェアおよび通信レイヤーの停止を明示的に処理する必要がありました。

  Oracle Database 10gでは、高速アプリケーション通知(FAN)によって、例外条件がアプリケーションに迅速に配信されました。ただし、FANおよび以前のOracleテクノロジでは、最後のトランザクションの結果がアプリケーションに報告されず、アプリケーションの観点から進行中の要求がリカバリされませんでした。結果として、停止がマスクされず、ユーザーに不便を強い、収益が失われました。ユーザーが意図せずに重複して品物を購入したり、1つの請求書に何回も支払う可能性もあります。複雑なケースでは、引き起こされた問題に対処するために、管理者が中間層をリブートする必要がありました。

  アプリケーション・コンティニュイティは、アプリケーションに依存しない機能であり、アプリケーションの観点から不完全な要求のリカバリを試行し、システム、通信、ハードウェアの多くの障害および記憶域の停止をエンド・ユーザーからマスクします。

関連項目:

- 詳細は、『Oracle Database概要』を参照してください
- アプリケーション・コンティニュイティの確保

- Java用のトランザクション・ガード

  この機能によって、新しいアプリケーション・コンティニュイティインフラストラクチャがJavaに公開されます。次のことがサポートされます。

  - トランザクションの冪等性など、トランザクションの実行を1回以下にするためのプロトコル
  - 論理トランザクションID (LTXID)を取得するためのAPI
  - 接続またはセッションのステータスを取得するための属性

関連項目:

詳細は、『Oracle Database JDBC開発者ガイド』を参照してください

- トランザクションの冪等性

  この機能は、アプリケーションに依存しない汎用インフラストラクチャを提供し、これにより、アプリケーションの観点からの作業のリカバリを可能にし、システム、通信およびハードウェアの多くの障害をユーザーからマスクします。トランザクションの冪等性によって、トランザクションは予定どおりに、最大1回実行されるようになります。

関連項目:

詳細は、『Oracle Database開発ガイド』を参照してください

- Oracle Flex Cluster

  場合によっては何千というノードで構成される大規模なクラスタは、Oracle RACにプラットフォームを提供して、大規模パラレル問合せ操作をサポートします。

関連項目:

Oracle Flex Clusterの詳細は、『Oracle Clusterware管理およびデプロイメント・ガイド』を参照してください。

- ディスク・グループの共有Oracle ASMパスワード・ファイル

  この機能によって、Oracle Automatic Storage Management(Oracle ASM)共有パスワード・ファイルをOracle ASMディスク・グループに格納する場合のブートストラップ問題に対処するために必要なインフラストラクチャが実装されます。

関連項目:

Oracle Automatic Storage Management管理者ガイド

- グローバル・データ・サービス

  Oracle RACがデータベース・サービスをサポートし、クラスタ内の複数のデータベース・インスタンスをまたいだサービスレ

ベルのワークロード管理を有効にするのと同様の方法で、Global Data Servicesでは、共有サービスを提供する一連のレプリケート・データベースに対する、Oracle RACに似た接続時ロード・バランシング、ランタイム・ロード・バランシング、フェイルオーバーおよびサービスの集中管理を提供します。この一連のデータベースには、Oracle RAC、およびOracle Data Guard、Oracle GoldenGateまたはその他のレプリケーション・テクノロジで相互接続された、非クラスタ化Oracle Databaseを含めることができます。

関連項目:

詳細は、Oracle Database Global Data Services概要および管理ガイドを参照してください

- 共有グリッド・ネーミング・サービス

  グリッド・ネーミング・サービス(GNS)の1つのインスタンスで、任意の数のクラスタにサービスを提供できます。

  関連項目:

  詳細は、『Oracle Clusterware管理およびデプロイメント・ガイド』を参照してください。

- Oracle RACのWhat-Ifコマンドの評価

  Oracle Clusterwareのこの機能を使用すると、システムの状態を変更することなく、仮定的な計画済または計画外イベントに対するポリシー・レスポンスを提供するメカニズムによって、リソースの管理および可用性が向上します。

  Oracle RACでは、SRVCTLに対する拡張が、特定のコマンドを実行してその潜在的影響を判別する前に、それらのコマンドの影響を特定するのに役立ちます。

  関連項目:

  - What-If機能を使用するSRVCTLコマンドのリストは、「SRVCTL使用情報」を参照してください
  - Oracle Clusterware制御(CRSCTL)ユーティリティ・コマンドおよび同様の拡張機能のリストは、『Oracle Clusterware管理およびデプロイメント・ガイド』を参照してください

- Oracle RACデプロイメントへのサービス登録の制限

  この機能により、デフォルトでローカルIPからのリスナー登録のみが許可され、登録要求がリスナーにより許可される一連のIPアドレスまたはサブネットを構成および動的に更新する機能が提供されます。

  関連項目:

  詳細は、『Oracle Database Net Services管理者ガイド』を参照してください

- 有効なノードの確認によるサービス登録の制限

  この機能によって、ネットワーク管理者は、単一クライアント・アクセス名(SCAN)リスナーが登録を受け入れたノードのリストおよびサブネット情報を指定できます。SRVCTLを使用してノードおよびサブネット情報を指定でき、SRVCTLではこの情報をSCANリスナー・リソース・プロファイルに格納し、この情報は`listener.ora`ファイルにも書き込まれます。
  データベースへのクライアント・アクセスを制限すると、Oracle RACがよりセキュアになり、セキュリティ上の問題や攻撃に対する脆弱性が軽減されます。

- プラガブル・データベース

  プラガブル・データベースによって、Oracle Databaseはスキーマ、スキーマ・オブジェクトおよび非スキーマ・オブジェクトのポータブル・コレクションを含むことができます。これらはOracle Netクライアントに個別のデータベースとして表示されます。この自己完結型コレクションは、プラガブル・データベース(PDB)と呼ばれます。コンテナ・データベース(CDB)とは、0 (ゼロ)、1つまたは数多くのユーザー作成のプラガブル・データベース(PDB)を含むOracle Databaseです。PDBはCDBから切断して、別のCDBに接続できます。

  関連項目:

  Oracle Database管理者ガイド

- WindowsでのOracleホーム・ユーザーのサポート

  Oracle Database 12c以降、Oracle Databaseでは、Oracleホーム・ユーザーの使用がサポートされており、インストール時に指定できます。Oracleホーム・ユーザーは、Windowsドメイン・ユーザーに関連付けられています。制限された権限セットをOracleホーム・ユーザーに確実に付与することで、Oracle製品の実行に必要な権限のみがOracle Databaseサービスに付与されるようにするには、Windowsドメイン・ユーザーを権限の弱い非管理者アカウントにする必要があります。

  Windowsの管理者ユーザーの権限は、インストール、アップグレード、パッチ適用などのOracleソフトウェア・メンテナンス・タスクを実行するために依然として必要です。Oracle Database管理ツールは、必要な場合はOracleホーム・ユーザーのパスワードを尋ねるように拡張されました。Oracle RAC環境では、Oracleホーム・ユーザーのパスワードをセキュア・ウォレットに保存できます。このようなウォレットが存在する場合、Oracle Database管理ツールでは、ウォレットからのパスワードが自動的に使用され、Oracleホーム・ユーザーのパスワードの入力は求められません。

- Oracle ACFSおよびOracle ADVMのクラスタ・リソース

  Oracle Clusterwareリソースのサポートには、Oracle Automatic Storage Management Cluster File System (Oracle ACFS)、Gridホーム用のOracle ACFS汎用ファイル・システム、およびOracle ASM Dynamic Volume Manager (Oracle ADVM)のボリュームに格納されたOracleホームの拡張が含まれます。Oracle Clusterwareによって管理されるこれらのリソースでは、Oracle ACFS、Oracle ADVMドライバとOKSドライバ、ディスク・グループのマウント、動的ボリュームの有効化、および自動Oracle ACFSファイル・システムのマウントがサポートされています。

  関連項目:

  詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください

- Oracle Highly Available NFS

  Oracle ACFSは、エクスポートされた可用性の高いファイル・システム・サービスとして構成できます。このサービスは、仮想IPアドレスと組み合せて、Oracle ACFSのクラスタ全体のデータの一貫性および整合性を使用し、NFSエクスポート用のフェイルオーバー機能を実現します。この仮想IPアドレスからNFSエクスポートをマウントすると、クラスタの1つのノードが使用可能である場合、NFSエクスポートは使用可能になることをクライアントに保証できます。

  関連項目:

  詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください

- ポリシーベースのクラスタ管理および運用

  Oracle Grid Infrastructureでは、1つのクラスタで複数のアプリケーションを実行することができます。ポリシーベースの方法を使用すると、これらのアプリケーションによって発生するワークロードを、ポリシーを使用したクラスタ全体に振り分けることができます。またポリシー設定によって、時間の経過とともに必要に応じて異なるポリシーをクラスタに適用することができます。Webベースのインタフェースまたはコマンドライン・インタフェースを使用して、ポリシー・セットを定義できます。

  同じクラスタ内で様々なワークロードを受け入れることで、共有インフラストラクチャにワークロードを集約することができ、高可用性とスケーラビリティが実現されます。集中管理されたポリシーベースの方法を使用することで、要求の変化に応じてリソースを動的に再配分し、優先度付けが可能になります。

  関連項目:

  詳細は、『Oracle Clusterware管理およびデプロイメント・ガイド』を参照してください。

- オンライン・リソース属性変更

  Oracle Clusterwareでは、リソース・モデルを使用して、高可用性のためにハードウェアおよびソフトウェアを管理します。リソース属性を使用して、Oracle Clusterwareによるこれらのリソースの管理方法を定義します。オンライン・リソース属性変更を使用すると、リソースを再起動しないで、特定のリソース属性を変更したり、これらの変更を実装することができます。特定のSRVCTLおよびCRSCTLコマンドを使用して、オンライン・リソース属性変更を管理します。

非推奨となった機能

単一文字SRVCTL CLIオプションの非推奨

すべてのSRVCTLコマンドは、単一文字オプションのかわりに完全単語オプションを受け入れるように拡張されました。このリリースで追加されたすべての新しいSRVCTLコマンド・オプションはフルワード・オプションのみをサポートしており、1文字のオプションはサポートしていません。SRVCTLコマンドでの単一文字オプションの使用は、今後のリリースではサポートされない可能性があります。

サポート対象外となった機能

関連項目:

詳細は、『Oracle Databaseアップグレード・ガイド』を参照してください。

- Windows用のOracle Cluster File System

  WindowsでのOracle Clusterファイル・システム(OCFS)はサポートされていません。

- Oracle Database用のRAW (ブロック)記憶域デバイスおよび関連テクノロジ

  Oracle Database 12cリリース1 (12.1)およびOracle Clusterwareなどの関連するグリッド技術は、RAWストレージ・デバイスまたはブロック・ストレージ・デバイスの直接使用を現在はサポートしていません。Oracle Clusterware 12cリリース1 (12.1)にアップグレードする前に、既存のファイルをRAWまたはブロック・デバイスからOracle ASMに移動する必要があります。

1 Oracle RACの概要

Oracle Real Application Clusters (Oracle RAC)のインストールと管理、および各種コンポーネントと機能についての概要を説明します。

この章の内容は次のとおりです。

- Oracle RACの概要
- Oracle RACのインストールの概要
- Oracle Real Application Clusters One Nodeの概要
- Oracle ClusterwareおよびOracle RACの概要
- Oracle RACのアーキテクチャおよび処理の概要
- 動的データベース・サービスによる自動ワークロード管理の概要
- サーバー・プールおよびポリシー管理データベースの概要
- Oracle Database Quality of Service Managementの概要
- ハング・マネージャの概要
- Oracle RACを含むOracle Multitenantの概要
- Database In-MemoryおよびOracle RACの概要
- Oracle RAC環境の管理の概要

Oracle RACの概要

このトピックでは、Oracle RACとその機能について紹介します。

非クラスタのOracle Databaseには、Oracle Databaseとインスタンス間に1対1関係があります。しかし、Oracle RAC環境では、データベースとインスタンス間に1対多の関係があります。Oracle RACデータベースには複数のインスタンスが存在でき、それらすべてが1つのデータベースにアクセスします。すべてのデータベース・インスタンスは同じインターコネクトを使用する必要があり、Oracle Clusterwareもこれを使用します。

各Oracle RACデータベース・インスタンスには次のものも存在するため、Oracle RACデータベースは、非クラスタのOracle Databaseとアーキテクチャが異なります。

- 各インスタンスの1つ以上の追加REDOスレッド
- インスタンス固有のUNDO表領域

複数のサーバーの処理能力を組み合せることによって、単一のサーバーの場合よりも優れたスループットおよびOracle RACスケーラビリティを実現できます。

クラスタは、相互に接続された複数のコンピュータまたはサーバーで構成され、エンド・ユーザーおよびアプリケーションからは1つのサーバーとして認識されます。Oracle DatabaseとともにOracle RACオプションを使用すると、Oracle Databaseをクラスタ化できます。Oracle RACでは、インフラストラクチャとしてOracle Clusterwareを使用し、複数のサーバーを関連付けてそれ

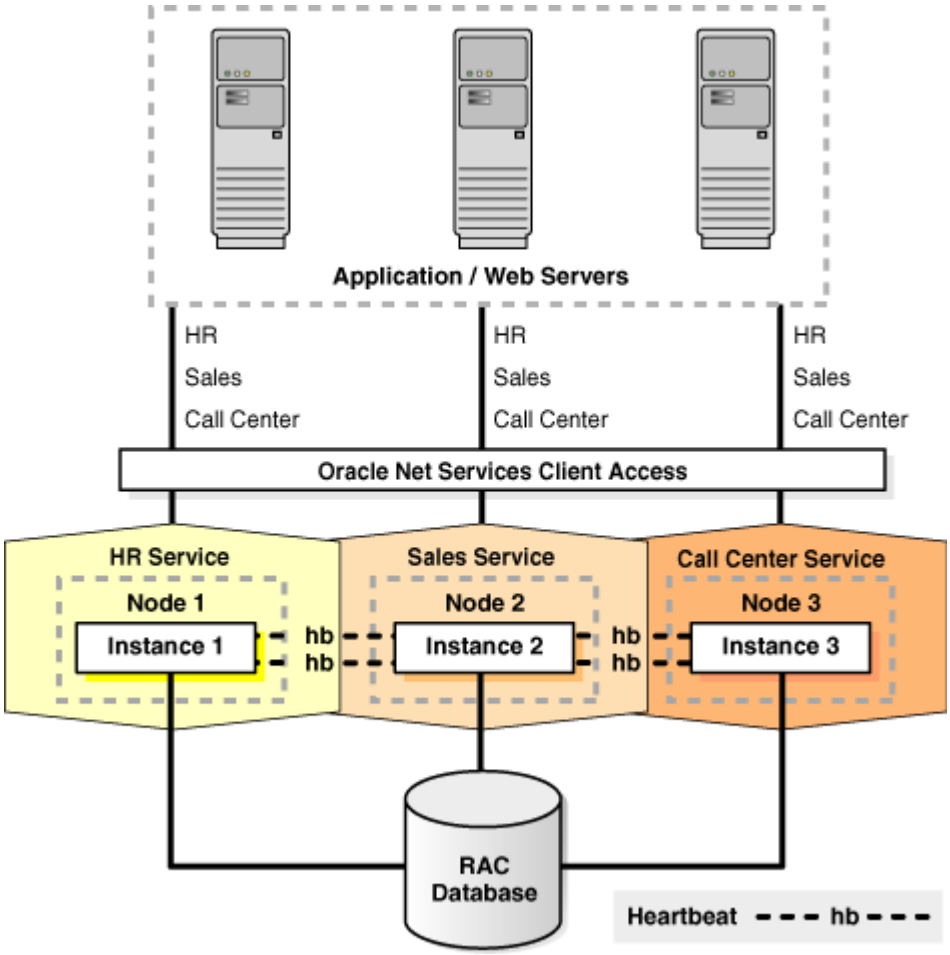
らが単一のシステムとして動作するように構成します。

Oracle Clusterwareは、Oracle Databaseと統合された、ポータブルなクラスタ管理ソリューションです。Oracle Clusterwareは、Oracle RACの実行に必要なインフラストラクチャを提供するOracle RACを使用するために必要なコンポーネントです。Oracle Clusterwareでは、仮想インターネット・プロトコル(VIP)・アドレス、データベース、リスナー、サービスなどのリソースも管理します。さらに、Oracle Clusterwareによって、非クラスタのOracle DatabaseとOracle RACデータベースの両方で、Oracle高可用性インフラストラクチャを使用できます。Oracle ClusterwareとOracle Automatic Storage Management (Oracle ASM) (この2つは一緒になってOracle Grid Infrastructureを構成します)が一緒になることによって、非クラスタとOracle RACデータベースの任意の組合せで使用されるストレージのクラスタ化プールを作成できます。

Oracle RACが動作するほとんどのプラットフォームにおいて、必要なクラスタウェアはOracle Clusterwareのみです。データベース・アプリケーションでベンダー・クラスタウェアが必要とされている場合、このベンダー・クラスタウェアがOracle RAC用に認証されていれば、このクラスタウェアはOracle Clusterwareとともに使用できます。

図1-1に、Oracle Databaseのオプションであるオラクル RACにより、複数のサーバーが1つのOracle Databaseにアクセスするための単一のシステム・イメージが提供される方法を示します。Oracle RACでは、各Oracleインスタンスは異なるサーバー上で実行する必要があります。

図1-1 Oracle RACアーキテクチャでのOracle Database



従来、Oracle RAC環境は、1つのデータ・センターにあります。ただし、Oracle RACはOracle拡張クラスタ上に構成できます。このアーキテクチャでは、サイト障害からの非常に高速なリカバリを実現し、すべてのサイトのすべてのノードで単一のデータベー

ス・クラスタの一部としてアクティブにトランザクションを処理できます。拡張クラスタでは、クラスタ内のノードは通常、2つのファイア・セルの間、2つの部屋や建物の間、2つの異なるデータ・センターや都市の間など、地理的に分散されます。可用性の理由から、データを両方のサイトに配置する必要があり、これにより、記憶域に対してディスク・ミラーリング技術の実装が必要になります。

このアーキテクチャの実装を選択する場合、特に距離、待機時間および提供される保護の程度を考慮し、このアーキテクチャがビジネスに対してよい解決策となるかどうかを評価する必要があります。拡張クラスタ上のOracle RACは、ローカルのOracle RACクラスタで可能となるよりも優れた高可用性を実現しますが、組織の障害時リカバリ要件を満たすとはかぎりません。適切な分離は一部の災害(局所的停電、サーバー室の冠水など)に対する有効な保護策となりますが、あらゆる種類の障害に効果があるわけではありません。破損や地域災害に対する防御を含む災害に対する包括的な保護策として、『*Oracle Data Guard*概要および管理』および次のMaximum Availability Architecture(MAA) Webサイトで説明するように、Oracle RACとともにOracle Data Guardを使用することをお薦めします。

Oracle RACは、すべてのタイプのアプリケーションに対して高可用性および高スケーラビリティを提供する特殊な技術です。また、Oracle RACインフラストラクチャは、Oracleエンタープライズ・グリッド・コンピューティング・アーキテクチャを実装するための主要なコンポーネントです。複数のインスタンスが単一のデータベースにアクセスすることで、サーバーがシングル・ポイント障害になることを防止できます。Oracle RACを使用すると、小規模な汎用サーバーをクラスタに組み込んで、ミッション・クリティカルなビジネス・アプリケーションをサポートするスケーラブルな環境を構築できます。Oracle RACデータベースにデプロイするアプリケーションは、コードを変更せずに使用できます。

関連項目

- Oracle Clusterwareの概要
- Oracle Grid Infrastructureのインストレーション・ガイド
- Oracle Data GuardとOracle Real Application Clusters
- Maximum Availability Architecture(MAA)

Oracle RACのインストールの概要

Oracle Universal Installerを使用してOracle Grid InfrastructureおよびOracle Databaseソフトウェアをインストールし、Oracle Database Configuration Assistant (Oracle DBCA)を使用してデータベースを作成します。

データベースの作成によって、Oracle RAC環境のネットワーク構成、データベース構造およびパラメータ設定が、選択された環境に最適なものになります。

また、Oracle RACのインストールにフリート・パッチ適用およびプロビジョニングを使用することもできます。これにより、Oracle Universal Installerと以前に指定したOracle DBCAを完全に活用できます。また、フリート・パッチ適用およびプロビジョニングでは標準化と自動化が可能です。

この項では、Oracle RACのインストール・プロセスについて説明します。内容は次のとおりです。

- Oracle RAC環境の互換性
- Oracle RACデータベース管理スタイルおよびデータベースのインストレーション
- Oracle RACデータベース管理スタイルおよびデータベースの作成

- Oracle RACクラスタの拡張の概要

ノート:

Oracle RAC をインストールする前に、まず Oracle Grid Infrastructure をインストールする必要があります。

関連項目

- 『Oracle Real Application Clustersインストレーション・ガイド』
- Oracle Grid Infrastructureのインストレーション・ガイド

Oracle RAC環境の互換性

同じクラスタ内で様々なバージョンのOracle Databaseを備えた構成でOracle RACを実行するには、まずOracle Grid Infrastructureをインストールする必要があります。これは、クラスタ内にデプロイする最高バージョンのOracle Databaseと同じバージョン以上にする必要があります。たとえば、Oracle RAC 12cデータベースおよびOracle RAC 18cデータベースを同じクラスタ内で実行するには、Oracle Grid Infrastructure 18cをインストールする必要があります。Oracle RAC環境におけるバージョンの互換性の詳細は、My Oracle Supportに問い合せてください。

ノート:

Oracle9i クラスタの Oracle Grid Infrastructure 12c 以降の環境へのデプロイはサポートされません。

Oracle RACデータベース管理スタイルおよびデータベースのインストレーション

Oracle RACデータベース・ソフトウェアをインストールし、それぞれのデータベースを作成する前に、「サーバー・プールおよびポリシー管理データベースの概要」で説明するとおり、Oracle RACデータベースに適用する管理スタイルについて決めます。

選択する管理スタイルは、ソフトウェアのデプロイメントおよびデータベースの作成に影響を与えます。管理者管理データベース・デプロイメント・モデルを選択し、ソフトウェアのノード単位のインストレーションを使用する場合、Oracle Databaseを実行する予定のノードにのみOracle Databaseソフトウェア(データベース・ホーム)をデプロイすれば十分です。

ソフトウェアのノード単位のインストレーションを使用して、ポリシー管理デプロイメント・モデルを選択した場合、クラスタ内のすべてのノードにソフトウェアをデプロイする必要があります。サーバー・プールへのサーバーの動的割当ては原則的に、データベース・インスタンスが実行される可能性のあるサーバーを予測しないためです。それぞれのデータベース・ホームをホストしないサーバーでのインスタンスの起動障害を避けるには、クラスタ内のすべてのノードにデータベース・ソフトウェアをデプロイすることを強くお薦めします。共有Oracle Databaseホームを使用する場合、クラスタ内のすべてのノードからこのホームへのアクセシビリティが想定されるため、設定では、必要に応じてすべてのサーバーにそれぞれのファイル・システムがマウントされるようにする必要があります。

クラスタ用にOracle Grid Infrastructureをすでにインストールして構成した場合、Oracle Universal Installerでは、クラスタ内のノードにわたってOracle Databaseホームをデプロイすることのみできます。Oracle Universal Installerで、クラスタ内のすべてのノードにわたってデータベース・ホームをデプロイするオプションが得られない場合は、Oracle Universal Installer

に表示される前提条件を確認してください。

インストール中は、データベース・ホームのインストール時にデータベースの作成を選択できます。Oracle Universal InstallerはDBCAを実行し、選択したオプションに従ってOracle RACデータベースを作成します。

関連項目:

このオプションを選択する場合の詳細は、「Oracle RACデータベース管理スタイルおよびデータベースの作成」を参照してください

> ノート:
>
> 
>
> データベースを作成する前に、Oracle Grid Infrastructure ホームでデフォルトのリスナーを実行しておく必要があります。デフォルトのリスナーが Oracle Grid Infrastructure ホームに存在しない場合は、デフォルトのリスナーを作成するために Oracle Grid Infrastructure ホームから NETCA を実行することを指示したエラーが DBCA から返されます。

Oracle RACソフトウェアは、Oracle Databaseインストール・メディアの一部として配布されます。Oracle Databaseソフトウェアのインストール・プロセスでは、クラスタ上でインストールを実行していることが認識されると、デフォルトでOracle RACオプションもインストールされます。Oracle Universal Installerでは、Oracle RACがOracleホームと呼ばれるディレクトリ構造にインストールされます(これは、システムで実行中の他のOracleソフトウェアのOracleホーム・ディレクトリとは別のものです)。Oracle Universal Installerは、クラスタに対応しているため、クラスタの一部として定義したすべてのノードにOracle RACソフトウェアをインストールします。

関連項目

- サーバー・プールおよびポリシー管理データベースの概要
- Oracle RACデータベース管理スタイルおよびデータベースの作成
- Oracle Database Net Services管理者ガイド

Oracle RACデータベース管理スタイルおよびデータベースの作成

Oracle Databaseのデプロイメントの一部は、データベースの作成です。

データベース・ソフトウェア・デプロイメントの一環としてデータベースの作成を選択することも、まずデータベース・ソフトウェアのみをデプロイし、その後にDBCAを使用して、新しく作成されたOracleホームから実行するデータベースの作成を選択することもできます。いずれの場合も、Oracle RACデータベースに使用する予定の管理スタイルを考慮する必要があります。

管理者管理データベースの場合、それぞれのデータベース・インスタンスを実行する予定のノードにデータベース・ソフトウェアをデプロイする必要があります。これらのノードが、データベース・ファイルを格納する記憶域にアクセスできるようにする必要もあります。記憶域の管理を簡略化するために、データベースのインストール時にOracle ASMを選択することをお薦めします。Oracle ASMは、ディスク・グループ内のすべてのデータベース・ファイルの記憶域を自動的に管理します。

ポリシー管理データベースの場合、アクティブなサーバー・プール設定を考慮に入れると、データベース・インスタンスを実行する可

能性のあるすべてのノードにデータベース・ソフトウェアをデプロイする必要があります。これらのノードが、データベース・ファイルを格納する記憶域にアクセスできるようにする必要もあります。管理者管理データベースのところですでに説明したとおり、Oracle ASMを使用することをお薦めします。

サーバー・プールは、Oracle Grid Infrastructure (特にOracle Clusterware)の機能です。Oracle Clusterwareレベルでサーバー・プールを設定するには様々な方法がありますが、それぞれのデータベースを作成する前に、データベース管理用のサーバー・プールを作成することをお薦めします。ただし、ポリシー管理データベースを作成する場合は、事前に作成されたサーバー・プールを使用するか、新しいサーバー・プールを作成するかの選択肢がDBCAによって提示されます。データベースの作成時に新しいサーバー・プールを作成できるかどうかは、そのときにアクティブになっているサーバー・プールの構成によって決まります。

デフォルトでは、DBCAによって1つのサービスがOracle RACインストール用に作成されます。これはデフォルトのデータベース・サービスで、ユーザーの接続用には使用しないでください。デフォルトのデータベース・サービスは、通常、DB_NAMEおよびDB_DOMAIN初期化パラメータの組合せdb_name.db_domainを使用して識別されます。データベースが制限モードになっていないかぎり、このデフォルトのサービスはOracle RAC環境のすべてのインスタンスで使用できます。

ノート:

SRVCTL または Oracle Enterprise Manager を使用して、メンテナンス操作用にデフォルトのデータベース・サービスを確保し、データベース作成後のステップとして、ユーザーまたはアプリケーションの接続用に動的データベース・サービスを作成することをお薦めします。DBCA では、Oracle RAC データベース用の動的データベース・サービス作成オプションを提供していません。Oracle RAC One Node データベースの場合、少なくとも 1 つの動的データベース・サービスを作成する必要があります。

関連項目

- Oracle RACデータベース管理スタイルおよびデータベースのインストレーション
- Oracle Clusterware管理およびデプロイメント・ガイド

Oracle RACクラスタの拡張の概要

初期デプロイメント後に、Oracle RACクラスタ(クローニングとも呼ばれます)を拡張し、既存の環境にノードを追加する場合は、クラスタ内で現在使用している管理スタイルを考慮して、複数のレイヤーでこれを行う必要があります。

Oracle RACクラスタを拡張するための様々な手段が用意されています。原則として、現在の環境を拡張するために次のアプローチから選択できます。

- 新しいOracle RACデータベースとその他のソフトウェアをプロビジョニングするためのフリート・パッチ適用およびプロビジョニング
- クローニング・スクリプトを使用したクローニング
- addnode.sh (Windowsの場合はaddnode.bat)スクリプトを使用したノードの追加

環境の初期のデプロイ方法にかかわらず、どちらのアプローチも適用できます。どちらのアプローチも、クラスタを追加する予定のノードに、必要なOracleソフトウェアをコピーします。ノードにコピーされるソフトウェアには、Oracle Grid Infrastructureソフトウェアおよびデータベースホームがあります。

Oracle Databaseホームの場合、クラスタにデプロイされている管理スタイルを考慮する必要があります。管理者管理データベースの場合、それぞれのデータベース・インスタンスを実行する予定のノードにデータベース・ソフトウェアをデプロイする必要があります。ポリシー管理データベースの場合、アクティブなサーバー・プール設定を考慮に入れると、データベース・インスタンスを実行する可能性のあるすべてのノードにデータベース・ソフトウェアをデプロイする必要があります。いずれの場合も、クラスタの一部にするつもりのすべてのノードにまずOracle Grid Infrastructureをデプロイする必要があります。

ノート:

Oracle クローニングは、Provisioning Pack の一部である Oracle Enterprise Manager を使用したクローニングにかわるものではありません。Oracle Enterprise Manager を使用して Oracle RAC をクローニングする場合、プロビジョニング・プロセスには、取得するホーム、デプロイする場所、および収集される他の様々なパラメータに関する詳細情報が記述された一連のステップが含まれています。

新規インストールの場合、または1つのOracle RACデータベースのみをインストールする場合は、従来の自動化された対話式インストール方法を使用します(Oracle Universal Installer、フリート・パッチ適用およびプロビジョニング、Oracle Enterprise ManagerのProvisioning Pack機能など)。クラスタ内のノードにOracle RACを追加したりノードから削除することが目的の場合は、「LinuxおよびUNIXシステムのノードでのOracle RACの追加と削除」に説明されている手順を使用できます。

クローニングのプロセスは、Oracle ClusterwareホームおよびOracle RACを含むOracleホームが1つ以上のノードに正常にインストールされていることを前提としています。さらに、クラスタ・データベースの拡張元となるノードですべてのルート・スクリプトが正常に実行されている必要があります。

関連項目

- 新規クラスタのノードへのOracle RACのクローニング
- LinuxおよびUNIXシステムのノードでのOracle RACの追加と削除
- WindowsシステムのノードでのOracle RACの追加と削除
- Oracle Clusterware管理およびデプロイメント・ガイド

関連項目:

- フリート・パッチ適用およびプロビジョニングの詳細は、*Oracle Clusterware*管理およびデプロイメント・ガイドを参照してください。
- Provisioning Packの詳細は、Oracle Enterprise Managerオンライン・ヘルプ・システムを参照してください。

Oracle Real Application Clusters One Nodeの概要

Oracle Real Application Clusters One Node (Oracle RAC One Node)はOracle Database 11gリリース2 (11.2)以降に使用可能なOracle Database Enterprise Editionのオプションです。

Oracle RAC One Nodeは、クラスタ内の1つのノードで通常の操作のみで実行されるOracle RAC対応データベースの単一インスタンスです。このオプションにより、企業内にOracle Databases用の標準デプロイメントを提供することで管理オー

バーヘッドを削減しながら、オラクル社がデータベースの統合に対して提供する柔軟性が向上します。Oracle RAC One Nodeデータベースには、Oracle Grid Infrastructureが必要なため、Oracle RACデータベースと同じハードウェア設定が必要になります。

Oracle RACが認証されているすべてのプラットフォームでOracle RAC One Nodeがサポートされています。Oracle RACと同様に、Oracle Virtual Machine(Oracle VM)でのOracle RAC One Nodeの動作が保証されます。Oracle RACまたはOracle RAC One NodeをOracle VMで使用すると、Oracle RACの高可用性およびスケーラビリティによってOracle VMのメリットが大きくなります。

Oracle RAC One Nodeでは、サーバーのスケーラビリティは無制限で、アプリケーションが増大して単一ノードで提供できるリソース以上のリソースを必要とする場合には、アプリケーションをOracle RACにオンラインでアップグレードできます。Oracle RAC One Nodeが実行されているノードがオーバーロードになった場合は、インスタンスをクラスタ内の別のインスタンスに再配置できます。Oracle RAC One Nodeでは、オンライン・データベース再配置機能を使用して、アプリケーション・ユーザーには停止時間なしでデータベース・インスタンスを再配置できます。あるいは、リソース・マネージャ・インスタンス・ケージングを使用して、クラスタ内のサーバーごとに個々のデータベース・インスタンスのCPU使用率を制限でき、必要な場合は要求シナリオに応じてこの制限を動的に変更できます。

単一クライアント・アクセス名(SCAN)を使用してデータベースに接続すると、クライアントは、データベースが実行されているノードのサービスを独自に特定できます。したがって、クライアント接続によっては、Oracle RAC One Nodeインスタンスの再配置がクライアントに対してほとんど透過的になります。クライアントでの再配置の影響を最小限に抑えるために、アプリケーション・コンティニュイティとOracleの高速アプリケーション通知または透過的アプリケーション・フェイルオーバーを使用することをお薦めします。

Oracle RAC One Nodeデータベースの管理は、Oracle RACデータベースまたは非クラスタ・データベースとは若干異なります。管理者管理Oracle RAC One Nodeデータベースの場合は、候補ノード・リストを監視し、可能であればサーバーがいつでもフェイルオーバーに使用できるようにしておく必要があります。候補サーバーは汎用サーバー・プールに存在し、データベースとそのサービスは、いずれかの候補サーバーにフェイルオーバーします。

ポリシー管理Oracle RAC One Nodeデータベースの場合、現在のノードが使用できなくなった場合にサーバーがデータベースのフェイルオーバーに使用できるように、サーバー・プールが構成されていることを確認する必要があります。この場合、オンライン・データベース再配置用の宛先ノードは、データベースが配置されているサーバー・プールに配置される必要があります。または、サイズが1(サーバー・プール内に1つのサーバー)のサーバー・プールを使用して、最小サイズを1に設定し、クラスタ内で使用されている他のすべてのサーバー・プールに対して重要性を十分に高く設定することで、このサーバー・プールで使用されている1つのサーバーで障害が発生した場合に、そのサーバー・プールに必要に応じて別のサーバー・プールまたは空きサーバー・プールから新しいサーバーが確実に再配置されるようにする方法もあります。

ノート:

- Oracle RAC One Node は、クライアントのフェイルオーバー用に、トランザクション・ガードおよびアプリケーション・コンティニュイティをサポートしています。
- すべての障害の可能性に準備するために、少なくとも 1 つの動的データベース・サービス(Oracle Clusterware 管理データベース・サービス)を Oracle RAC One Node データベースに追加する必要

があります。

関連項目

- Oracle Real Application Clustersインストレーション・ガイドfor Linux and UNIX Systems
- クライアント・フェイルオーバーを向上させるためのトランザクション・ガード

Oracle ClusterwareおよびOracle RACの概要

Oracle Clusterwareは、すべてのOracle Databaseプラットフォームを対象とした完全な統合クラスタウェア管理ソリューションです。

このクラスタウェア機能は、クラスタ・データベースの管理に必要なすべての機能(ノードのメンバーシップ、グループ・サービス、グローバル・リソース管理および高可用性機能)を提供します。

Oracle Clusterwareは、単独でインストールすることも、Oracle RACインストール・プロセスの前提条件としてインストールすることもできます。サービスなどのOracle Database機能は、基盤となるOracle Clusterwareメカニズムを使用して高度な機能を提供します。特定のプラットフォームについては、一部のサード・パーティ製クラスタウェア製品も引き続きサポートされます。

Oracle Clusterwareは、Oracle RACのために設計され、Oracle RACに密接に統合されています。Oracle Clusterwareを使用して、クラスタで高可用性操作を管理できます。任意の管理ツールを使用してOracle RACデータベースを作成する場合、データベースは、VIPアドレス、単一クライアント・アクセス名(SCAN) (SCAN VIPおよびSCANリスナーを含みます)、Oracle Notification Service、Oracle Netリスナーなど他の必須コンポーネントとともにOracle Clusterwareに登録され、これによって管理されます。ノードが起動されるとこれらのリソースは自動的に起動され、リソースに障害が発生すると自動的に再起動されます。Oracle Clusterwareデーモンは各ノードで実行されます。

Oracle Clusterwareが管理するものはすべてCRSリソースと呼ばれます。CRSリソースには、データベース、インスタンス、サービス、リスナー、VIPアドレス、アプリケーション・プロセスがあります。Oracle Clusterwareは、Oracle Cluster Registry(OCR)に格納されているリソースの構成情報に基づいてCRSリソースを管理します。SRVCTLコマンドを使用して、すべてのOracle定義のCRSリソースを管理できます。Oracle Clusterwareは、クラスタ内のOracleによって事前定義されていないサーバー上で動作するすべてのプロセスを管理するためのCRSリソースを作成できるフレームワークを提供します。Oracle Clusterwareは、これらのコンポーネントの構成を説明する情報を管理可能なOCRに保存します。

この項には次のトピックが含まれます:

- Oracle Flex Clusterの概要
- リーダー・ノードの概要
- ローカル一時表領域の概要

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle Clusterware管理およびデプロイメント・ガイド

Oracle Flex Clusterの概要

Oracle Flex Clusterは、多数のノードを持つOracle RACデータベースなどの各種アプリケーションにプラットフォームを提供します。

Oracle Flex Clusterでは、高可用性のために調整および自動化が必要な他のサービス・デプロイメントのプラットフォームも提供されます。

Oracle Flex Cluster内のすべてのノードは、単一のOracle Grid Infrastructureクラスタに属します。このアーキテクチャでは、様々なサービス・レベル、負荷、障害のレスポンス、およびリカバリに対処するために、アプリケーション・ニーズに基づいてリソースのデプロイメントに対するポリシー決定が集中管理されます。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

リーダー・ノードの概要

リーダー・ノードは、主にレポート作成および分析の目的で読取り専用アクセスを提供するOracle RACデータベースのインスタンスです。

読取り専用インスタンスの利点は、通常の(読取り/書込み)データベース・インスタンスとは異なり、クラスタの再構成時(ノードでメンテナンスが行われたり、障害が発生した場合など)にパフォーマンスへの影響を受けないことです。

リーダー・ノードで実行する読取り専用のインスタンスに問合せを転送するサービスを作成できます。このようなサービスでは、さらにパフォーマンスを向上するためにパラレル問合せを使用できます。これらのリーダー・ノードのメモリーのサイズは、できるかぎり大きくして、パラレル問合せが最高のパフォーマンスを発揮するためのメモリーを使用できるようにします。

リーダー・ノードが書込み可能なデータベース・インスタンスをホストすることは可能ですが、最適なパフォーマンスを実現するために、リーダー・ノードを読取り専用インスタンスのホスティング専用にすることをお薦めします。

ローカル一時表領域の概要

ローカル一時表領域を使用してリーダー・ノードのローカル・ディスクに作成されたローカル(非共有)の一時表領域にスピル・オーバーを書き込みます。

ハッシュ集約、ソート、ハッシュ結合、WITH句のカーソル持続期間一時表の作成、ディスク(具体的には共有ディスク上のグローバル一時表領域)へのスピル・オーバーのためのスター型変換などのSQL操作は引き続き可能です。ローカル一時表領域の管理は、既存の一時表領域の管理と同じです。

ローカル一時表領域は、次の点で読取り専用インスタンスでの一時表領域管理を向上します。

- リーダー・ノードのプライベート記憶域に一時ファイルを保存することで、ローカル記憶域のI/Oのメリットを活用します。
- コストの高いインスタンス間の一時表領域管理を回避します。
- 一時表領域のアクセス性が向上されます。
- ディスク上の領域メタデータ管理の排除により、インスタンスのウォームアップ・パフォーマンスが向上します。

ノート:

ローカル一時表領域は、データベース・オブジェクト(表や索引など)の保存には使用できません。これと同じ制限が Oracle グローバル一時表にも適用されます。

この項には次のトピックが含まれます:

- カーソル持続期間一時表領域のパラレル実行のサポート
- ローカル一時表領域の編成
- 一時表領域の階層
- ローカル一時表領域の機能
- ローカル一時ファイルのメタデータ管理
- ローカル一時表領域のDDLサポート
- ユーザー用のローカル一時表領域
- コマンドの原子性要件
- ローカル一時表領域とディクショナリのビュー

カーソル持続期間一時表領域のパラレル実行のサポート

WITH句およびスター型変換のために作成された一時表領域は、共有ディスク上の一時表領域に存続します。パラレル問合せの子プロセスのセットは、このような一時表領域に問合せの中間結果をロードします。この結果は、後から別の子プロセスのセットによって読み込まれます。こうした結果を読み取る子プロセスの割当て方法には制限はありません。これは、任意のインスタンス上の任意のパラレル問合せ子プロセスが、共有ディスク上に存在する一時表領域を読み取ることができるためです。

読取り/書込みおよび読取り専用インスタンス・アーキテクチャの場合、パラレル問合せ子プロセスは、これらのインスタンスのローカル一時表領域に中間結果をロードするため、中間結果が保存されているインスタンスに属しているパラレル問合せ子プロセスは、中間結果の読取りをアフィニティと共有するため、中間結果を読み取れるようになります。

ローカル一時表領域の編成

ローカル一時表領域は、次のように作成できます。

```
CREATE LOCAL TEMPORARY TABLESPACE TEMPFILE¥
   '/u01/app/oracle/database/12.2.0.1/dbs/temp_file'¥
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

- ローカル一時表領域を作成すると、単一のファイルではなく、インスタンスごとにローカル一時ファイルを作成することになります。これは、現時点では、共有グローバル一時表領域にも当てはまります。
- ローカル一時表領域は、読取り専用インスタンスと読取り/書込みインスタンスの両方に作成できます。たとえば:

```
CREATE LOCAL TEMPORARY TABLESPACE TEMPFILE¥
   '/u01/app/oracle/database/12.2.0.1/dbs/temp_file' ¥
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

一時表領域の階層

ローカル一時表領域と共有(既存の)一時表領域を定義するときに、それらが使用する階層が存在します。この階層を理解するために、データベース用のデフォルト共有一時表領域と個別のユーザーに割り当てられる複数の一時表領域のように、1つのデータベース内に複数の共有一時表領域が存在することに注意してください。ユーザーに共有一時表領域が割り当てられている場合は、その表領域が最初に使用されます。それ以外の場合は、デフォルトの一時表領域が使用されます。

問合せの処理時に書出し用の表領域が選択されると、別の表領域に切り替えられることはなくなります。たとえば、ユーザーに共有一時表領域が割り当てられていて、その領域が書出し中に使い果たされても、代替の表領域に切り替えられることはありません。その場合は、書出しによるエラーが発生します。さらに、共有一時表領域はインスタンス間で共有されることに注意してください。

ローカル一時表領域への書出し用の一時領域の割当ては、読取り専用インスタンスと読取り/書込みインスタンスでは異なります。読取り専用インスタンスの場合、書出しに使用する一時的な場所を選択する際の優先順位は次のようになります。

1. ユーザーのローカル一時表領域からの割当て。
2. データベースのデフォルト・ローカル一時表領域からの割当て。
3. ユーザーの一時表領域からの割当て。
4. データベースのデフォルト一時表領域からの割当て。

ノート:

データベースにローカル一時表領域が存在しない場合、読取り専用インスタンスは共有一時表領域に書出しするようになります。

読取り/書込みインスタンスの場合、割当ての優先順位は前述の割当て順序と異なります。これは、次に示すように、共有一時表領域に優先順位が与えられているためです。

1. ユーザーの共有一時表領域からの割当て。
2. ユーザーのローカル一時表領域からの割当て。
3. データベースのデフォルト共有一時表領域からの割当て。
4. データベースのデフォルト・ローカル一時表領域からの割当て。

ローカル一時表領域の機能

インスタンスはローカル一時表領域を共有できないため、あるインスタンスが別のインスタンスからローカル一時表領域を取得することはできません。あるインスタンスが書出し中に一時表領域を使い果たすと、その文によってエラーが発生します。

- ローカル一時表領域は、表領域ごとに1つのBIGFILEのみをサポートします。
- BIGFILEベースのローカル一時表領域が1つしかないために発生する競合の問題に対処するために、それぞれのユーザーにデフォルトとして複数のローカル一時表領域を割当てできます。
- データベース管理者は、ALTER USER構文を使用して、デフォルトの一時表領域をユーザーに指定できます。たとえば:

  ```
  ALTER USER MAYNARD LOCAL TEMPORARY TABLESPACE temp_ts;
  ```

- ユーザーは、2つのデフォルト一時表領域で構成できます。

- 1つのローカル一時: ユーザーがリーダー・ノードで実行している読取り専用インスタンスに接続する場合。
- 1つの共有一時表領域: 同じユーザーがハブ・ノードで実行している読取り/書込みインスタンスに接続したときに使用されます。

ローカル一時ファイルのメタデータ管理

現時点では、一時ファイルの情報(ファイル名、作成サイズ、作成SCN、一時ブロック・サイズ、ファイル・ステータスなど)は、自動エクステントの属性および初期ファイルと最大ファイルとともに制御ファイルに保存されます。ただし、制御ファイルのローカル一時ファイルに関する情報は、適用可能なすべてインスタンスに共通です。

インスタンス固有の情報(割当てのビットマップ、一時ファイルの現在のサイズ、ファイル・ステータスなど)は、インスタンスのSGAに保存され、制御ファイルには保存されません。これは、この情報がインスタンスごとに異なることがあるためです。インスタスは、起動時に制御ファイルの情報を読み取って、そのインスタンスのローカル一時表領域を構成する一時ファイルを作成します。1つのノードで実行しているインスタンスが複数存在する場合、それぞれのインスタンスが専用のローカル一時ファイルを持つようになります。

ローカル一時表領域の場合は、関連するインスタンスごとに個別のファイルが存在します。ローカル一時ファイルの名前は、ローカル一時表領域の作成時に指定した一時ファイルの名前にインスタンス番号が付加されるというネーミング規則に従います。

たとえば、読取り専用ノードN1で番号3および4の2つの読取り専用Oracleデータベース・インスタンスを実行していると仮定します。次に示すDDLコマンドにより、ノードN1に2つのファイル/temp/temp_file_3と/temp/temp_file_4が、それぞれインスタンス3と4に作成されます。

```
CREATE LOCAL TEMPORARY TABLESPACE TEMPFILE  '/temp/temp_file'¥
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

2つの読取り/書込みインスタンス(インスタンス番号1および2)と、2つの読取り専用インスタンス(インスタンス番号3および4)があるとします。次に示すDDLコマンドにより、4つのファイルが作成されます。インスタンス1と2には、それぞれ/temp/temp_file_all_1と/temp/temp_file_all_2が作成され、インスタンス3と4には、それぞれ/temp/temp_file_all_3と/temp/temp_file_all_4が作成されます。

```
CREATE LOCAL TEMPORARY TABLESPACE temp_ts TEMPFILE  '/temp/temp_file_all'¥
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

ローカル一時表領域のDDLサポート

ローカル一時表領域と一時ファイルの管理には、ALTER TABLESPACEとALTER DATABASEのどちらか、または両方のDDLコマンドを使用します。ローカル一時表領域の管理と作成に関連するすべてのDDLコマンドは、読取り/書込みインスタンスから実行します。その他すべてのDDLコマンドの実行は、すべてのインスタンスに同様に作用します。

たとえば、次のコマンドは、一時ファイルとすべての読取り専用インスタンスのサイズを変更します。

```
ALTER TABLESPACE temp_ts RESIZE 1G;
```

ローカル一時表領域の場合は、現時点で一時ファイルに対してアクティブな割当てオプションとその制限がサポートされます。

読取り専用インスタンスのローカル一時表領域に対してDDLコマンドを実行するには、クラスタ内に少なくとも1つの読取り専用インスタンスが必要です。ユーザーは、ALTER DATABASEコマンドにDEFAULT LOCAL TEMPORARY TABLESPACE句を追加するこ

とで、デフォルト一時表領域をデータベースに割り当てることができます。

たとえば:

```
ALTER DATABASE DEFAULT LOCAL TEMPORARY TABLESPACE temp_ts;
```

データベース管理者は、次に示すように、データベースの作成時にデフォルト一時表領域を指定できます。

```
CREATE DATABASE .. DEFAULT TEMPORARY TABLESPACE temp_ts_for_dbtemp_ts TEMPFILE¥
   '/temp/temp_file_for_db' EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

CREATE DATABASEコマンドを使用して、デフォルト一時表領域を指定することはできません。データベースの作成時、そのデータベースのデフォルト・ローカル一時表領域はデフォルト共有一時表領域を指すようになります。データベース管理者は、ALTER DATABASEコマンドを実行して、既存のローカル一時表領域をデータベースのデフォルトとして割り当てる必要があります。

ユーザー用のローカル一時表領域

ユーザーの作成時に、共有またはローカルの一時表領域を指定しないと、そのユーザーは対応するデフォルトの一時表領域から共有またはローカルの一時表領域を継承します。ユーザー用のローカル一時表領域は、次のように指定できます。

```
CREATE USER new_user IDENTIFIED BY new_user LOCAL TEMPORARY TABLESPACE temp_ts_for_all;
```

ユーザー用のローカル一時表領域は、次に示すようにALTER USERコマンドを使用して変更できます。

```
ALTER USER maynard LOCAL TEMPORARY TABLESPACE temp_ts;
```

前述したように、デフォルトのユーザー・ローカル一時表領域は、一時領域で共有されることがあります。ALTER USER...TEMPORARY TABLESPACEコマンドでは、次の事項について考慮します。

- ユーザーのデフォルト・ローカル一時表領域は、任意のローカル一時表領域に変更できます。
- ユーザーのデフォルト・ローカル一時表領域を共有一時表領域Tに設定する場合、Tはデフォルト共有一時表領域と同じであることが必要です。
- デフォルトのユーザー・ローカル一時表領域が共有一時表領域を指しているときに、ユーザーのデフォルト共有一時表領域を変更する場合は、デフォルトのローカル一時表領域も、その表領域に変更します。

次に、ALTERコマンドを使用してローカル一時領域を管理するいくつかの例を示します。

- ローカル一時表領域をオフラインにするには:

  ```
  ALTER DATABASE TEMPFILE  '/temp/temp_file'  OFFLINE;
  ```

- ローカル一時表領域のサイズを縮小するには:

  ```
  ALTER TABLESPACE temp_ts SHRINK SPACE KEEP 20M
  ```

- ローカル一時ファイルの自動拡張の属性を変更するには:

  ```
  ALTER TABLESPACE temp_ts AUTOEXTEND ON NEXT 20G
  ```

- ローカル一時ファイルのサイズを変更するには:

  ```
  ALTER TABLESPACE temp_ts RESIZE 10G
  ```

ノート:

ローカル一時ファイルのサイズを変更すると、そのサイズは個別のファイルに適用されます。

前述のいずれかのコマンドを実行すると、一部の読取り専用インスタンスは停止することがあります。これによって、コマンドの正常な実行が妨げられることはありません。その理由は、その後の起動時に、読取り専用インスタンスは制御ファイルの情報に基づいて新しい一時ファイルを作成するためです。作成は短時間で完了します。これは、一時ファイルのヘッダー・ブロック(特に、ファイル・サイズに関する情報を記録するブロック)のみがリフォーマットされるためです。一時ファイルのいずれかが作成できない場合は、読取り専用インスタンスが停止したままになっています。読取り/書込みインスタンスか発行したコマンドは、オープンしているすべての読取り専用インスタンスで即座にリプレイされます。

コマンドの原子性要件

読取り/書込みインスタンスから実行するすべてのコマンドは、原子的な方法で実行されます。つまり、コマンドは、すべてのライブ・インスタンスで成功した場合にのみ成功するということです。

ローカル一時表領域とディクショナリのビュー

ディクショナリのビューは、ローカル一時表領域に関する情報を表示するように拡張されています。次に示す変更が行われています。

- AWRやSQLモニタなどのユーティリティを通じて公開される一時表領域と一時ファイルに関連する診断可能なすべての情報は、ローカル一時表領域とローカル一時ファイルについても使用できます。この情報は、一時表領域と一時ファイルの既存ディクショナリ・ビュー(DBA_TEMP_FILES、DBA_TEMP_FREE_SPACE)で得られます。
- ディクショナリ・ビューのUSER_TABLESPACESとDBA_TABLESPACESは、SHAREDという列によって拡張されています。この列は、一時ファイルがローカルと共有のどちらであるかを示します。
- DBA_TEMP_FILESディクショナリ・ビューは、2つの列SHAREDとINST_IDによって拡張されています。SHARED列は、一時ファイルがローカルと共有のどちらであるかを示します。INST_ID列には、インスタンス番号が格納されます。共有一時ファイルの場合、ファイルごとに1つの行が存在し、INST_IDはnullになります。ローカル一時ファイルの場合、この列には、バイト単位のファイル・サイズ(BYTES列)など、インスタンスごとの一時ファイルに関する情報が格納されます。
- DBA_TEMP_FREE_SPACEディクショナリ・ビューは、2つの列SHAREDとINST_IDによって拡張されています。SHARED列は、一時ファイルがローカルと共有のどちらであるかを示します。INST_ID列には、インスタンス番号が格納されます。共有一時ファイルの場合、ファイルごとに1つの行が存在し、INST_IDはnullになります。ローカル一時ファイルの場合、この列には、使用可能な合計空き領域(FREE_SPACE列)など、インスタンスごとの一時ファイルに関する情報が格納されます。
- DBA_TABLESPACESなどのディクショナリ・ビューでは、次の値を含むSHARED列を使用して表領域のタイプが区別されます。
    - SHARED: 共有一時表領域の場合

ノート:

現時点では、問合せの一時表領域への書出し(ソートやハッシュ結合の書出しなど)は、自動的に暗号化されます。これは、ローカル一時表領域への書出しにも当てはまります。

関連項目

- Oracle Database SQL言語リファレンス

Oracle RACのアーキテクチャおよび処理の概要

Oracle RACの最小要件として、Oracle Clusterwareソフトウェア・インフラストラクチャが満たしている必要があるのは、同じ記憶域および同じ一連のデータ・ファイルにクラスタ内のすべてのノードから同時にアクセスできること、クラスタ内のノード同士でプロセス間通信(IPC)ができる通信プロトコルが用意されていること、論理的に結合された単一のキャッシュ上にデータが存在するかのように、複数のデータベース・インスタンスでデータを処理できること、およびクラスタ内のノードのステータスを監視して通信するメカニズムが用意されていることです。

詳細は、次の項を参照してください。

- クラスタを認識する記憶域ソリューション
- Oracle RACおよびネットワーク接続性
- Oracle Databasesに接続するための動的データベース・サービスの使用の概要
- Oracle RACのサービス登録の制限
- Oracle RACソフトウェア・コンポーネント
- Oracle RACバックグラウンド・プロセス

クラスタを認識する記憶域ソリューション

Oracle RACデータベースは、Shared Everythingデータベースです。Oracle RAC環境のすべてのデータ・ファイル、制御ファイル、SPFILEおよびREDOログ・ファイルは、すべてのクラスタ・データベース・インスタンスがこれらの記憶域コンポーネントにアクセスできるように、クラスタ対応共有ディスクに存在している必要があります。Oracle RACデータベースはShared Everythingアーキテクチャを使用するため、Oracle RACでは、すべてのデータベース・ファイルに対して、クラスタで認識される記憶域が必要です。

Oracle RACでは、Oracle Databaseソフトウェアによってディスク・アクセスが管理され、様々な記憶域アーキテクチャでの使用が保証されています。記憶域の構成方法は自由に選択できますが、サポートされているクラスタを認識する記憶域ソリューションを使用する必要があります。Oracle Databaseでは、次のようなOracle RAC用の記憶域オプションが用意されています。

- Oracle Automatic Storage Management(Oracle ASM)

  記憶域の管理にはこのソリューションをお薦めします。
- 認定されたクラスタ・ファイル・システム
  - Oracle Automatic Storage Management Cluster File System (Oracle ACFS)をお薦めします。
  - クラスタ対応ボリューム・マネージャ上の、Oracle RAC用に認定されているサード・パーティのクラスタ・ファイル・システム。たとえば:
    - Oracle OCFS2 (Linuxのみ)
    - IBM GPFS (IBM AIXのみ)
- 認定されたネットワーク・ファイル・システム(NFS)・ソリューション

Oracle RACおよびネットワーク接続性

Oracle RAC環境内のすべてのノードは、ユーザーおよびアプリケーションがデータベースにアクセスできるようにするために、少なくとも1つのLocal Area Network (LAN) (一般にパブリック・ネットワークと呼ばれます)に接続する必要があります。

Oracle RACでは、パブリック・ネットワークに加えて、ノードとこのノード上で動作するインスタンスの間の通信専用に使用するプライベート・ネットワーク接続が必要になります。このネットワークは、一般にインターコネクトと呼ばれます。

インターコネクト・ネットワークは、クラスタ内のすべてのサーバーに接続するプライベート・ネットワークです。インターコネクト・ネットワークは、1つ以上のスイッチと1つのギガビット・イーサネット・アダプタを使用する必要があります。

ノート:

- より大きい帯域幅とのインタフェースはサポートされていますが、インターコネクトとの間のクロスオーバー・ケーブルの使用はサポートされていません。
- キャッシュ・フュージョンによってインターコネクトがインスタンス間通信で使用されるため、ユーザー通信でインターコネクト(プライベート・ネットワーク)を使用しないでください。

インターコネクト上のインスタンス間通信用にユーザー・データグラム・プロトコル(UDP)またはリライアブル・データ・ソケット(RDS)・プロトコルのいずれかを使用するようにOracle RACを構成できます。Oracle Clusterwareは、UDPプロトコルを使用して同じインターコネクトを使用しますが、RDSを使用するように構成することはできません。

ネットワーク接続ストレージ(NAS)を使用する場合、追加のネットワーク接続が必要になります。ネットワーク接続ストレージは、NFSファイラなどの通常のNASデバイスに、またはFibre Channel over IPなどを使用して接続されるストレージにできます。この追加のネットワーク通信チャネルは、Oracle RAC (パブリックおよびプライベート・ネットワーク通信)で使用されている他の通信チャネルとは独立させる必要があります。他のネットワーク通信チャネルの1つを使用してストレージ・ネットワーク通信を収束する必要がある場合は、ストレージ関連の通信が1番目の優先順位を得るようにする必要があります。

Oracle Databasesに接続するための動的データベース・サービスの使用の概要

アプリケーションは、動的データベース・サービス機能を使用して、パブリック・ネットワークを介してOracle Databaseに接続する必要があります。

動的データベース・サービスでは、規則および特性を定義して、ユーザーおよびアプリケーションからデータベース・インスタンスへの接続方法を制御できます。これらの特性には、一意の名前、ワークロード・バランシング、フェイルオーバー・オプションおよび高可用性特性が含まれます。

ユーザーは、クライアント/サーバー構成を使用するか、または接続プーリングを任意に使用し、1つ以上の中間層を介してOracle RACデータベースにアクセスします。デフォルトでは、Oracle RACデータベースへのユーザー接続は、TCP/IPプロトコルを使用して確立されますが、他のプロトコルもサポートされています。Oracle RACデータベース・インスタンスには、クラスタのSCANを使用してアクセスする必要があります。

関連項目

- 動的データベース・サービスによる自動ワークロード管理の概要

仮想IPアドレスの概要

ノードVIPは、クライアントがOracle RACデータベースへの接続に使用する仮想IP (VIP)アドレスです。

Oracle Clusterwareは、パブリック・ネットワーク上のノードのVIPアドレスをホストします。データベース・クライアントからOracle RACデータベース・インスタンスへの通常の接続試行は、次のようにまとめることができます。

1. データベース・クライアントは、SCAN (パブリック・ネットワーク上のSCAN VIPを含む)に接続して、有効なサービス名をSCANリスナーに提供します。
2. 次にSCANリスナーは、このサービスをホストするデータベース・インスタンスを判別し、それぞれのノード上のローカル・リスナーまたはノード・リスナーにクライアントをルーティングします。
3. ノード・リスナーは、ノードVIPおよび特定のポートでリスニングして、接続リクエストを取得し、クライアントをローカル・ノードのインスタンスに接続します。

クラスタ上で複数のパブリック・ネットワークを使用して、複数のサブネットを介したクライアント接続をサポートする場合、サブネット内で前述の操作を実行します。

ノードで障害が発生した場合、VIPアドレスは、VIPアドレスがTCP接続を受け入れることができる別のノードにフェイルオーバーされますが、このノードはOracle Databaseへの接続は受け入れません。ホーム・ノードに存在しないVIPアドレスに接続を試行するクライアントは、TCP接続タイムアウト・メッセージを待機するかわりにrapid connection refusedエラーを受け取ります。VIPが構成されたネットワークがオンラインに戻ると、Oracle Clusterwareは、接続が受け入れられたホーム・ノードにVIPをフェイルバックします。通常、VIPアドレスは次の場合にフェイルオーバーされます。

- VIPアドレスが実行されているノードで障害が発生した場合
- VIPアドレスのすべてのインタフェースに障害が発生した場合
- VIPアドレスのすべてのインタフェースがネットワークから切断された場合

Oracle RACでは、異なるサブネットを介したクラスタへのアクセスを可能にする複数のパブリック・ネットワークをサポートしています。各ネットワーク・リソースは専用のサブネットを表し、各データベース・サービスは特定のネットワークを使用してOracle RACデータベースにアクセスします。各ネットワーク・リソースは、Oracle Clusterwareで管理されるリソースで、これにより、すでに説明したVIPの動作が可能になります。

SCANは、組織のドメイン・ネーム・サーバー(DNS)に、または3つのIPアドレスにラウンド・ロビンするグリッド・ネーミング・サービス(GNS)に定義された単一ネットワーク名です。Oracle RACデータベースへのすべての接続で、クライアント接続文字列にSCANを使用することをお薦めします。受信する接続は、3つのSCANリスナーを介して、要求されたサービスを提供するアクティブなインスタンス間でロード・バランシングされます。SCANを使用すると、クラスタの構成を変更(ノードの追加や削除)した場合にも、クライアント接続を変更する必要はありません。SCANでは複数のサブネットを完全にサポートしているため、クラスタを動作させるサブネットごとにSCANを1つ作成できます。

Oracle RACのサービス登録の制限

有効なノードの確認機能により、登録要求がリスナーにより許可される一連のIPアドレスまたはサブネットを構成および動的に更新する機能が提供されます。

リスナーへのデータベース・インスタンスの登録は、リクエスト元が有効なノードである場合にのみ成功します。ネットワーク管理者は、有効なノードおよび除外ノードのリストを指定したり、有効なノードの確認を完全に無効にすることができます。有効なノードのリストでは、データベースに登録できるノードやサブネットを明示的にリストします。除外ノードのリストでは、データベースに登録できないノードを明示的にリストします。動的登録を制御することによって、Oracle RACデプロイメントの管理性およびセキュリティが向上します。

デフォルトでは、SCANリスナー・エージェントはREMOTE_ADDRESS_REGISTRATION_listener_nameをプライベートIPエンドポイントに設定します。SCANリスナーは、プライベート・ネットワークからの登録要求のみを受け入れます。SCANリスナーのプライベート・ネットワークにアクセスできないリモート・ノードは、listener.oraファイルのregistration_invited_nodes_aliasパラメータを使用して、またはコマンドライン・インタフェースのSRVCTLを使用してSCANリスナーを変更して、有効なノードのリストに含める必要があります。

ノート:

Oracle Grid Infrastructure 12c 以降、SCAN リスナーについて、VALID_NODE_CHECKING_REGISTRATION_listener_name および REGISTRATION_INVITED_NODES_listener_name パラメータが listener.ora ファイルに設定されている場合、リスナー・エージェントはこれらのパラメータを上書きします。

SRVCTLユーティリティを使用してinvitednodes値とinvitedsubnets値を設定すると、リスナー・エージェントは自動的にVALID_NODE_CHECKING_REGISTRATION_listener_nameをSUBNETに設定し、REGISTRATION_INVITED_NODES_listener_nameをlistener.oraファイルで指定されたリストに設定します。

CRSによって管理されるその他のリスナーの場合、リスナー・エージェントは、listener.oraファイルでまだ設定されていない場合にのみ、listener.oraファイルでVALID_NODE_CHECKING_REGISTRATION_listener_nameをSUBNETに設定します。SRVCTLユーティリティでは、SCAN以外のリスナーについてinvitednodes値とinvitedsubnets値の設定はサポートされていません。リスナー・エージェントは、SCAN以外のリスナーについてlistener.oraファイルのREGISTRATION_INVITED_NODES_listener_nameを更新しません。

Oracle RACソフトウェア・コンポーネント

通常、Oracle RACデータベースには、それぞれにメモリー構造およびバックグラウンド・プロセスが含まれる複数のデータベース・インスタンスがあります。

Oracle RACデータベースには、非クラスタOracle Databaseと同じプロセスおよびメモリー構造があり、さらに、Oracle RAC固有の追加プロセスおよびメモリー構造があります。1つのインスタンスのデータベース・ビューは、同じOracle RACデータベース内の他のインスタンスのビューとほぼ同じで、ビューはその環境の単一のシステム・イメージです。

各インスタンスのシステム・グローバル領域(SGA)には、バッファ・キャッシュが存在します。キャッシュ・フュージョンの使用によって、Oracle RAC環境で各インスタンスのバッファ・キャッシュが論理的に結合され、論理的に結合された単一のキャッシュにデータが存在する場合と同様に、インスタンスでデータを処理できます。

ノート:

- インメモリー・トランザクション・マネージャはキャッシュ・フュージョン・プロトコルと統合しています。
- キャッシュ・フュージョンによって、Oracle RAC での SGA のサイズ要件は、非クラスタ Oracle Database での SGA のサイズ要件より大きくなります。

問合せまたはトランザクションを正しく処理するために必要なブロックを、各Oracle RACデータベース・インスタンスが確実に取得できるようにするために、Oracle RACインスタンスでは、グローバル・キャッシュ・サービス(GCS)およびグローバル・エンキュー・サービス(GES)の2つのプロセスが使用されます。GCSおよびGESは、グローバル・リソース・ディレクトリ(GRD)を使用して、各データ・ファイルおよび各キャッシュ・ブロックのステータスのレコードをメンテナンスします。GRDの内容はすべてのアクティブ・インスタンス間で分散され、Oracle RACインスタンスのSGAのサイズが実質的に増加します。

1つのインスタンスがデータをキャッシュした後は、同じクラスタ・データベース内の他のインスタンスは、ディスクからブロックを読み取るよりも高速に、同じデータベース内の別のインスタンスからブロック・イメージを取得できるようになります。そのため、キャッシュ・フュージョンは、ディスクからブロックを再読取りするのではなく、現行のブロックをインスタンス間で移動します。一貫性のあるブロックや変更されたブロックが別のインスタンスで必要な場合、キャッシュ・フュージョンは、影響を受けるインスタンス間でブロック・イメージを直接転送します。Oracle RACは、インスタンス間通信およびブロック転送にプライベート・インターコネクトを使用します。GES Monitorおよびインスタンス・エンキュー・プロセスは、キャッシュ・フュージョン・リソースへのアクセスおよびエンキューのリカバリ・プロセスを管理します。

キャッシュ・フュージョンでは、プライベート・ネットワークの待機時間およびディスク上のサービス時間を監視して、最適なパスを自動的に選択します。共有ディスクに待機時間が短いSSDがある場合は、最適なパスが自動的に選択されます。

関連項目

- 『Oracle Database In-Memoryガイド』

Oracle RACバックグラウンド・プロセス

グローバル・キャッシュ・サービス(GCS)および**グローバル・エンキュー・サービス**(GES)プロセスは、**グローバル・リソース・ディレクトリ**(GRD)と連携してキャッシュ・フュージョンを有効にします。

Oracle RACプロセスおよびその識別子は次のとおりです。

- ACMS: メモリー・サービスへのアトミック制御ファイル(ACMS)

  Oracle RAC環境では、各インスタンスのACMSのプロセスは、分散SGAメモリー更新が成功時にグローバルにコミットされるようにしたり、障害発生時にグローバルに終了されるようにするエージェントです。

- GTX0-j: グローバル・トランザクション・プロセス

  GTX0-jプロセスは、Oracle RAC環境でXAグローバル・トランザクションを透過的にサポートします。これらのプロセスの数は、データベースによって、XAグローバル・トランザクションのワークロードに基づいて自動調整されます。

- LMON: グローバル・エンキュー・サービス・モニター

  LMONプロセスでは、グローバル・エンキューおよびクラスタ全体のリソースが監視され、グローバル・エンキュー・リカバリ操作

が実行されます。

- LMD: グローバル・エンキュー・サービス・デーモン

  LMDプロセスでは、各インスタンス内の受信リモート・リソース要求が管理されます。

- LMS: グローバル・キャッシュ・サービス・プロセス

  LMSプロセスでは、情報を**グローバル・リソース・ディレクトリ**(GRD)に記録することにより、データファイルのステータスおよび各キャッシュ・ブロックのレコードがメンテナンスされます。LMSプロセスでは、リモート・インスタンスへのメッセージ・フローの制御、グローバル・データ・ブロック・アクセスの管理、異なるインスタンスのバッファ・キャッシュ間のブロック・イメージの送信も行われます。この処理は、キャッシュ・フュージョンの一部です。

- LCK0: インスタンス・エンキュー・プロセス

  LCK0プロセスでは、ライブラリや行キャッシュ要求などの非キャッシュ・フュージョン・リソース要求が管理されます。

- RMSn: Oracle RAC管理プロセス(RMSn)

  RMSnプロセスでは、Oracle RACの管理性タスクが実行されます。RMSnプロセスによって実行されるタスクには、新規インスタンスがクラスタに追加された際のOracle RAC関連リソースの作成があります。

- RSMN: リモート・スレーブ・モニターは、リモート・インスタンスでのバックグラウンド・セカンダリ・プロセスの作成と通信を管理します。これらのバックグラウンド・セカンダリ・プロセスでは、別のインスタンスで実行されている調整プロセスのためのタスクが実行されます。

ノート:

この項で説明する多くの Oracle Database コンポーネントは、『*Oracle Database* 概要』で説明するシングル・インスタンスの Oracle Database の追加コンポーネントです

関連項目

- Oracle Database概要

動的データベース・サービスによる自動ワークロード管理の概要

サービスは、共通の属性、パフォーマンスしきい値および優先度を持つアプリケーションのグループを表します。

アプリケーション機能は、サービスによって識別されるワークロードに分割できます。たとえば、Oracle E-Business Suiteでは、総勘定元帳、売掛金勘定、受注など、職務ごとにサービスを定義できます。サービスはOracle Databaseの1つ以上のインスタンス、グローバル・クラスタ内の複数のデータベースにわたることができ、単一インスタンスで複数のサービスをサポートできます。サービスを提供するインスタンスの数は、アプリケーションに対して透過的です。サービスは、競合するアプリケーションを管理する単一のシステム・イメージを提供し、各ワークロードを1つの単位として管理できるようにします。

中間層アプリケーションおよびクライアントでは、サービス名をTNS接続文字列内の接続の一部として指定することで、サービスを選択します。たとえば、Oracle WebLogic Serverのデータ・ソースは、サービスにルーティングするように設定されます。Net Easy*Connectionを使用する場合、この接続は、user_name/password@SCAN/service_nameのように、サービス名とネットワーク・アドレスだけで構成されます。Oracle Scheduler、パラレル問合せ、Oracle GoldenGateキューなどのサーバー側の作業では、ワークロード定義の一部としてサービス名を設定します。Oracle Schedulerの場合、ジョブがジョブ・クラスに割り

当てられ、サービス内で複数のジョブ・クラスを実行できます。パラレル問合せとパラレルDMLの場合、問合せコーディネータはサービスに接続し、パラレル問合せスレーブはパラレル実行中そのサービスを継承します。Oracle GoldenGateの場合、ストリーム・キューはサービスを使用してアクセスされます。サービス下で実行される作業は、そのサービスのしきい値および属性を継承し、サービスの一部として測定されます。

Oracle Database Resource Managerでは、サービスをコンシューマ・グループおよび優先度にバインドします。これによって、データベースはサービスをその重要性の順に管理できます。たとえば、DBAでは、優先度の高いオンライン・ユーザー向けと、優先度の低い内部レポート・アプリケーション向けのサービスを個別に定義できます。同様に、DBAでGold、SilverおよびBronzeのサービスを定義して、同じアプリケーションの要求に対してサービスを提供する順番に優先度を付けることができます。システムのサービスを計画する場合、その計画には、他のサービスに対する相対的な各サービスの優先度が含まれている必要があります。このようにして、Oracle Database Resource Managerは優先度が1位のサービス、次に優先度2位のサービス、というように対処できます。

ユーザーまたはアプリケーションがデータベースに接続するときは、接続文字列のCONNECT_DATA部分に指定されたサービスを使用することをお薦めします。Oracle Databaseでは、データベースが作成されると自動的に1つのデータベース・サービスが作成されますが、このサービスの動作は、その後自分で作成するデータベース・サービスの動作とは異なります。データベースを使用したワークロード管理の柔軟性を高めるために、Oracle Databaseでは、複数のサービスを作成し、どのインスタンス(またはサービス・プール)でサービスが起動されるかを指定できます。より柔軟なワークロード管理が必要な場合は、この章を読み進めると、サービスで使用できる追加機能を理解できます。

ノート:

この章で説明する機能は、デフォルトのデータベース・サービス(DB_NAME、DB_UNIQUE_NAME、PDB_NAME、SYS$BACKGROUND および SYS$USERS)では機能しません。これらのサービスを、データベースに接続するアプリケーションに使用しないことをお薦めします。このような機能を活用するには、クラスタ管理サービスを作成する必要があります。自分が作成したサービスのみ管理できます。データベースによって自動的に作成されたサービスはデータベース・サーバーによって管理されます。

動的データベース・サービス

動的データベース・サービスによってワークロードの分散を管理し、ユーザーおよびアプリケーションに対してパフォーマンスを最適化できます。動的データベース・サービスは、次の機能を提供します。

- サービス: 企業のグリッド構想を可能にするために、Oracle Databaseでは、サービスと呼ばれる強力な自動ワークロード管理機能が導入されています。サービスは、Oracle RACデータベースで定義できるエンティティで、これを使用してデータベース・ワークロードをグループ化し、サービスの提供を割り当てられている最適なインスタンスに作業をルーティングし、計画済および計画外のアクションの高可用性を実現できます。
- 高可用性フレームワーク: Oracle Databaseでコンポーネントを常に稼働状態に維持できるOracle RACコンポーネント。
- 高速アプリケーション通知(FAN): インスタンス、サービスまたはノードのUPやDOWNイベントなどのクラスタ状態の変更およびロード・バランシング・アドバイザのイベントについての情報をOracle RACアプリケーションおよびクライアントに提供します。FANには、クライアントにイベントを発行する方法が2つあり、1つは、Oracle Notification Serviceデーモン

で、Oracle Application Serverを含むJava Database Connectivity (JDBC)クライアントによって使用され、もう1つは、Oracle GoldenGateアドバンスト・キューイングで、以前のリリースのOracle Call Interface (OCI)およびOracle Data Provider for .NET (ODP.NET)クライアントによってのみ使用されます。

ノート:

Oracle Database 12c リリース 2 (12.2)からは、すべてのクライアントが Oracle Notification Service を使用します。

- トランザクション・ガード: 計画外停止および重複送信の場合に、トランザクションの実行を1回以下にするためのプロトコルおよびAPIを提供するツール。
- アプリケーション・コンティニュイティ: リカバリ可能なエラーが発生した場合に、多くのシステム、通信、記憶域の停止およびハードウェア障害をマスクして、処理中の要求をリプレイする汎用インフラストラクチャを提供します。既存のリカバリ・テクノロジとは異なり、この機能では、アプリケーション下でトランザクションおよび非トランザクション・セッション状態をリカバリしようとするため、停止はアプリケーションにとって実行の遅延のように見えます。
- コネクション・ロード・バランシング: 要求されたデータベース・サービスを提供するすべてのインスタンスで、受信する接続を均等に分散するOracle Net Servicesの機能。
- ロード・バランシング・アドバイザ: データベースとそのインスタンスが提供する現在のサービス・レベルについてアプリケーションに情報を提供します。ロード・バランシング・アドバイザは、サービス用に定義した管理ポリシーに基づいて最適なサービスを得るために、アプリケーション・リクエストの宛先に関する推奨事項をアプリケーションに提供します。ロード・バランシング・アドバイザのイベントは、Oracle Notification Serviceを介してパブリッシュされます。
- 自動ワークロード・リポジトリ(AWR): サービス・レベルの統計をメトリックとして追跡します。サーバーによって生成されるアラートは、特定のしきい値を超えたり、必要なしきい値に達しない場合、これらのメトリックに対して作成されます。
- 高速接続フェイルオーバー(FCF): これは、FANイベントをサブスクライブすることによって、高速な接続のフェイルオーバーを提供する、Oracleクライアントの機能です。
- ランタイム接続ロード・バランシング: アプリケーションが接続にいくつかの作業を完了するように要求すると、データベース・インスタンスによって提供されている現在のサービス・レベルに基づいて、接続プールで高度な接続の割当てを行うOracleクライアントの機能です。
- 単一クライアント・アクセス名(SCAN): Oracle RACに接続しているクライアントに単一の名前を提供し、この名前は、クラスタのノードを追加または削除しても、クラスタの存続期間中は変更されません。SCANに接続しているクライアントでは、Thin JDBC URLやEZConnectなどの単一の接続文字列を使用でき、ロード・バランシングおよびクライアントの接続フェイルオーバーが実現されます。

Oracle RACおよび非クラスタOracle Database環境をデプロイして、多くの異なる方法で動的データベース・サービス機能を使用できます。ノード数および使用環境の使用目的と複雑さによって異なりますが、最適な自動ワークロード管理および高可用性構成は、この章で説明する考慮事項を検討して決定してください。

関連項目

- Oracle Database管理者ガイド
- 動的データベース・サービスによるワークロード管理

サーバー・プールおよびポリシー管理データベースの概要

サーバー・プールは、ポリシー管理データベースの基盤です。

次のデプロイメント・モデルを使用すると、マルチノードであれ、Oracle Real Application Clusters One Node (Oracle RAC One Node)であれ、Oracle RACデータベースを作成できます。

- 管理者管理デプロイメントは、Oracle Database 11gリリース2 (11.2)の前に存在していたOracle RACデプロイメント・タイプに基づき、クラスタ内の特定のノードで実行されるように各データベース・インスタンスを静的に構成する必要があり、また、preferredおよびavailable宛先を使用して、特定のデータベースに属する特定のインスタンスで実行されるようにデータベース・サービスを構成する必要があります。
- ポリシー管理デプロイメントは、サーバー・プールに基づき、この場合、データベース・サービスは、サーバー・プール内でシングルトンまたは均一として、サーバー・プール内のすべてのサーバーにわたって実行されます。データベースは1つ以上のサーバー・プールにデプロイされ、サーバー・プールのサイズによってデプロイメント内のデータベース・インスタンスの数が決まります。

関連項目

- Oracle Database Quality of Service Managementユーザーズ・ガイド

サーバー・プールの概要

サーバー・プールは、データベース・サービスまたはアプリケーション・サービスを提供するサーバーのグループにクラスタを論理的に割り当てます。

サーバー・プールのプロパティは、これらのデータベースおよびアプリケーションのスケーラビリティおよび可用性を制御します。各サーバー・プールは最小サイズおよび最大サイズを使用して構成でき、これによって、スケーラビリティが決まります。Oracle Clusterwareは、サーバー・プール間の可用性を管理し、個々のサーバー・プールの重要度の値を構成することによって可用性をさらに調整できます。

サーバーは名前によってサーバー・プールに割り当てられるのではなく、番号によって割り当てられます。したがって、任意のデータベースを実行するように任意のサーバーを構成する必要があります。たとえば、異機種サーバーやストレージ接続のためにサーバーを構成できない場合は、サーバー・カテゴリ定義を使用してサーバー・プールのメンバーシップ適格性を判別することによってサーバーを制限できます。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

サーバー・プールの使用例

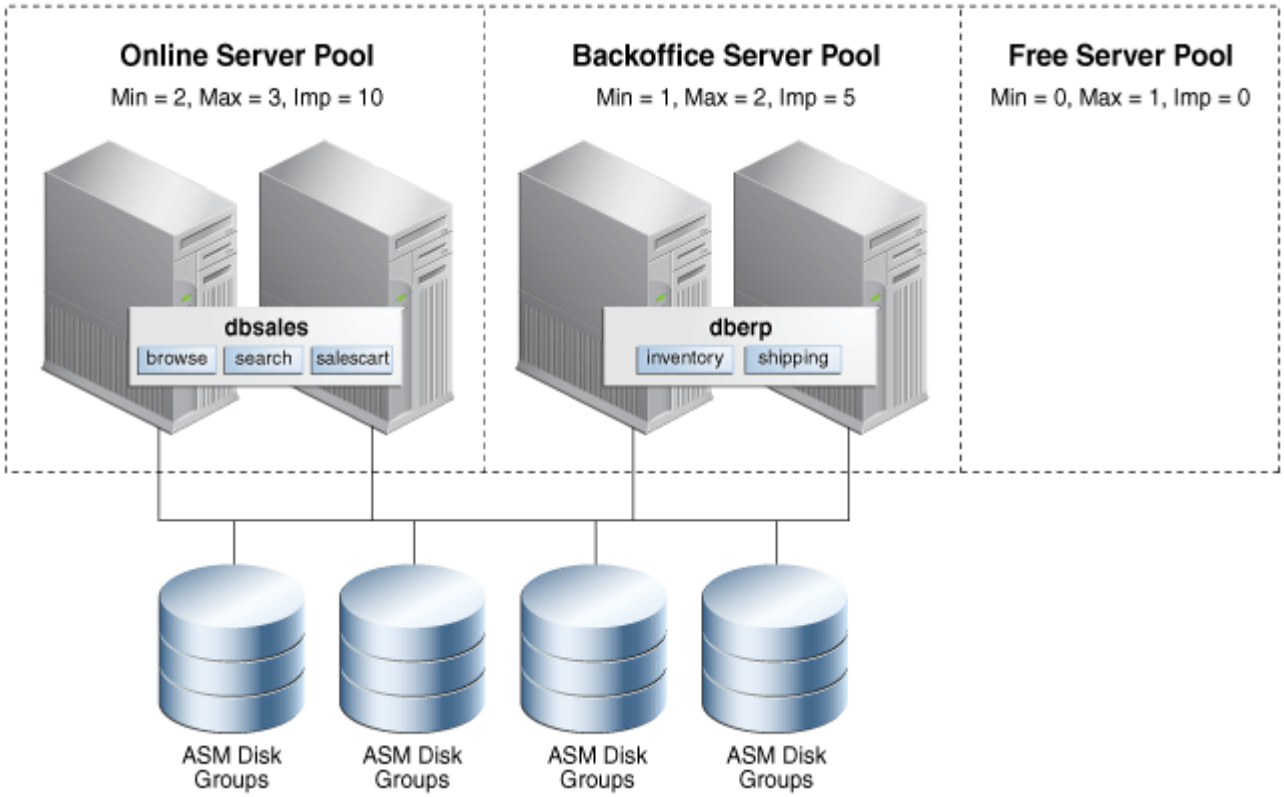
この項では、次のサーバー・プールの使用例を示します。

- サーバーの最小数および最大数
- サーバー・プールのIMPORTANCE属性
- データベースの統合

サーバーの最小数および最大数

onlineとbackofficeという2つのサーバー・プールに4ノード・クラスタが構成されているとします。onlineサーバー・プールではdbsalesというデータベースが実行され、browse、searchおよびsalescartサービスを提供しています。backofficeサーバー・プールではdberpというデータベースが実行され、図1-2に示すとおり、inventoryおよびshippingサービスを提供しています。通常の営業時間中、企業は通常の要求に対応するために、dbsalesデータベースの最低2つのインスタンスと、dberpデータベースの1つのインスタンスを必要とします。

図1-2 最小数および最大数によるサーバーの配置



このポリシー管理デプロイメントでは、onlineサーバー・プールのMIN_SIZEサーバー・プール属性の値は2で、backofficeサーバー・プールのMIN_SIZEサーバー・プール属性の値は1です。このように構成されたOracle Clusterwareでは、常にonlineサーバー・プールに2つのサーバーがあり、backofficeサーバー・プールに1つのサーバーがあります。これは4ノード・クラスタであるため、いずれのサーバー・プールにも割り当てられていないサーバーが1つ残ります。最後のサーバーがデプロイされる場所は、各サーバー・プールのMAX_SIZEサーバー・プール・パラメータによって決まります。各サーバー・プールのMAX_SIZEサーバー・プール属性の値の合計がクラスタ内のサーバーの総数より小さい場合、残りのサーバーは、デプロイされたノードの障害を待つ空きサーバー・プールにとどまります。

MAX_SIZEの値がMIN_SIZEの値より大きい場合、図1-2で示すとおり、また、次の項で詳細に説明するとおり、残りのサーバーは、重要度の値が最大のサーバー・プールにデプロイされます。この場合、サーバーは、サーバーが必要とされるサーバー・プールを結合するためにオンラインで再配置できる共有可能なリソースになります。たとえば、営業時間中は、サーバーをonlineサーバー・プールに与えて、dbsalesデータベースのインスタンスを追加できますが、営業時間後は、backofficeサーバー・プールに再配置して、dberpデータベース・インスタンスを追加できます。このような動作はすべてオンラインであり、インスタンスはトランザクション的に停止されます。

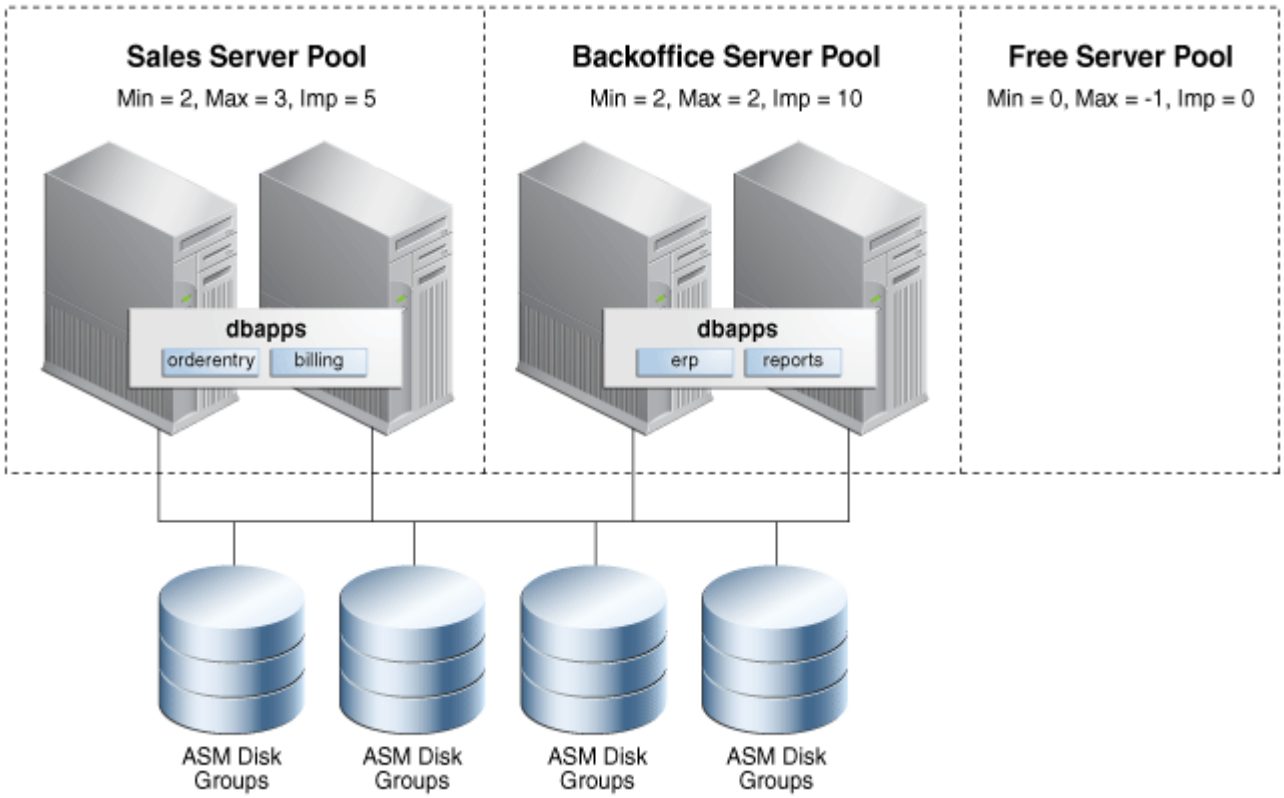
これらの2つのポリシー管理データベースは、必要とされるインスタンスのみを実行し、要求またはビジネス要件を満たすように動的に増減できます。

サーバー・プールのIMPORTANCE属性

IMPORTANCEサーバー・プール属性は、クラスタの起動時、およびノードの障害または削除に応じて使用されます。管理者管理データベースとは対照的に、最初に起動するデータベースを決めるように、また、マルチノードの停止時にオンラインのままにしておくデータベースを決めるように、様々な重要度でサーバー・プールを構成できます。

salesおよびbackofficeという2つのサーバー・プール内で、dbappsというデータベースをホストする4ノード・クラスタを考えてみます。図1-3に示すとおり、2つのサービスorderentryおよびbillingは、salesサーバー・プールで実行され、他の2つのサービスerpおよびreportsは、backofficeサーバー・プールで実行されます。backofficeサーバー・プールの値より大きいsalesサーバー・プールのIMPORTANCEサーバー・プール属性の値を構成することによって、マルチノードの障害の後に残って実行されているサーバーが1つのみでも、クラスタが起動すると、salesのサービスは最初に起動して常に使用可能になります。IMPORTANCEサーバー・プール属性を使用すると、サービスをランク付けすることができ、常に使用可能にするためにクラスタ内のすべてのノードでサービスを実行する必要もなくなります。

図1-3 サーバー・プールの重要度



データベースの統合

いくつかの様々なアプローチを個別にまたは組み合せて使用すると、Oracle Databasesを統合できます。ポリシー管理デプロイメントでは統合は容易です。スキーマ統合の場合、複数のアプリケーションは、個別スキーマまたはプラガブル・データベース(PDB)に分離された単一データベースによってホストされているため、サーバー・プールを使用して必要な容量を満たすことができます。サーバー・プールの動的スケーリング・プロパティのため、現在の要求またはビジネス要件に合うようにデータベース・インスタンスの数を増減できます。サーバー・プールによって、一緒にまたは別個に実行されるサービスも決定されるため、必要なアフィニティまたは分離を構成および維持できます。

たとえば、バージョン要件のためにスキーマ統合を使用できない場合、単一セットのサーバー上で複数のデータベースをホストで

きます。ポリシー管理データベースを使用すると、ポリシー管理データベースはインスタンス・ケージングを利用することによって同じサーバー・プールを共有できるため、このデータベース統合が容易になります。これにより、要求またはビジネス・ポリシーとスケジュールに合うように、水平方向(サーバー・プール・サイズを使用)と垂直方向(CPU_COUNTサーバー構成属性を使用)の両方にデータベースを動的に増減できます。

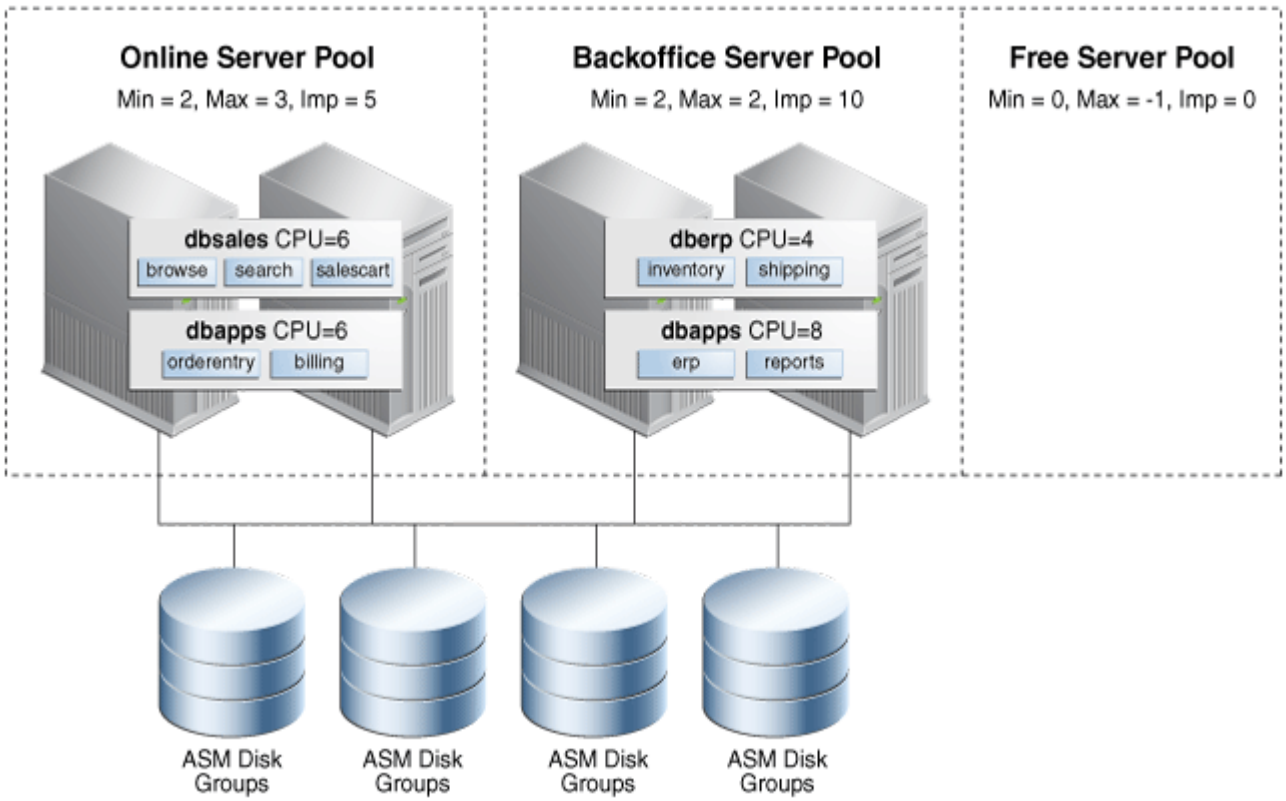
対照的に、管理者管理データベースでは、データベース・インスタンスまたはサーバーに障害が発生した場合、ワークロードのフェイルオーバーを吸収するために各サーバーに容量を確保する必要があります。ただし、ポリシー管理データベースの場合、MIN_SIZE、MAX_SIZEおよびIMPORTANCEサーバー・プール属性を使用して、実行中のワークロードのビジネスの必要性別にサーバー・プールを効果的にランク付けできます。

サーバーの障害によってサーバー・プールがサーバーの構成済最小数を下回ると、重要性の少ないサーバー・プールからの別のサーバーがそのかわりにるため、サーバーの数は構成済最小数に戻ります。これにより、残りのサーバーのオーバーロードによるカスケード障害のリスクがなくなり、処理障害のために容量を確保する必要性を大幅に減らしたり、その必要性をなくすことさえできます。

ポリシー管理データベースに移行または変換すると、クラスタの統合も可能になり、より大きなクラスタを作成することで、データベースのホストおよび拡張に使用可能なサーバー数が増加するため可用性およびスケーラビリティの向上が実現されます。ポリシー管理データベースでは、インスタンス名を特定のサーバーにバインディングすること、およびサービスを特定のインスタンスにバインディングすることが不要なため、大きなクラスタを構成および管理する際の複雑さが大幅に軽減されます。

図1-4に、デプロイメント例を示します。ここでは、前の2つのクラスタ例(図1-2および図1-3)は、単一クラスタに統合され、構成されたデータベース統合(インスタンス・ケージングを使用)とクラスタ統合(サーバー・プールを使用)の両方を利用して、ワークロードのサイズ設定および優先度付けが適切に行われるようにします。

図1-4 データベースの統合



関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

ポリシー管理データベースのデプロイ

ポリシー管理データベースをデプロイする場合、サービスはサーバー・プールにわたらないことを考慮に入れて、まずサービスとその必要なサイズ設定を決定する必要があります。

このデータベースを他のデータベースと同じ場所に配置する場合、ホストされている他のデータベースに関するCPU要件を考慮する必要があり、インスタンス・ケージングのCPU_COUNT属性の値も考慮する必要があります。これにより、1つ以上のサーバー・プールで垂直方向と水平方向の両方にデータベースのサイズを指定できます。

このデータベースのサーバー・プールを他のサーバー・プールと同じ場所に配置する場合、会議の要求およびビジネス要件を最適化するためにカレンダまたはイベントに基づいてサーバー・プール・サイズを調整するようにサーバー・プールを構成することを検討します。サーバー・プールのサイズを決定し、MIN_SIZEおよびMAX_SIZEサーバー・プール属性の適切な値を構成したら、各サーバー・プールの相対的な重要度を決定できます。

クラスタ管理者として、srvctl add serverpoolコマンドを使用してポリシー管理データベース・サーバー・プールを作成します。Oracle Grid Infrastructureホームで、srvctl modify serverpoolコマンドを使用すると、サーバー・プールのプロパティを変更できます。

DBCAを使用してサーバー・プールを作成できる間は、小さい単一サーバー・プールのデプロイメント用のもののみをお薦めします。サーバーがすでに他のサーバー・プールに割り当てられている場合、DBCAに障害が発生するためです。また、クラスタが、古いサーバーや新しいサーバーなど、異なる容量のサーバーで構成されている場合、各サーバー・プールを結合するためのサーバーの最小サーバー要件を定義するサーバー・カテゴリ定義を設定することをお薦めします。

サーバー・プールの作成後、適切なデータベース・ホームからDBCAを実行できます。データベースのタイプおよびタスクに応じて、様々なデフォルト・オプションが提示されます。コンテナ・データベース(CDB)を含め、すべての新しいOracle RACおよびOracle RAC One Nodeデータベースの場合、ポリシー管理オプションはデフォルトで、これがお薦めするオプションです。

管理者管理データベースから、またはOracle Database 11gリリース2 (11.2)より前のデータベースからデータベースをアップグレードする場合、ポリシー管理データベースに直接アップグレードするオプションはありません。ただし、アップグレード後、srvctl modify databaseコマンドを使用して、データベースをポリシー管理に変換できます。

管理者管理データベースからポリシー管理データベースに変換する場合、インスタンス名は、アンダースコアを含むように自動的に更新されます(たとえば: orcl1はorcl_1になります)。サーバー・プールのサイズが大きくなるときデータベースが自動的にインスタンスを作成できるようにするためには、アンダースコアが必要です。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle Clusterware管理およびデプロイメント・ガイド
- 管理者管理データベースのポリシー管理データベースへの変換

ポリシー管理データベースの管理

ポリシー管理データベースの管理では、作成、サイズ設定、パッチ適用およびロード・バランシングに関して、管理者管理データベースより構成および再構成のステップが少なくて済みます。

また、クラスタ内のサーバー・プールのサーバーはどのデータベースでも実行できるため、データベース・インスタンスとノード名のマッピングを作成して維持する必要はありません。ただし、データベースが特定のノードで実行するときには必ず特定のインスタンス名を使用するようにする場合は、srvctl modify instance -db db_unique_name -instance inst_name -node node_nameコマンドを使用して、インスタンスからノード名へのマッピングを作成できます。これは、特定のノードのスクリプトで固定のORACLE_SID値を使用してデータベースに接続する場合に役立ちます。

サーバーを空きプールに再配置することによって、またはサーバー・プールの最小サイズと最大サイズを調整することによって、パッチ適用などのメンテナンス・タスクを実行できるため、必要な可用性が維持されます。

ポリシー管理データベースでは、サービスの管理も容易になります。ポリシー管理データベースは、単一サーバー・プールに割り当てられ、プール内のすべてのサーバーにわたってシングルトンまたは均一として実行されるためです。サービスごとに明示的に優先かつ使用可能なデータベース・インスタンス・リストを作成または維持する必要はなくなりました。手動による再配置または高可用性イベントのためサーバーをサーバー・プールに移動すると、統一されたすべてのサービスおよびその依存データベース・インスタンスは自動的に起動します。1つ以上のシングルトン・サービスをホストしているサーバーがダウンした場合、これらのサービスは、サーバー・プール内の残りの1つ以上のサーバーで自動的に起動します。Oracle RAC One Nodeの場合、対応するデータベース・インスタンスも自動的に起動します。

相互に関連するサービスの管理は、各サーバー・プールの重要度属性を利用することによって向上します。サーバー・プールで実行される各サービスは、クラスタ内で他のサーバー・プールにホストされているサービスに関連するサーバー・プールの重要度を継承します。最も重要なサーバー・プールの最小サイズが0(ゼロ)より大きい場合、このサーバー・プール内のサービスおよび関連のデータベース・インスタンスは、クラスタの起動時に最初に起動し、クラスタ内に1つのサーバーが実行されているかぎり、実行されている最後のサービスおよびデータベース・インスタンスになります。重要でないサービスは最も重要でないサーバー・プールのビジネスに提供でき、要求または障害のために十分なリソースが使用できない場合、これらのサービスは最終的に停止され、よりビジネスクリティカルなサービスが使用可能なまま存続するようにできます。

数多くの管理タスクは、統合環境内の複数のデータベース、サービスまたはサーバー・プールに影響を与える可能性のある変更を伴うことがあるため、特定のSRVCTLコマンドの評価モードを使用して、コマンドのリソース影響のレポートを取得できます。

サーバー・プールを変更するシステムに対する影響を評価する次の例を考えてください。

```
$ srvctl modify srvpool -l 3 -g online -eval
Service erp1 will be stopped on node test3
Service reports will be stopped on node test3
Service inventory will be stopped on node test3
Service shipping will be stopped on node test3
Database dbsales will be started on node test3
Service orderentry will be started on node test3
Service billing will be started on node test3
Service browse will be started on node test3
Service search will be started on node test3
Service salescart will be started on node test3
Server test3 will be moved from pool backoffice to pool online
```

前述の例に示すとおり、サーバー・プールを変更すると、数多くのリソース状態の変更を招くことがあります。Oracle ClusterwareまたはOracle Database Quality of Service Managementを介してポリシー・セットを使用できます。

関連項目

- srvctl modify srvpool
- srvctl relocate server
- SRVCTL使用情報
- Oracle Clusterware管理およびデプロイメント・ガイド

ポリシー・ベースのクラスタ管理

Oracle Clusterwareでは、ネイティブなOracle Clusterware機能としてクラスタ構成ポリシー・セットの管理をサポートします。

クラスタ構成ポリシーには、システム内で定義された、サーバー・プールごとに1つの定義が含まれます。また、クラスタ構成ポリシーでは、リソース配置およびクラスタ・ノードの可用性についても指定します。クラスタ構成ポリシー・セットでは、クラスタ内に構成されるすべてのサーバー・プール名が定義され、1つ以上の構成ポリシーが含まれます。

一度に1つの構成ポリシーのみが常に有効になります。ただし、管理者は通常、カレンダ日付または時間パラメータに基づいて様々なビジネス・ニーズおよび要求を反映させるために、いくつかの構成ポリシーを作成します。たとえば、一般的に、就業日の午前中は多くのユーザーがログインし、電子メールをダウンロードしますが、夜間や週末は電子メール関連のワークロードが通常は少なくなります。このような場合、クラスタ構成ポリシーを使用して、予想される需要に基づいてサーバー割当てを定義します。より具体的には、この例の場合、より多くのサーバーをOLTPワークロードに割り当てる構成ポリシーを就業日の午前中に有効にし、週末および就業日の夜には、別の構成ポリシーで、より多くのサーバーをバッチ・ワークロードに割り当てます。

クラスタ構成ポリシーを使用すると、様々なコンピュータやメモリー・サイズ(異機種)など、様々な機能のサーバーを構成するクラスタの管理に役立てることもできます。異機種サーバー・タイプで構成されるクラスタ用に管理および可用性ポリシーを作成するために、クラスタ管理者は、サーバー属性に基づいてサーバー・カテゴリを作成できます。これらの属性によって、どのサーバーをどのサーバー・プールに割り当てることができるかを制限できます。たとえば、古いハードウェアを実行するサーバーがクラスタ内に存在する場合に、それらのサーバーを、オンライン販売やその他の業務上重要なアプリケーションに使用されるサーバー・プールではなく、バッチ・ジョブやテストをサポートするサーバー・プールにのみ割り当てるように指定する属性を使用できます。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

Oracle Database Quality of Service Managementの概要

Oracle Database Quality of Service Management (Oracle Database QoS Management)は、システム全体のワークロード・リクエストを監視する自動化されたポリシーベースの製品です。

Oracle Database QoS Managementは、アプリケーション間で共有されるリソースを管理し、システム構成を調整して、アプリケーションの実行をビジネスに必要なパフォーマンス・レベルに維持します。Oracle Database QoS Managementは、システム構成および要求で発生した変更に対応し、アプリケーションのパフォーマンス・レベルがそれ以上変動することを回避します。

Oracle Database QoS Managementは、Oracle RACデータベースのワークロード・パフォーマンスの目標を監視および管理します。そのために、この目標に影響を与えているボトルネック・リソースを特定し、パフォーマンスをリストアするためのアクション推奨して実施します。管理者管理デプロイメントでは、データベース・インスタンスをノードにバインドしますが、ポリシー管理デプロイメントでは、このようなバインドは実施しません。そのため、Oracle Database QoS Managementサーバー・プール・サイズのリソース制御は後者でのみ使用できます。その他すべてのリソース管理制御は、どちらのデプロイメントでも使用できます。

Oracle Database QoS Managementは、測定のみ、モニター、および管理の各モードによって、管理者管理Oracle RACデータベースとOracle RAC One Nodeデータベースをサポートします。これにより、データベースで実行中のパフォーマンス・クラスのCPU共有を調整することで、管理者管理Oracle RACデータベース内でのスキーマ統合のサポートが可能になります。さらに、同じ物理サーバーでホストされているデータベースのCPU数を調整することで、データベース統合がサポートされています。

管理者管理データベースはサーバー・プールで実行しないため、サーバー・プール・サイズを変更することでインスタンスの数を増減する機能は、ポリシー管理データベースのデプロイメントでサポートではサポートされていますが、管理者管理データベースでは利用できません。この新しいデプロイメントのサポートは、Oracle Enterprise Manager Cloud ControlのOracle QoS管理のページに統合されています。

ハング・マネージャの概要

ハング・マネージャは、システムのハングアップを自動的に検出して解決するOracle Databaseの機能です。

ハング・マネージャは、Oracle Database 11g リリース1 (11.1)で初めて使用できるようになりました。当初は、システムのハングを識別して、ハングについての関連情報をトレース・ファイルにダンプしていました。Oracle Database 12c リリース2 (12.2)では、ハング・マネージャはシステム・ハングに対処できるようになり、その解決を試行します。ハング・マネージャは、単一インスタンスとOracle RACデータベース・インスタンスのどちらでも実行します。

ハング・マネージャの機能は、次のとおりです。

- 最初にシステム・ハングを検出し、そのハングを分析してから、ハングの原因を確認します。その後、ハングを解決するための対処方針を決定するためにヒューリスティックを適用します。
- My Oracle Supportにトレース・ファイルを提出してハングの原因の特定を依頼するために、DBAによる手動のステップが必要だったタスクを自動化し、データベースとアプリケーションのダウンタイムを最小化または排除します。
- すべてのプロセスを定期的にスキャンして、連続するスキャンでリソースを保持しているプロセスの小さなサブセットを分析します。ハング・マネージャは、リソースを待機しているものがない場合はプロセスを無視します。
- インスタンス間のハング(Oracle ASMインスタンスからの応答を待機しているデータベース・プロセスがホルダーの場合のハング)を考慮します。
- リーダー・ノード・インスタンスで実行しているプロセスを認識して、それらのプロセスのいずれかがハブ・ノードの進行をブロックしているかどうかをチェックして、可能な場合は処置を実施します。
- ホルダーのOracle Database Quality of Service Management設定を考慮します。
- ホルダー・プロセスを終了して、そのリソースを待機している次のプロセスの進行を可能にしてハングを防止します。
- アラート・ログのORA-32701エラー・メッセージでDBAに通知します。

Oracle RACを含むOracle Multitenantの概要

Oracle RACを使用するようにマルチテナント・コンテナ・データベース(CDB)を構成できます。

各PDBをOracle RAC CDBの各データベース・インスタンスまたはインスタンスのサブセットで使用可能にすることができます。いずれの場合も、PDBへのアクセスは、動的データベース・サービスを使用して規制されます。アプリケーションは、シングル・インスタンスの非CDBに接続する場合と同様に、これらのサービスを使用してPDBに接続します。

同じOracle RACデータベースまたはインスタンスを共有している別のPDBを妨害する可能性のある、特定のPDBでの特定の操作が実行されないようにするために、PDBを分離できます。PDBの分離により、より大規模な統合が可能になります。

CDBとしてOracle RACデータベースを作成し、PDBをCDBに接続する場合、デフォルトでは、いずれのインスタンス上でも、PDBは自動的には起動されません。PDBに割り当てられた最初の動的データベース・サービス(データベース名と同じ名前のデフォルト・データベース・サービス以外)によって、サービスが実行されているインスタンス上でそのPDBが使用可能になります。

Oracle RACのCDBの複数のインスタンスでPDBが有効かどうかにかかわらず、CDBは通常、PDBで実行されるサービスで管理されます。PDBをそのインスタンスで手動で起動することにより、各インスタンスで手動でPDBのアクセスを有効にできます。

Database In-MemoryおよびOracle RACの概要

すべてのOracle RACノードには、独自のインメモリー(IM)列ストアがあります。デフォルトでは、移入オブジェクトはクラスタ内のすべてのIM列ストアにわたって分散されます。

すべてのOracle RACノードでIM列ストアを同じサイズにすることをお薦めしますOracle RACノードがIM列ストアを必要としない場合、INMEMORY_SIZEパラメータを0に設定します。

Oracle Database 19c, リリース更新19.8以降、Database In-Memoryには、Database In-Memoryオプションを必要とせずに最大16 GBの列ストアでDatabase In-Memoryを使用できる新しいベース・レベル機能があります。Oracle RACデータベースでは、各データベース・インスタンスのINMEMORY_SIZE設定が16 GBを超えないようにする必要があります。この機能を有効にするには、INMEMORY_FORCEパラメータをBASE_LEVELに設定します。

完全に異なるオブジェクトを各ノードに移入させたり、より大きなオブジェクトをクラスタ内のすべてのIM列格納間で分散させることが可能です。Oracle Engineered Systemsでは、同じオブジェクトを各ノードのIM列ストアに表示させることも可能です。クラスタ内のIM列ストア間のオブジェクトの分散は、INMEMORY属性への追加の副句(DISTRIBUTEおよびDUPLICATE)により制御されます。

Oracle RAC環境では、指定されたINMEMORY属性のみを含むオブジェクトが、クラスタ内のIM列ストア間で自動的に分散されます。DISTRIBUTE句を使用して、クラスタ間でのオブジェクトの分散方法を指定できます。デフォルトで、使用されるパーティション化のタイプ(ある場合)によりオブジェクトの分散方法が決定されます。オブジェクトがパーティション化されないと、ROWID範囲に分散されます。あるいは、DISTRIBUTE句を指定して、デフォルトの動作をオーバーライドできます。

Oracle Engineered Systemでは、クラスタ内のIM列ストア間で移入されたオブジェクトを複製またはミラー化できます。この技法により最高レベルの冗長性が提供されます。DUPLICATE句は、オブジェクトの複製方法を制御します。DUPLICATEのみを指定すると、データのミラー化されたコピーが1つ、クラスタ内のIM列ストア間で分散されます。各IM列ストア内のすべてのオブ

ジェクトを複製するには、DUPLICATE ALLを指定します。

ノート:

非エンジニアド・システムで Oracle RAC にデプロイする場合、DUPLICATE 句は NO DUPLICATE として扱われます。

関連項目

- 『Oracle Database In-Memoryガイド』

Oracle RAC環境の管理の概要

この項では、Oracle RAC環境の管理について説明します。内容は次のとおりです。

- Oracle RAC環境の設計およびデプロイ
- Oracle RAC環境の管理ツール
- Oracle RAC環境の監視
- Oracle RAC環境でのパフォーマンス評価

Oracle RAC環境の設計およびデプロイ

Oracle RACによる高可用性計画を設計および実装しているすべての企業は、高可用性を必要とするビジネス要因を綿密に分析することから始める必要があります。

高可用性を実現するためのビジネス要件を分析し、様々な高可用性ソリューションの実装に必要な投資レベルについて理解しておくことによって、ビジネスと技術の両方の目的を達成する高可用性アーキテクチャの開発が可能になります。

関連項目

- 設計およびデプロイメント方法

関連項目:

可用性の要件に最適なアーキテクチャを選択および実装するには、次の情報が役立ちます。

- 「設計およびデプロイ方法」では、ビジネスの高可用性要件を評価する場合に使用できる高水準の概要を示しています。
- 『*Oracle Database*高可用性概要』では、組織に最適なアーキテクチャを選択する方法および複数の高可用性アーキテクチャについて説明し、要件を満たす最適なアーキテクチャを選択するためのガイドラインを示し、Oracle Maximum Availability Architectureの情報も示します

Oracle RAC環境の管理ツール

管理者は、サーバー制御ユーティリティ(SRVCTL)、Oracle Enterprise Manager、SQL*Plus、およびその他のユーティリ

ティを使用して、クラスタ・データベースを単一システム・イメージとして管理します。

- サーバー制御ユーティリティ(SRVCTL): シングル・ポイントからOracle RACデータベースを管理するためのコマンドライン・インタフェース。SRVCTLを使用して、データベースおよびインスタンスの起動と停止、インスタンスおよびサービスの削除または移動を実行できます。SRVCTLを使用して、構成情報、Oracle Real Application Clusters One Node(Oracle RAC One Node)Oracle ClusterwareおよびOracle ASMの管理もできます。
- Oracle Fleet Patching and Provisioning (Oracle FPP): Oracle Fleet Patching and Provisioningは、Oracle RACデータベースのパッチ適用、アップグレード、およびプロビジョニングに使用します。
- Oracle Enterprise Manager: 非クラスタ・データベースおよびOracle RACデータベース環境を管理するOracle Enterprise Manager Cloud Control GUIインタフェース。可能な場合は、Oracle Enterprise Managerを使用して管理タスクを実行することをお薦めします。

  Oracle Enterprise Manager Cloud Controlを使用して、Oracle RAC One Nodeデータベースを管理することもできます。
- SQL*Plus: SQL*Plusコマンドは、現行のインスタンスで動作します。現行のインスタンスは、SQL*Plusセッションを開始したローカルのデフォルト・インスタンスまたはOracle Net Servicesの接続先リモート・インスタンスです。
- クラスタ検証ユーティリティ(CVU): クラスタとOracle RACの様々なコンポーネント(共有ストレージ・デバイスなど)、ネットワーク構成、システム要件、Oracle Clusterware、およびオペレーティング・システムのグループやユーザーの検証に使用するコマンドライン・ツール。インストール前およびインストール後のクラスタ環境のチェックにもCVUを使用できます。CVUは、Oracle ClusterwareおよびOracle RACコンポーネントのインストール前およびインストール時に特に役立ちます。Oracle ClusterwareおよびOracle Databaseのインストール後に、CVUを実行して環境を検証します。

  Oracle RACをインストールする前にCVUをインストールして、構成がOracle RACのインストールの最小要件を満たしていることを確認します。また、CVUを使用して、ノードの追加や削除などの管理タスクの完了を検証できます。
- Oracle DBCA: Oracle RAC、Oracle RAC One NodeおよびOracleの非クラスタ・データベースを作成し、最初に構成する場合の推奨ユーティリティ。
- NETCA: Oracle RAC環境のネットワークを構成します。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- データベース・インスタンスおよびクラスタ・データベースの管理
- Oracle RACおよびOracle Clusterwareの監視
- サーバー制御ユーティリティのリファレンス
- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle Database Net Services管理者ガイド

関連項目:

- SRVCTL、Oracle Enterprise ManagerおよびSQL*Plusを使用したOracle RAC管理の概要は、「データベース・インスタンスおよびクラスタ・データベースの管理」を参照してください
- 「Oracle RACおよびOracle Clusterwareの監視」

- SRVCTLの参照情報については、「サーバー制御ユーティリティのリファレンス」を参照してください
- クラスタ検証ユーティリティ(CVU)、およびOIFCFGツール(ネットワーク・インタフェースの割当てと割当て解除)やOCRCONFIGコマンドライン・ツール(OCRの管理)などのその他のOracle Clusterwareツールの詳細は、『*Oracle Clusterware*管理およびデプロイメント・ガイド』を参照してください。
- NETCAの詳細は、『*Oracle Database Net Services*管理者ガイド』を参照してください。

Oracle RAC環境の監視

WebベースのOracle Enterprise Manager Cloud Controlを使用すると、Oracle RACデータベースを監視できます。

Oracle Enterprise Manager Cloud Controlは、グラフィカル・ユーザー・インタフェース(GUI)を介してアクセスするOracle環境を集中的に制御します。Oracle Enterprise Managerを使用してOracle RAC環境を監視する方法の詳細は、Oracle RACおよびOracle Clusterwareの監視を参照してください。

Oracle RAC環境を監視する場合の推奨事項は次のとおりです。

- Oracle Enterprise Manager Cloud Controlを使用して、クラスタ・データベース管理タスクを開始します。
- Oracle Enterprise Manager Cloud Controlを使用して、複数または個々のOracle RACデータベースを管理します。
- V$ビューに基づいたグローバル・ビュー(GV$ビュー)を使用します。GV$ビューは、catclustdb.sqlスクリプトによって作成されます。データベースの作成にOracle DBCAを使用しない場合は、このスクリプトを実行します。それ以外の場合は、Oracle DBCAによってこのスクリプトが実行されます。

  ほとんどのV$ビューに対して、対応するGV$グローバル・ビューが存在します。V$情報に加えて、各GV$ビューにはINST_IDという名前の追加の列があり、ここにインスタンス番号が表示され、この番号に基づいて関連するV$ビュー情報が取得されます。
- Automatic Database Diagnostic Monitor (ADDM)および自動ワークロード・リポジトリ(AWR)を含む、Oracle Enterprise ManagerのOracle Database Diagnostic and Tuningパックの高度な管理および監視機能を使用します。

ノート:

Statspack は下位互換性に使用できますが、Statspack で実行できるのはレポートのみです。ブロック競合およびセグメント・ブロックの待機に関連する統計を収集するには、Statspack をレベル 7 で実行する必要があります。

関連項目

- Oracle RACおよびOracle Clusterwareの監視
- Oracle Databaseパフォーマンス・チューニング・ガイド

Oracle RAC環境でのパフォーマンス評価

Oracle RACに対して特別なチューニングは必要ありません。Oracle RACは特別な構成変更がなくてもスケーラビリティが向上します。

アプリケーションが非クラスタOracle RACデータベースで正常に動作する場合は、Oracle RAC環境でも正常に動作します。非クラスタOracle Databaseで実行する多くのチューニング・タスクによって、Oracle RACデータベースのパフォーマンスも向上できます。これは、より多くのCPUをまたいだスケーラビリティが必要な環境にとって特に当てはまることです。

Oracle RAC固有のパフォーマンス機能には、次のものがあります。

- 動的リソース割当て
    - 必要に応じて、キャッシュ・フュージョン・リソースが動的に割り当てられます。
    - リソースを動的に取得すると、リソースをデータ・ブロックに対してローカルなままに保持できるため、パフォーマンスが向上します。
- キャッシュ・フュージョンによる簡素化されたチューニング方法
    - キャッシュ・フュージョン用にパラメータをチューニングする必要はありません。
    - アプリケーション・レベルのチューニングは必要ありません。
    - 既存のアプリケーションに対してほとんど影響を及ぼすことなく、ボトムアップ・チューニングを実行できます。
- 詳細なパフォーマンス統計
    - Oracle RACパフォーマンスを監視するために様々なビューを使用できます。
    - Oracle Enterprise ManagerのOracle RAC固有のパフォーマンス・ビュー

2 Oracle RACの記憶域の管理

従来のボリューム・マネージャ、ファイル・システムおよびRAWデバイスに対して代替方法を提供するストレージ管理ソリューションとしてOracle Automatic Storage Management(Oracle ASM)をお薦めします。

Oracle ASMは、Oracleデータベース・ファイルのボリューム・マネージャ兼ファイル・システムで、単一インスタンスOracle DatabaseおよびOracle Real Application Clusters(Oracle RAC)構成をサポートします。

Oracle ASMでは、データファイルの格納にディスク・グループが使用されます。Oracle ASMディスク・グループとは、Oracle ASMで1つの単位として管理されるディスクの集合のことです。ディスク・グループ内では、各Oracleデータベース・ファイルに使用するファイル・システム・インタフェースがOracle ASMによって公開されます。ディスク・グループ内に保存されたファイルの内容は、均等に分散されるため、ホット・スポットがなくなり、ディスク間のパフォーマンスが均一になります。このパフォーマンスは、RAWデバイスのパフォーマンスに匹敵します。

ディスク・グループに対してディスクの追加や削除を行う際、そのディスク・グループのファイルにアクセス中のデータベースがあっても、そのアクセスが妨げられることはありません。Oracle ASMは、ディスク・グループに対するディスクの追加や削除が行われる際、ファイルの内容を自動的に再分散します。再分散のための停止時間は発生しません。

Oracle ASMのボリューム・マネージャ機能には、サーバーベースのフレキシブルなミラー化オプションが用意されています。標準冗長性と高冗長性のOracle ASMディスク・グループは、それぞれ双方向ミラー化と3方向ミラー化を可能にします。外部冗長性を使用すると、Redundant Array of Independent Disks(RAID)ストレージ・サブシステムでミラー化保護機能を実行できるようになります。

また、Oracle ASMではOracle Managed Files機能を使用してデータベース・ファイル管理を簡略化しています。Oracle Managed Filesは、指定した場所にファイルを自動的に作成します。さらに、Oracle Managed Filesでは、ファイルの命名およびその削除が行われ、表領域またはファイルを削除するときに領域が解放されます。

Oracle ASMは、データ記憶域を少数のディスク・グループに統合することで、データベース記憶域の管理にまつわるオーバーヘッドを低減します。より少ないディスク・グループで複数データベースのストレージを統合し、I/Oパフォーマンスを高めます。

Oracle ASMファイルは、RAWディスクやサード・パーティのファイル・システムなど他のストレージ管理オプションと共存できます。この機能により、Oracle ASMを既存の環境に統合する作業が簡素化されます。

Oracle ASMは、SQL*Plus、Oracle ASMコマンドライン・ユーティリティ(ASMCMD)・コマンドライン・インタフェース、Oracle ASM Configuration Assistant (ASMCA)などの使いやすい管理インタフェースを備えています。

この章の内容は次のとおりです。

- Oracle RAC用の記憶域管理の概要
- Oracle RACでのデータ・ファイルへのアクセス
- ストレージ用のNFSサーバー
- Oracle RACでのREDOログ・ファイル記憶域
- Oracle RACでの自動UNDO管理

- Oracle RACによるOracle Automatic Storage Management

関連項目

- Oracle Automatic Storage Management管理者ガイド
- Oracle Automatic Storage Management管理者ガイド

Oracle RAC用記憶域管理の概要

Oracle RACデータベース用のすべてのデータ・ファイル(各インスタンスのUNDO表領域を含む)およびREDOログ・ファイル(各インスタンスに少なくとも2つ)は、共有記憶域に存在する必要があります。

これらのファイルをOracle ASMディスク・グループに格納するには、Oracle ASMを使用することをお薦めします。

認定された**クラスタ・ファイル・システム**など、共有記憶域の代替使用方法がサポートされています。さらに、インスタンス固有のエントリを使用した単一の共有サーバー・パラメータ・ファイル(SPFILE)を使用することをお薦めします。Oracle RAC 12cでは、Oracle ASMでの共有パスワード・ファイルの格納、およびOracle Automatic Storage Management Cluster File System (Oracle ACFS)でのOracle Databaseファイルの格納ができます。

ノート:

Oracle Database、および Oracle Clusterware などの関連テクノロジは、RAW (ブロック)記憶域デバイスをサポートしていません。Oracle Clusterware 12c にアップグレードする前に、ファイルを Oracle ASM に移動する必要があります。

特に明記されていないかぎり、Oracle RAC環境内での、Oracle ASM、Oracle Managed Files、自動セグメント領域管理などのOracle Database記憶域の機能は、非クラスタのOracle Database環境と同じです。

関連項目

- Oracle Databaseソフトウェアのインストールとデータベースの作成の概要
- Oracle Automatic Storage Managementの概要
- Oracle Databaseの構造と記憶域

Oracle RACでのデータ・ファイルへのアクセス

すべてのOracle RACインスタンスは、すべてのデータ・ファイルにアクセスできる必要があります。データベースのオープン中にデータ・ファイルのリカバリが必要になった場合は、最初に起動するOracle RACインスタンスがリカバリを実行し、ファイルへのアクセスを検証します。他のインスタンスも、起動時に、データ・ファイルへのアクセスを検証します。同様に、表領域またはデータ・ファイルを追加したり、表領域またはデータ・ファイルをオンライン状態にする場合も、すべてのインスタンスがファイルへのアクセスを検証します。

他のインスタンスがアクセスできないデータ・ファイルをディスクに追加すると、検証に失敗します。インスタンスが同一データ・ファイルの異なるコピーへアクセスした場合も、検証に失敗します。いずれのインスタンスについても検証に失敗した場合は、問題を診断し解決してください。その後、各インスタンスに対してALTER SYSTEM CHECK DATAFILES文を実行して、データ・ファイルへのアクセスを検証します。

ストレージ用のNFSサーバー

Oracleデータベースは、ネットワーク・ファイル・システム(NFS)サーバーとして機能できます。データベースは、NFSクライアントからのNFSリクエストに応答し、データベース内のファイルとそのメタデータの両方を格納します。

プライマリ・データベースに関連付けられたファイル(SQLスクリプトなど)は、スタンバイ・データベースに自動的にレプリケートできます。また、構造化されていないデータ(電子メールなど)もデータベースに格納できます。

NFSサーバーを使用して、Oracleファイル・システムを作成または破棄できます。また、そのファイル・システムにアクセスすることもできます。この手順の詳細は、『*Oracle Database SecureFiles*およびラージ・オブジェクト開発者ガイド』を参照してください。

関連項目

- Oracle Database SecureFiles and Large Objects開発者ガイド
- Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス
- Oracle Databaseリファレンス

Oracle RACでのREDOログ・ファイル記憶域

Oracle RACデータベースでは各インスタンスは、REDOログ・ファイルの少なくとも2つのグループを備える必要があります。REDOログ・グループを割り当ててから、ALTER DATABASE ENABLE INSTANCE instance_nameコマンドを使用して新規インスタンスを有効化する必要があります。DBCAを使用してデータベースを作成する場合、DBCAは、必要に応じてREDOログ・ファイルをインスタンスに自動的に割り当てます。初期データベースの作成時または作成後のステップとして必要に応じて、REDOログ・グループの数およびREDOログ・ファイルのサイズを変更できます。

現在のグループが一杯になると、インスタンスは次のログ・ファイル・グループへの書込みを開始します。データベースがARCHIVELOGモードの場合は、各インスタンスは一杯になったオンライン・ログ・グループをアーカイブREDOログ・ファイルとして保存し、このファイルは制御ファイルに記録されます。データベースのリカバリ時に、有効化されているすべてのインスタンスについて、リカバリが必要かどうかがチェックされます。Oracle RACデータベースからインスタンスを削除する場合は、データベースのリカバリ時にスレッドをチェックする必要がないように、インスタンスのREDOスレッドを無効化する必要があります。

REDOログの管理は、特定の本番Oracle RACデータベースのインスタンスの数を変更するときに考慮する必要があります。たとえば、ポリシー管理データベースのサーバー・プールのカーディナリティを増やし、新しいサーバーをサーバー・プールに割り当てると、新しいサーバーのインスタンスが起動されます。新しいサーバーのデータベース・インスタンスが起動すると、一連のREDOログ・グループが必要になります。Oracle ASMディスク・グループに基づいたOracle Managed Filesを使用すると、必要なREDOログ・スレッドおよびそれぞれのファイルの割り当てが自動的に実行されます。REDOログ・グループは、管理者管理データベースを使用する場合にのみ作成する必要があります。

管理者管理データベースの場合、インスタンスごとに、独自のオンラインREDOログ・グループがあります。これらのREDOログ・グループを作成し、グループ・メンバーを設定します。REDOログ・グループを特定のインスタンスに追加するには、ALTER DATABASE ADD LOGFILE文でINSTANCE句を指定します。REDOログ・グループの追加時にインスタンスを指定しない場合は、現在接続しているインスタンスにREDOログ・グループが追加されます。

各インスタンスには、2つ以上のREDOログ・ファイルのグループが必要です。REDOログ・グループを割り当ててから、ALTER

DATABASE ENABLE INSTANCE instance_nameコマンドを使用して新規インスタンスを有効化する必要があります。現在のグループが一杯になると、インスタンスは次のログ・ファイル・グループへの書込みを開始します。データベースがARCHIVELOGモードの場合は、各インスタンスは一杯になったオンライン・ログ・グループをアーカイブREDOログ・ファイルとして保存し、このファイルは制御ファイルに記録されます。

データベースのリカバリ時に、有効化されているすべてのインスタンスについて、リカバリが必要かどうかがチェックされます。Oracle RACデータベースからインスタンスを削除する場合は、データベースのリカバリ時にスレッドをチェックする必要がないように、インスタンスのREDOスレッドを無効化する必要があります。

関連項目

- Oracle RAC環境の設計およびデプロイについて
- Oracle Database管理者ガイド
- Oracle Database SQL言語リファレンス

Oracle RACでの自動UNDO管理

インスタンスに割り当てられた特定のUNDO表領域内のUNDOセグメントは、Oracle Databaseによって自動的に管理されます。インスタンスは、読取り一貫性のためにいつでも、クラスタ環境内のすべてのUNDOブロックを読み取ることができます。また、UNDO表領域が別のインスタンスにUNDO生成またはトランザクション・リカバリのために使用されていなければ、どのインスタンスもトランザクション・リカバリ中にそのUNDO表領域を更新できます。

Oracle RAC管理者管理データベース内にUNDO表領域を割り当てるには、SPFILEまたは個別のPFILEで各インスタンスのUNDO_TABLESPACEパラメータに別の値を指定します。ポリシー管理データベースの場合、Oracle Managed Filesが有効化されていれば、インスタンスの起動時にUNDO表領域が自動的に割り当てられます。Oracle RACデータベースでは、自動UNDO管理モードと手動UNDO管理モードを同時に使用することはできません。Oracle RACデータベースのすべてのインスタンスは、同じUNDOモードで操作してください。

関連項目

- Oracle RACのSPFILEパラメータ値の設定
- Oracle Database管理者ガイド

Oracle RACによるOracle Automatic Storage Management

Oracle ASMは、管理対象のディスク間で記憶域構成を管理することにより、自動的に最大のI/Oパフォーマンスを引き出します。

Oracle ASMはこれを行うために、Oracle ASM内のディスク・グループに割り当てられている使用可能なすべての記憶域にわたってデータベース・ファイルを均等に分散します。Oracle ASMによって、ディスク領域全体の要件は、ディスク・グループ内のすべてのディスクに均等なサイズで割り当てられます。Oracle ASMでは、データの損失を防止するために、ファイルのミラー化も自動的に行われます。Oracle ASMのこれらの機能により、管理オーバーヘッドも大幅に削減されます。

Oracle ASMインスタンスは、Oracle Clusterwareをインストールする各ノードに作成されます。各Oracle ASMインスタンスには、SPFILEまたはPFILEタイプのパラメータ・ファイルが存在します。パラメータ・ファイルおよびデフォルト以外のOracle Netリ

スナーのTNSエントリをバックアップすることをお薦めします。

Oracle RACでOracle ASMを使用するには、Database Configuration Assistant(DBCA)を使用してデータベースを作成する際に、Oracle ASMを記憶域オプションとして選択します。非クラスタのOracle Databaseの場合と同様、Oracle RACでOracle ASMを使用する場合もI/Oチューニングは不要です。

次の項では、Oracle ASMおよびOracle ASMの管理について説明します。

- Oracle RACでの記憶域管理
- Oracle ASM用ディスク・グループ構成の変更
- Oracle ASMディスク・グループの管理
- 拡張遠距離クラスタでの優先読取りミラー・ディスクの構成
- クラスタ化されていないOracle ASMからクラスタ化されたOracle ASMへの変換
- Oracle RACでのSRVCTLを使用したOracle ASMインスタンスの管理

関連項目

- Oracle Automatic Storage Management管理者ガイド

Oracle RACでの記憶域管理

Oracle ASMディスク・グループの作成およびOracle ASMディスク・グループのミラー化の構成は、Oracle ASMコンフィギュレーション・アシスタント(ASMCA)で実行できます。

または、Oracle Enterprise Managerを使用して、Oracle Enterprise Managerで対応するサーバーを見つけた後に、Oracle ASMディスク・グループを管理できます。

Oracle ASMの管理に使用するASMCA、Oracle Enterprise ManagerなどのOracleツールと、サイレント・モードのインストールおよびアップグレード・コマンドでは、Oracle ASMインスタンスおよびディスク・グループを管理するオプションを使用できます。

クラスタ全体のOracle ASMの整合性は、クラスタ検証ユーティリティ(CVU)で検証できます。通常、このチェックによりすべてのノードのOracle ASMインスタンスが同じOracleホームから実行されていることが保証され、asmlibが存在する場合は、このライブラリのバージョンが有効であることと所有権が有効であることが保証されます。次のコマンドを実行して、このチェックを実行します。

```
cluvfy comp asm [-n node_list] [-verbose]
```

node_listの部分は、チェックを実行するノード名をカンマで区切ったリストに置き換えてください。allを指定すると、クラスタ内のすべてのノードがチェックされます。

cluvfy comp ssaコマンドを使用して、共有記憶域を検出します。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

Oracle ASM用ディスク・グループ構成の変更

クラスタにディスク・グループを作成する場合、またはクラスタ化されている既存のディスク・グループに新しいディスクを追加する場合は、共有ディスク上に基盤となる物理記憶域を準備して、Oracleユーザーにディスクに対する読取り/書込み権限を付与します。

Oracle ASMをOracle RACデータベースで使用する場合と非クラスタのOracle Databaseで使用する場合の実質的な相違点は、共有ディスクが必要かどうかのみです。Oracle ASMでは、ディスクまたはディスク・グループの追加または削除後に、データ・ファイルの再分散が自動的に行われます。

クラスタでは、各Oracle ASMインスタンスが、各ノードのディスク・グループの更新メタデータを管理します。また、各Oracle ASMインスタンスが、ディスク・グループのメタデータとクラスタの他のノード間の調整を行います。非クラスタのOracle Databaseの場合と同様、Oracle RACで使用されるOracle ASM用ディスク・グループの管理に、Oracle Enterprise Manager、ASMCA、SQL*PlusおよびサーバーI制御ユーティリティ(SRVCTL)を使用できます。SQL*Plusを使用してOracle ASMインスタンスを管理する方法の詳細は、『*Oracle Automatic Storage Management*管理者ガイド』を参照してください。その他のツールの使用方法については、次の各項で説明します。

ノート:

ASMCA を起動したときに Oracle ASM インスタンスが存在しない場合は、Oracle ASM インスタンスの作成を求めるプロンプトが表示されます。

関連項目

- Oracle Automatic Storage Management管理者ガイド

Oracle ASMディスク・グループの管理

Oracle ASMを使用するには、Oracle DBCAでデータベースを作成する前に、ASMCAでディスク・グループを作成しておく必要があります。

ディスク・グループ管理コマンドを使用すると、データベースの作成とは関係なく、Oracle ASMインスタンスおよびその関連のディスク・グループの作成および管理もできます。Oracle Enterprise ManagerまたはASMCAを使用すると、ディスク・グループへのディスクの追加、1つまたはすべてのディスク・グループのマウント、Oracle ASMインスタンスの作成を実行できます。また、Oracle Enterprise Managerを使用して、ディスク・グループをディスマウントまたは削除したり、Oracle ASMインスタンスを削除することもできます。

Oracle ASMインスタンスは、Oracle Clusterwareをインストールするときに作成されます。Oracle ASMディスク・グループを作成するには、Grid_home/binディレクトリからASMCAを実行します。Oracle ASMの管理には、ASMCAの「Oracle ASMディスク・グループ」ページも使用できます。つまり、データベースの作成とは別にOracle ASM記憶域を構成できます。たとえば、「ASMディスク・グループ」ページから、ディスク・グループの作成、既存のディスク・グループへのディスクの追加、または現在マウントされていないディスク・グループのマウントを実行できます。

ASMCAを起動したときにASMインスタンスが存在しない場合は、インスタンスの作成を求めるプロンプトが表示されます。

sysasmのパスワードとASMSNMPのパスワードを求めるプロンプトが表示されます。

関連項目

- Oracle Automatic Storage Management管理者ガイド

拡張遠距離クラスタでの優先読取りミラー・ディスクの構成

優先読取りディスクを構成してパフォーマンスを向上させることができます。

Oracle Automatic Storage Management (Oracle ASM)障害グループを構成している場合、ノードは、そのノードに最も近いエクステントから(そのエクステントがセカンダリ・エクステントであっても)読み取る方が効率的です。ノードから遠い場所にあるプライマリ・コピーからではなく、ノードにより近い場所にあるセカンダリ・エクステントから読み取るようにOracle ASMを構成できます。拡張遠距離クラスタでは、優先読取り障害グループを使用するのが最も有効です。

この機能を構成するには、ASM_PREFERRED_READ_FAILURE_GROUPS初期化パラメータで、障害グループ名のリストを優先読取りディスクとして指定します。エクステントの1つ以上のミラー・コピーを、拡張クラスタ内のノードに対してローカルなディスクから構成することをお薦めします。ただし、1つのインスタンスに優先される障害グループが、同じOracle Real Application Clusters (Oracle RAC)データベース内の別のインスタンスに対してはリモートである可能性があります。優先読取り障害グループのパラメータ設定は、インスタンス固有です。

関連項目

- 優先読取りの障害グループ
- ASM_PREFERRED_READ_FAILURE_GROUPS

クラスタ化されていないOracle ASMからクラスタ化されたOracle ASMへの変換

Oracle Grid Infrastructureをインストールすると、クラスタ化されていないOracle Automatic Storage Management (Oracle ASM)インスタンスは、クラスタ化されたOracle ASMに自動的に変換されます。

関連項目

- 優先読取りの障害グループ

Oracle RACでのSRVCTLを使用したOracle ASMインスタンスの管理

サーバー制御ユーティリティ(SRVCTL)を使用すると、Oracle ASMインスタンスを追加または削除できます。

SRVCTLコマンドを発行してOracle ASMを管理するには、Oracle Grid Infrastructureホームを所有するオペレーティング・システム・ユーザーとしてログインして、Oracle Grid InfrastructureホームのbinディレクトリからSRVCTLコマンドを発行します。

Oracle ASMインスタンスを追加するには、次の構文を使用します。

```
srvctl add asm
```

Oracle ASMインスタンスを削除するには、次の構文を使用します。

```
srvctl remove asm [-force]
```

Oracle ASMインスタンスの起動、停止およびステータスの取得にも、SRVCTLを使用できます。次に例を示します。

Oracle ASMインスタンスを起動にするには、次の構文を使用します。

```
srvctl start asm [-node node_name] [-startoption start_options]
```

Oracle ASMインスタンスを停止するには、次の構文を使用します。

```
srvctl stop asm [-node node_name] [-stopoption stop_options]
```

Oracle ASMインスタンスの構成を表示するには、次の構文を使用します。

```
srvctl config asm -node node_name
```

Oracle ASMインスタンスの状態を表示するには、次の構文を使用します。

```
srvctl status asm [-node node_name]
```

関連項目

- サーバー制御ユーティリティのリファレンス
- Oracle Automatic Storage Management管理者ガイド

3 データベース・インスタンスおよびクラスタ・データベースの管理

この章では、Oracle Real Application Clusters(Oracle RAC)データベースおよびデータベース・インスタンスの管理方法について説明します。

内容は次のとおりです。

- Oracle RACデータベースの管理の概要
- インスタンスおよびOracle RACデータベースの起動および停止
- Oracle RACでのPDBの起動および停止
- インスタンスの実行の確認
- 特定のクラスタ・インスタンス上でのセッションの終了
- Oracle RACでの初期化パラメータ・ファイルの概要
- Oracle RACでの初期化パラメータの使用
- 管理者管理データベースのポリシー管理データベースへの変換
- データベース・サーバーのメモリー不足の管理
- Oracle RACデータベースの静止
- LinuxおよびUNIXプラットフォームでの複数のクラスタ・インターコネクトの管理
- Oracle ClusterwareでのOracle RACデータベースの管理方法のカスタマイズ
- Oracle Enterprise Managerの高度な管理

関連項目:

Oracle Enterprise Manager Cloud Controlの詳細は、Oracle Enterprise Manager Cloud Controlのオンライン・ヘルプを参照してください。

Oracle RACデータベースの管理の概要

Oracle RACデータベースの管理には、特定の権限が必要になり、ポリシー管理または管理者管理のどちらかのデプロイメント・モデルが必要になります。

Oracle RACデータベースの管理に必要な権限

セキュリティの強化と管理業務の分離を進めるために、Oracle RACデータベースの管理者はOracle RACデータベースをSYSRAC管理権限で管理します。SYSDBA管理権限は必要ではなくなりました。SYSRAC管理権限は、SRVCTLなどのOracle RACユーティリティのかわりに、Oracle Clusterwareエージェントでデータベースに接続する際のデフォルト・モードです。つまり、Oracle RACクラスタの日常管理作業には、データベースへのSYSDBA接続が不要になったということです。

Oracle RACデータベースのデプロイメント・モデル

Oracle RACデータベースは、異なる2つの管理スタイルおよびデプロイメント・モデルをサポートしています。

- 管理者管理デプロイメントは、Oracle Database 11gリリース2 (11.2)の前に存在していたOracle RACデプロイメント・タイプに基づき、クラスタ内の特定のノードで実行されるように各データベース・インスタンスを静的に構成する必要があり、また、preferredおよびavailable宛先を使用して、特定のデータベースに属する特定のインスタンスで実行されるようにデータベース・サービスを構成する必要があります。
- ポリシー管理デプロイメントは、サーバー・プールに基づき、この場合、データベース・サービスは、サーバー・プール内でシングルトンまたは均一として、サーバー・プール内のすべてのサーバーにわたって実行されます。データベースは1つ以上のサーバー・プールにデプロイされ、サーバー・プールのサイズによってデプロイメント内のデータベース・インスタンスの数が決まります。

同じコマンドまたは方法(DBCAやOracle Enterprise Managerなど)を使用して、管理者管理またはポリシー管理デプロイメント・モデルでデータベースを管理できます。すべてのコマンドおよびユーティリティは、管理者ベース管理(Oracle Database 11gリリース2 (11.2)より前のOracle Databases)のみをサポートしているOracle Databaseの管理をサポートするために下位互換性を維持しています。

一般にデータベースは、Oracle Clusterwareのリソースとして定義されます。データベース・リソースは、DBCAでデータベースを作成したとき、または高速ホーム・プロビジョニングを使用してデータベースをプロビジョニングしたときに自動的に作成されます。また、SRVCTLでデータベースを追加することで、データベース・リソースを手動で作成することもできます。データベース・リソースには、Oracleホーム、SPFILE、1つ以上のサーバー・プール、およびデータベースの起動に必要な1つ以上のOracle ASMディスク・グループが含まれます。Oracle ASMディスク・グループの指定には、srvctl add databaseコマンドまたはsrvctl modify databaseコマンドのどちらかを使用します。また、このリストに登録されていないディスク・グループのデータ・ファイルをデータベースが開くときに、ディスク・グループがリストに追加されます。

また、データベース・リソースにはリスナー・タイプに弱い起動依存性があり、これは、データベース・インスタンスの起動時にリソースがノードのすべてのリスナーを起動しようとすることを意味しています。Oracle Clusterwareは、データベース・インスタンスが起動するノードのリスナーを起動しようとします。リスナーを順に起動すると、ノードのVIPが起動します。

管理者管理データベースのデータベース・リソースを確認すると、そのOracle Databaseと同じ名前で定義されたサーバー・プールが表示されます。このサーバー・プールは、Oracleで定義される特別なサーバー・プールの一部で、Genericと呼ばれます。Oracle RACは、Genericサーバー・プールを管理して管理者管理データベースをサポートします。SRVCTLまたはDBCAのいずれかを使用して管理者管理データベースを追加または削除すると、Genericのメンバーであるサーバー・プールがOracle RACによって作成または削除されます。Genericサーバー・プールの変更に、SRVCTLまたはCRSCTLコマンドを使用することはできません。

ポリシー管理データベースを使用して、動的システムの管理を簡素化します。ポリシー管理により、クラスタおよびデータベースは、要件の変更に応じて拡張または縮小できます。ポリシー管理データベースを使用する場合は、クラスタ内のすべてのノードにOracleホーム・ソフトウェアをインストールする必要があります。ポリシー管理データベースは、Oracle Database 11gリリース2 (11.2)以上のソフトウェアを使用する必要があり、管理者管理データベースと同じサーバーに共存させることはできません。

ノート:

同じノードで同じデータベースの複数のインスタンスを実行することはできません。

ポリシー管理データベースは、カーディナリティ(通常の操作で実行する必要があるデータベース・インスタンス数)で定義されます。ポリシー管理データベースは、クラスタ管理者がクラスタに作成した1つ以上のデータベース・サーバー・プールで実行することも、別のサーバーで異なるタイミングで実行することもできます。ポリシー管理データベースの各サーバー・プールは、少なくとも1つのデータベース・サービスを持つ必要があります。データベース・インスタンスは、データベースに定義されたサーバー・プール内のサーバーで起動します。データベース記憶域に、Oracle Automatic Storage Management(Oracle ASM)およびOracle Managed Filesを使用していて、かつインスタンスの起動時に使用可能なREDOスレッドがない場合、Oracle RACは自動的にREDOスレッドを使用可能にし、必要なREDOログ・ファイルとUNDO表領域を作成します。クライアントは、その時点で実行されているサーバーに関係なく、同じSCANベース接続文字列を使用してポリシー管理データベースに接続することができます。

ポリシー管理データベース・インスタンスには、db_unique_name_cardinalityという名前が付けられます。cardinalityは、サーバー・プール内のサーバーのカーディナリティIDです。ローカル・ノード上のインスタンス名を取得するには、srvctl status database -sidコマンドを使用します。また、srvctl modify instanceコマンドを使用すると、ノードからインスタンス名への固定マッピングを作成することもできます。

管理者管理データベースとポリシー管理データベースへの同じクラスタの使用

すでにポリシー管理データベースがホストされているクラスタに管理者管理データベースを作成する場合、管理者管理データベース用のノードは慎重に選択する必要があります。これは、管理者管理データベース用に選択したノードはポリシー管理サーバー・プールにあり、このプロセスの一部としてGenericサーバー・プールに移動されるためです。

すでに他のポリシー管理データベース・インスタンスを実行しているノードを選択した場合、DBCAが管理者管理データベースを作成する際に停止されるインスタンスおよびサービスがリストされたメッセージがDBCAによって表示されます。DBCAによって「続行しますか。」と尋ねられ、そのダイアログ・ボックスで「はい」を選択すると、ポリシー管理データベースのインスタンスおよびサービスは、管理者管理データベース作成プロセスのために停止されます。

ノート:

srvctl add instance コマンドを使用した場合も同様で、データベースが停止されるという内容の同様のメッセージが返されます。srvctl add instance コマンドで強制オプション(-f)を使用した場合も、DBCA ダイアログで「はい」を選択するのと同じです。これにより、ノードを Generic サーバー・プールに移動する前に、そのノードで実行中の一部のポリシー管理データベースが停止されます。

関連項目

- 『Oracle Databaseセキュリティ・ガイド』
- Oracle Clusterware管理およびデプロイメント・ガイド
- 管理者管理データベースのポリシー管理データベースへの変換

Oracle RACの管理ツール

次の項では、Oracle RACデータベースおよびインスタンスを管理するために一般に使用する3つのツール(SRVCTLユーティリティ、Oracle Enterprise ManagerおよびSQL*Plus)を使用したOracle RAC管理について説明します。多くの場合、これらのツールを使用してOracle RAC環境を管理する方法は、非クラスタのOracle Databaseを管理する場合と同様です。

- SRVCTLを使用したOracle RACの管理
- Oracle Enterprise Managerを使用したOracle RACの管理
- SQL*Plusを使用したOracle RACの管理

SRVCTLを使用したOracle RACの管理

サーバー制御ユーティリティ(SRVCTL)は、集中管理的にOracle Databasesを管理するために使用できるコマンドライン・インタフェースです。

Oracleは、クラスタ用のOracle Grid Infrastructureに基づいて、非クラスタ環境とOracle RACデータベースの両方向けのOracle Grid InfrastructureのOracle ASMを使用して、単一インスタンスOracle Databases用のOracle Database 11gリリース2 (11.2)で中央集中型のSRVCTLベースのデータベース管理を使用できるようにしました。これにより、SRVCTLを使用した、すべてのOracle Databaseタイプの同機種管理が可能になります。SRVCTLを使用して、データベースおよびインスタンスの起動と停止、インスタンスおよびサービスの削除または移動を実行できます。また、SRVCTLを使用すると、クラスタ内の他のリソースに加え、サービスの追加および構成情報の管理を行うこともできます。

SRVCTLを使用してクラスタに構成操作を実行すると、SRVCTLは、クラスタ内のOracle Cluster Registry (OCR)に、またはOracle Restart環境内のOracle Local Registry (OLR)に構成データを格納します。SRVCTLは、Oracle Clusterwareリソース(Oracle Call Interface APIを使用してデータベースの起動および停止操作を実行するエージェントを定義)を構成および管理することによって、インスタンスの起動および停止のようなその他の操作を実行します。

ノート:

特定の環境変数を使用してデータベース(またはデータベース・インスタンス)を起動する必要がある場合は、srvctl setenv コマンドを使用して、SRVCTL の使用によってデータベース用に維持されているデータベース・プロファイルに対してこれらの変数を設定します。ORACLE_HOME および ORACLE_SID 環境変数を設定する必要はありません。SRVCTL が自動的にこれらのパラメータを維持および設定するためです。

関連項目

- サーバー制御ユーティリティのリファレンス

Oracle Enterprise Managerを使用したOracle RACの管理

Oracle Enterprise Managerでは、Oracle RAC環境を集中的に制御し、複数のクラスタ・データベースで同時に管理タスクを実行できます。

Oracle Enterprise Manager Cloud Control (Oracle Enterprise Manager 11gではGrid Control)グラフィカル・ユーザー・インタフェース(GUI)に基づいて、非クラスタ環境とOracle RAC環境の両方を管理できます。

Oracle Enterprise Managerでは、通常、Oracle RAC固有の管理タスクは、クラスタ・データベース全体に関係するタスクおよび特定のインスタンスに関係するタスクの2つのレベルが中心です。たとえば、Oracle Enterprise Managerでは、ジョブのスケジュールやメトリックのアラートしきい値の設定に加え、データベース、クラスタ・データベース・インスタンスおよびそのリスナーを起動、停止および監視できます。または、パラメータの設定やリソース・プランの作成のようなインスタンス固有のコマンドを実行できます。また、Oracle Enterprise Managerを使用して、スキーマ、セキュリティおよびクラスタ・データベースの記憶域機能を管理できます。

関連項目

- Oracle Enterprise Managerの高度な管理

SQL*Plusを使用したOracle RACの管理

SRVCTLまたはOracle Enterprise Managerとは異なり、SQL*Plusはインスタンス指向の管理ツールです。

SQL*Plusコマンドは、現行のインスタンスで動作します。現行のインスタンスは、SQL*Plusセッションを開始したローカルのデフォルト・インスタンスまたはOracle Net Servicesの接続先リモート・インスタンスです。1つのデータベースで複数のインスタンスを同時に実行するOracle RAC環境の場合、これは、このインスタンス上でSQL*Plusが動作できる範囲を考慮する必要があることを意味しています。これらの制限のため、ポリシー管理データベースを管理する場合はSQL*Plusを使用しないでください。

ノート:

Oracle Grid Infrastructure 21c以降、ポリシー管理データベースは非推奨です。

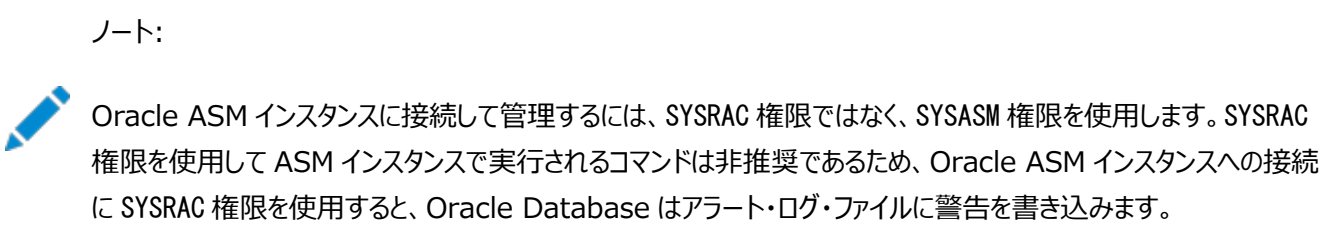
たとえば、プラガブル・データベース(PDB)が管理者管理スタイルとポリシー管理スタイルのいずれで管理されているかにかかわらず、これらのデータベースを使用する場合、SQL*Plus接続を使用してPDBに実行される変更は、デフォルトで現行のインスタンスにのみ影響することを考慮する必要があります。PDBに属するすべてのインスタンスに影響するような変更を行うには、ALTER PLUGGABLE DATABASEコマンドとinstance=allを使用する必要があります。PDBを使用する場合、動的データベース・サービス(net_service_name)を使用してインスタンスに接続する必要があります。PDBは、自らを、Oracle RACデータベースの1つ以上のインスタンスに関連付けられた動的データベース・サービスとして表しているためです。

デフォルトでは、SQL*Plusのプロンプトで現行のインスタンスが識別されないため、正しいインスタンスにコマンドを発行する必要があります。SQL*Plusセッションを開始して、インスタンスを指定せずにデータベースに接続すると、すべてのSQL*Plusコマンドはローカル・インスタンスで処理されます。この場合も、デフォルト・インスタンスが現行のインスタンスです。

デフォルトでは、SQL*Plusのプロンプトでは現行のインスタンスが識別されないため、正しいインスタンスにコマンドを発行する必要があります。SQL*Plusセッションを開始して、インスタンスを指定せずにデータベースに接続すると、すべてのSQL*Plusコマンドはローカル・インスタンスで処理されます。この場合も、デフォルト・インスタンスが現行のインスタンスです。SQL*Plusで別のインスタンスに接続するには、次の例のように、新しいCONNECTコマンドを発行し、リモート・インスタンスのネット・サービス名を指定します(passwordはパスワードです)。

```
CONNECT user_name@net_service_name
Enter password: password
```

SYSOPERまたはSYSRACとして接続すると、インスタンスの起動や停止などの権限が必要になる操作を実行できます。複数のSQL*Plusセッションが、同時に同じインスタンスに接続できます。他のインスタンスに接続すると、SQL*Plusによって最初のインスタンスとの接続が自動的に切断されます。



ノート:

Oracle ASM インスタンスに接続して管理するには、SYSRAC 権限ではなく、SYSASM 権限を使用します。SYSRAC 権限を使用して ASM インスタンスで実行されるコマンドは非推奨であるため、Oracle ASM インスタンスへの接続に SYSRAC 権限を使用すると、Oracle Database はアラート・ログ・ファイルに警告を書き込みます。

関連項目

- Oracle ASMインスタンスにアクセスするための認証
- ネーミング・メソッドの構成
- ALTER PLUGGABLE DATABASE文を使用したPDBの変更

インスタンスへのSQL*Plusコマンドの適用方法

Oracle RACデータベースでのインスタンスの起動および停止に、SQL*Plusを使用できます。

ほとんどのSQL文は、現行のインスタンスに適用されます。SQL*Plusコマンドを、LinuxおよびUNIXシステムではrootとして、またWindowsシステムではAdministratorとして実行する必要はありません。非クラスタのOracle Databaseで通常使用する権限を持つ適切なデータベース・アカウントのみが必要です。SQL*Plusコマンドのインスタンスへの適用方法の例を示します。

- ALTER SYSTEM CHECKPOINT LOCALは、デフォルトのインスタンスまたはすべてのインスタンスではなく、現在接続しているインスタンスにのみ適用されます。
- ALTER SYSTEM CHECKPOINTまたはALTER SYSTEM CHECKPOINT GLOBALは、クラスタ・データベースのすべてのインスタンスに適用されます。
- ALTER SYSTEM SWITCH LOGFILEは、現行のインスタンスにのみ適用されます。
    - グローバル・ログ・スイッチを強制するには、ALTER SYSTEM ARCHIVE LOG CURRENT文を使用します。
    - ALTER SYSTEM ARCHIVE LOGのINSTANCEオプションにより、特定のインスタンスについて各オンラインREDOログ・ファイルをアーカイブできます。

次の表に、SQL*Plusコマンドのインスタンスへの適用方法を示します。

表3-1 インスタンスへのSQL*Plusコマンドの適用方法

SQL*Plusコマンド	関連するインスタンス
ARCHIVE LOG	常に現行のインスタンスに対して適用されます。
CONNECT	CONNECT コマンドにインスタンスが指定されていない場合は、デフォルト・インスタンスに適用されます。
HOST	現行のインスタンスおよびデフォルト・インスタンスの場所に関係なく、SQL*Plus セッションを

SQL*Plusコマンド	関連するインスタンス
	実行しているノードに適用されます。
RECOVER	特定のインスタンスではなく、データベースに適用されます。
SHOW INSTANCE	現行のインスタンスに関する情報を表示します。リモート・インスタンスにコマンドをリダイレクトしている場合、現行のインスタンスはデフォルトのローカル・インスタンスではない場合があります。
SHOW PARAMETER および SHOW SGA	現行のインスタンスからパラメータおよび SGA 情報を表示します。
STARTUP および SHUTDOWN	常に現行のインスタンスに対して適用されます。権限付きの SQL*Plus コマンドです。

インスタンスおよびOracle RACデータベースの起動および停止

Oracle Enterprise Manager、SQL*PlusまたはSRVCTLを使用して、インスタンスを起動および停止できます。

Oracle Enterprise ManagerおよびSRVCTLでは、Oracle RACデータベースのすべてのインスタンスの起動および停止を1つのステップで行うオプションを提供しています。

任意のツールを使用して、データベースを起動する起動状態を選択できます。データベースおよびデータベース・インスタンスの状態によって、実行できる操作が決まります。データベースがMOUNT (NOMOUNT)状態にある場合のみ特定の操作を実行できます。他の操作を実行するには、データベースがOPEN状態である必要があります。

ノート:

同じノードで同じデータベースの複数のインスタンスを実行することはできません。

クラスタ内のノードでOracle RACデータベース・インスタンスを起動するには、まずノードでOracle Grid Infrastructureスタックを起動する必要があります。Oracle Grid Infrastructureスタックが実行されていないサーバーでは、Oracle RACデータベース・インスタンスは起動しません。

Oracle Database QoS Managementポリシー・ワークロードの重要性によるデータベースの起動順序の決定

ユーザー作成のOracle Database Quality of Service Management (Oracle Database QoS Management)

ポリシーがアクティブな場合、パフォーマンス・クラスのランク付けされた順序により、関連するOracle RACデータベースの開始順序またはリクエスト・リアルタイムLMSプロセス・スロットが決定されます。パフォーマンス・クラス・ランキングの使用により、統合環境で実行しているミッションクリティカルなデータベースに、リアルタイムで実行するLMSプロセスが確実に含まれることになるので、ノード間通信でのリソースのボトルネックが解消されます。Oracle Database QoS Managementポリシーでは各ワークロードのランクを指定するので、各データベースにMax (Ranks) の値を使用することにより、各データベースのビジネスの重要度を一貫性のある表現で示すことができます。

次の各項の手順では、Oracle RACデータベース・インスタンスの起動および停止について説明します。

- SRVCTLを使用した1つ以上のインスタンスおよびOracle RACデータベースの起動
- SRVCTLを使用した1つ以上のインスタンスおよびOracle RACデータベースの停止
- CRSCTLを使用したすべてのデータベースおよびインスタンスの停止
- SQL*Plusを使用した個々のインスタンスの起動と停止

関連項目

- Oracle Database概要

SRVCTLを使用した1つ以上のインスタンスおよびOracle RACデータベースの起動

SRVCTLを使用して、Oracle RACデータベースおよびインスタンスを起動します。

ノート:

この項では、データベースに SPFILE を使用していることを前提にしています。

次のSRVCTL構文をコマンドラインから入力し、必要なデータベース名およびインスタンス名を提供するか、または複数のインスタンス名を指定して、複数の特定のインスタンスを起動します。

- クラスタ・データベース全体(すべてのインスタンスおよび使用可能なサービス)を起動または停止するには、次のSRVCTLコマンドを入力します。

```
$ srvctl start database -db db_unique_name [-startoption start_options]

$ srvctl stop database -db db_unique_name [-o stop_options]
```

たとえば、次のSRVCTLコマンドは、Oracle RACデータベースの実行中でないすべてのインスタンスをマウントします。

```
$ srvctl start database -db orcl -startoption mount
```

- 管理者管理データベースを起動する場合は、インスタンス名のカンマ区切りリストを入力します。

```
$ srvctl start instance -db db_unique_name -instance instance_name_list
  [-startoption start_options]
```

Windowsでは、カンマ区切りリストを二重引用符(“”)で囲む必要があります。

- ポリシー管理データベースを起動する場合は、単一のノード名を入力します。

```
$ srvctl start instance -db db_unique_name -node node_name
```

```
  [-startoption start_options]
```

また、このコマンドでは、使用可能で実行されていないすべてのサービスが開始されます。そのサービスには、AUTOMATIC管理ポリシーが設定され、サービスのいずれかのロールがデータベース・ロールと一致します。

- 1つ以上のインスタンスを停止するには、コマンドラインから次のSRVCTL構文を入力します。

```
$ srvctl stop instance -db db_unique_name [-instance "instance_name_list" |
  -node node_name] [-stopoption stop_options]
```

複数のインスタンス名のカンマ区切りリストを指定して複数のインスタンスを停止することも、1つのノード名を指定して1つのインスタンスを停止することもできます。Windowsでは、カンマ区切りリストを二重引用符(""")で囲む必要があります。

このコマンドにより、インスタンスが実行されていたノード上で終了したインスタンスと関連するサービスも停止します。例のとおり、次のコマンドで、immediate stopオプションを使用してorclデータベースの2つのインスタンス(orcl3およびorcl4)を停止することができます。

```
$ srvctl stop instance -db orcl -instance "orcl3,orcl4" -stopoption immediate
```

関連項目

- サーバー制御ユーティリティのリファレンス

SRVCTLを使用した1つ以上のインスタンスおよびOracle RACデータベースの停止

SRVCTLを使用して、インスタンスとOracle RACデータベースを停止します。

Oracle RACインスタンスの停止手順は、次に説明する例外を除いて、非クラスタのOracle Databaseのインスタンスの停止と同じです。

- Oracle RACでは、1つのインスタンスを停止しても、実行中の他のインスタンスの操作を妨げることはありません。
- Oracle RACデータベースを完全に停止するには、データベースがオープン状態またはマウントされた状態となっているすべてのインスタンスを停止します。
- NORMALまたはIMMEDIATEでの停止後は、インスタンスのリカバリは不要です。ただし、SHUTDOWN ABORTコマンドを発行した後、またはインスタンスが異常終了した後は、リカバリが必要です。まだ実行中のインスタンスが、停止したインスタンスに対してインスタンス・リカバリを実行します。他に実行中のインスタンスがない場合は、次にデータベースをオープンするインスタンスが、リカバリが必要なすべてのインスタンスのリカバリを実行します。
- SHUTDOWN TRANSACTIONALコマンドとともにLOCALオプションを使用すると、特定のOracle RACデータベース・インスタンスを停止する場合に有用です。他のインスタンス上のトランザクションがこの操作を妨げることはありません。LOCALオプションを省略した場合、この操作は、SHUTDOWNコマンドを実行する前に起動した他のすべてのインスタンスのトランザクションがコミットまたはロールバックされるまで待機します。これは、Oracle RACデータベースのすべてのインスタンスを停止する場合は有効なアプローチです。

ノート:

SHUTDOWN TRANSACTIONAL と SHUTDOWN TRANSACTIONAL LOCAL の両方は、非クラスタ化データベースに対して同じアクションを実行しますが、この 2 つのコマンドは、Oracle RAC データベースでは異なってい

ます。

次のSRVCTL構文をコマンドラインから入力し、必要なデータベース名およびインスタンス名を指定するか、または複数のインスタンス名を指定して、複数の特定のインスタンスを停止します。

- クラスタ・データベース全体(すべてのインスタンスおよび有効になっているサービス)を停止するには、次のSRVCTLコマンドを入力します。

```
$ srvctl stop database -db db_unique_name [-stopoption stop_options]
```

TRANSACTIONAL停止オプションはsrvctl stop databaseコマンドとともに、TRANSACTIONAL LOCAL停止オプションはsrvctl stop instanceコマンドとともに使用します。

- 1つ以上のノードでOracle Clusterwareによって管理されているすべてのインスタンスおよび有効になっているサービスを停止するには、次のSRVCTLコマンドを入力します。

```
$ srvctl stop instance -node "node_list" [-stopoption stop_options]
```

- 1つ以上のインスタンスを停止するには、コマンドラインから次のSRVCTL構文を入力します。

```
$ srvctl stop instance -db db_unique_name {-node "node_list" | -instance "inst_name_list"}
   [-stopoption stop_options]
```

複数のインスタンス名のカンマ区切りリストを指定して複数のインスタンスを停止することも、1つのノード名を指定して1つのインスタンスを停止することもできます。Windowsでは、カンマ区切りリストを二重引用符("")で囲む必要があります。

このコマンドにより、インスタンスが実行されていたノード上で終了したインスタンスと関連するサービスも停止します。例のとおり、次のコマンドで、そこからサービスを実行する別のノードを検索するCRSのfailoverオプションと、immediate stopオプションを使用して、orclデータベースの2つのインスタンス(orcl3およびorcl4)を停止することができます。

```
$ srvctl stop instance -db orcl -instance "orcl3,orcl4" -failover -stopoption immediate
```

関連項目

- srvctl stop database
- srvctl stop instance
- データベースとインスタンスの停止の概要
- データベースの停止

CRSCTLを使用したすべてのデータベースおよびインスタンスの停止

ノード上でcrsctl stop crsコマンドを使用するか、crsctl stop cluster -allコマンドを使用してノード上またはクラスタ全体のすべてのインスタンスを停止できます。

ノード全体またはクラスタ全体を停止する場合(たとえばメンテナンス目的など)、必要なクラスタ権限があるときは、ノードでcrsctl stop crsコマンドを実行するか、crsctl stop cluster -allコマンドを実行します。これらのコマンドによって、サーバー上またはクラスタ内で実行されているすべてのデータベース・インスタンスは停止し、クラスタを再起動した後でその状態は確実にリカバリされます。CRSCTLを使用すると、Oracle Clusterwareは、他の場所で実行できるサービスおよび他のリソースを再配置することもできます。

これらのCRSCTLコマンドのいずれかを使用してサーバー上またはクラスタ内のすべてのデータベース・インスタンスを停止すると、

shutdown abortと同様にデータベース・インスタンスが停止される可能性があり、起動時にインスタンスのリカバリが必要になります。クラスタを停止する前にSRVCTLを使用してデータベース・インスタンスを手動で停止する場合は、shutdown abortを回避できますが、この場合は、Oracle Clusterwareの再起動後にデータベース・インスタンスを手動で再起動する必要があります。

SQL*Plusを使用した個々のインスタンスの起動と停止

ローカル・ノードに接続された状態で、1つのインスタンスのみを起動または停止するには、最初に現行の環境にローカル・インスタンスのSIDが含まれていることを確認する必要があります。

SQL*Plusセッションかどうかに関係なく、セッション内の後続のすべてのコマンドは、そのSIDに対応付けられます。

ノート:

この項では、SPFILE を使用していることを前提にしています。

ローカル・インスタンスを起動または停止するには、SQL*Plusセッションを開始し、SYSRACまたはSYSOPER権限で接続した後に、必要なコマンドを発行します。たとえば、ローカル・ノード上でインスタンスを起動しマウントする場合は、SQL*Plusセッション内で次のコマンドを実行します。

```
CONNECT / AS SYSRAC
STARTUP MOUNT
```

ノート:

Oracle ASM ディスク・グループを使用する場合、Oracle ASM インスタンスへの接続と管理には、SYSRAC 権限ではなく SYSASM 権限を使用します。

Oracle RAC 環境で Oracle ASM インスタンスを管理する場合は、SQL*Plus を使用しないことをお薦めします。Oracle Clusterware は、必要に応じて Oracle ASM インスタンスを自動的に管理します。手動操作が必要な場合は、それぞれの SRVCTL コマンドを使用します。

Oracle Net Servicesを使用して、単一のSQL*Plusセッションから複数のインスタンスを起動できます。Net Services接続文字列(通常は、tnsnames.oraファイルからのインスタンス固有の別名)を使用して、各インスタンスに順次接続します。

たとえば、ローカル・ノード上でSQL*Plusセッションを使用すると、インスタンスの個々の別名を使用して、各インスタンスに順次接続し、リモート・ノード上の2つのインスタンスのトランザクションの停止を実行できます。最初のインスタンスの別名をdb1、2つ目のインスタンスの別名をdb2と仮定します。次のコマンドを入力して、最初のインスタンスに接続してから停止します。

```
CONNECT /@db1 AS SYSRAC
SHUTDOWN TRANSACTIONAL
```

ノート:

正しいインスタンスに接続するには、接続文字列で 1 つのインスタンスにのみ対応付けられた別名を使用する必要があります。サービスに接続する TNS 別名を使用する接続文字列、または複数の IP アドレスを表示する Oracle Net アドレスを使用する場合、停止する特定のインスタンスに接続できないことがあります。

次のコマンドを入力して、SQL*Plusセッションから2つ目のインスタンスに接続した後、停止します。

```
CONNECT /@db2 AS SYSRAC
SHUTDOWN TRANSACTIONAL
```

SQL*Plusでは複数のインスタンスを同時に起動または停止することはできないため、単一のSQL*Plusコマンドでクラスタ・データベースのすべてのインスタンスを起動または停止することはできません。各インスタンスに順次接続し、起動および停止するスクリプトを作成することができます。ただし、インスタンスの追加または削除を行う場合は、このスクリプトを手動でメンテナンスする必要があります。

関連項目

- Oracle Automatic Storage Management管理者ガイド
- SQL*Plusユーザーズ・ガイドおよびリファレンス

Oracle RACでのPDBの起動および停止

プラガブル・データベース(PDB)の管理には、非CDBを管理するために必要なタスクのごく一部が必要です。

Oracle RACベースのマルチテナント・コンテナ・データベース(CDB)の管理は、非CDBの管理に似ています。違いは、ある管理タスクはCDB全体に適用され、ある管理タスクはCDBルートに適用され、ある管理タスクは特定のPDBに適用されるということです。この一部のタスクでは、ほとんどがPDBおよび非CDBに対して同じです。ただし、PDBのオープン・モードを変更する場合など、いくつかの違いがあります。また、PDB管理者は、単一PDBの管理のみを行い、CDB内の他のPDBによる影響は受けません。

サービスを管理することによって、Oracle RAC CDB内のPDBを管理します。これは、PDBがポリシー管理かまたは管理者管理かどうかに関係なく当てはまります。1つの動的データベース・サービスを各PDBに割り当てて、クラスタ化コンテナ・データベース内のインスタンスにわたってPDBの起動、停止および配置を調整します。

たとえば、prodというサーバー・プールにsparkというポリシー管理PDBを備えたraccontというCDBを所有している場合、次のコマンドを使用してplugというサービスをこのデータベースに割り当てます。

```
srvctl add service -db raccont -pdb spark -service plug -serverpool prod
```

サービスplugは、サーバー・プール内のすべてのノードにわたって均一に管理されます。同じサーバー・プールでこのサービスをシングルトン・サービスとして実行する場合は、前述のコマンドとともに-cardinality singletonパラメータを使用します。

PDB sparkを開くには、次のように、サービスplugを起動する必要があります。

```
srvctl start service -db raccont -service plug
```

サービスplugを停止するには:

```
srvctl stop service -db raccont -service plug
```

PDB sparkは、SQLコマンドALTER PLUGGABLE DATABASE PDB_NAME CLOSE IMMEDIATEを使用してPDBを閉じるまで開いたままです。srvctl status serviceコマンドを使用すると、データベースのステータスを確認できます。

PDBは動的データベース・サービスを使用して管理されるため、通常のOracle RACベースの管理プラクティスが適用されます。このため、サービスplugがオンライン状態で、このサービスをホストしているサーバー上でOracle Clusterwareが停止している場合、このサーバー上のOracle Clusterwareの再起動後に、サービスは元の状態にリストアされます。このようにして、PDBの起動は、他のOracle RACデータベースと同様に自動化されます。

ノート:

SQL*Plus とは異なり、SRVCTL は、クラスタ・データベース全体で動作します。したがって、サービスが同時に複数のサーバー上で実行されるように定義され、クラスタの現行のステータスがこの配置を可能にしている場合、サービスを使用した PDB の起動は、クラスタ化された CDB の複数のインスタンスに同時に適用されます。

関連項目

- 動的データベース・サービスによるワークロード管理

インスタンスの実行の確認

データベース・インスタンスが使用可能であることを確認するには、Oracle Enterprise Manager、SRVCTLまたはSQL*Plusを使用します。

- インスタンスが実行中であることを確認するためのSRVCTLの使用
- インスタンスが実行中であることを確認するためのSQL*Plusの使用

インスタンスが実行中であることを確認するためのSRVCTLの使用

SRVCTLを使用すると、特定のデータベースでインスタンスが実行中であることを確認できます。

次のコマンドは、mailというOracle RACデータベースのデータベース・インスタンスのステータスを確認するためのSRVCTLの使用例を示しています。

```
$ srvctl status database -db mail
```

このコマンドによって、次のような出力が返されます。

```
Instance mail1 is running on node beta1011Instance mail2 is running on node beta1010
```

また、次のようにして、割り当てられたサービスの可用性を確認することによって、クラスタ内でPDBが実行されているかどうかを確認できます。

```
$ srvctl status service -db db_unique_name -service service_name
```

インスタンスが実行中であることを確認するためのSQL*Plusの使用

SQL*Plusを使用すると、データベース・インスタンスが実行中であることを確認できます。

1. 任意のノードで、SQL*Plusプロンプトから、Net Services接続文字列(通常はtnsnames.oraファイルからのインスタンス固有の別名)を使用することによって、データベース・インスタンスに接続します。

```
CONNECT /@db1 as SYSRAC
```

2. 次の文を使用して、V$ACTIVE_INSTANCESビューを問い合せます。

```
CONNECT SYS/as SYSRAC
Enter password: password
SELECT * FROM V$ACTIVE_INSTANCES;
```

次のような出力が表示されます。

```
INST_NUMBER INST_NAME
-----------  ----------------
1            db1-sun:db1
2            db2-sun:db2
3            db3-sun:db3
```

次の表に、この例で出力される列を示します。

表3-2 V$ACTIVE_INSTANCES列の説明

列	説明
INST_NUMBER	インスタンス番号を識別します。
INST_NAME	ホスト名およびインスタンス名を host_name:instance_name として識別します。

特定のクラスタ・インスタンス上でのセッションの終了

ALTER SYSTEM KILL SESSION文を使用して、特定のインスタンスのセッションを終了できます。

セッションが停止すると、セッションのアクティブ・トランザクションがロールバックされ、そのセッションが保持していたリソース(ロックやメモリー領域など)がただちに解放されて、他のセッションで使用可能になります。

ALTER SYSTEM KILL SESSION文を使用すると、Oracle RAC環境で厳密なアプリケーション品質保証契約を維持できます。多くの場合、品質保証契約は、指定された期限内にトランザクションを実行することを目的としています。Oracle RAC環境では、このためには、指定された期限内にインスタンスでトランザクションを終了し、別のインスタンスでトランザクションを再試行することが必要な場合があります。

ノート:

当初アプリケーションがアプリケーション・コンティニュイティ対応の動的データベース・サービスを使用してデータベース・インスタンスに接続する場合、アプリケーション・コンティニュイティを使用して、トランザクションの取消しをユーザーから隠すことができます。

サービス・レベル管理に対してよりきめ細かいアプローチを求める場合、すべての Oracle RAC ベースのデータベースに対して、Oracle Database Quality of Service Management (Oracle Database QoS

Management)を使用することをお薦めします。

セッションを終了するには、次のステップに従います。

1. GV$SESSION動的パフォーマンス・ビューのINST_ID列の値を問い合せ、どのセッションを終了するかを特定します。
2. ALTER SYSTEM KILL SESSIONを発行し、GV$SESSION動的パフォーマンス・ビューを使用して特定したセッションのセッション索引番号(SID)とシリアル番号を指定します。

   ```
   KILL SESSION 'integer1, integer2[, @integer3]'
   ```

   - integer1には、SID列の値を指定します。
   - integer2には、SERIAL#列の値を指定します。
   - オプションのinteger3には、終了するセッションが存在するインスタンスのIDを指定します。GV$表を問い合せると、インスタンスIDを見つけることができます。

   この文を使用するには、インスタンスでデータベースがオープン状態であり、integer3を指定しない場合には、セッションと終了するセッションが同じインスタンスにある必要があります。

完了する必要があるアクティビティ(リモート・データベースからの応答の待機やトランザクションのロールバックなど)がセッションで実行されている場合、Oracle Databaseはそのアクティビティが完了するのを待機し、セッションに終了のマークを付けてから、ユーザーに制御を戻します。待機が数分間続く場合、セッションに終了予定のマークが付けられ、セッションに終了予定のマークを付けたというメッセージとともにユーザーに制御が戻されます。アクティビティが完了すると、PMONバックグラウンド・プロセスによってセッションに終了のマークが付けられます。

セッションの特定および終了の例

次の例に、ユーザーが特定のセッションを特定し、終了する3つのシナリオを示します。各例で、SYSDBAは、まずSCOTTユーザーのセッションのGV$SESSIONビューを問い合せて終了するセッションを特定し、次に、ALTER SYSTEM KILL SESSION文を実行してインスタンスでセッションを終了します。

例3-1 ビジー状態のインスタンスでのセッションの特定および終了

この例で、実行しているセッションがインスタンスINST_ID=1のSYSDBAであるとします。一部のアクティビティを完了してからセッションを終了する必要があるため、ORA-00031メッセージが戻されます。

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';
       SID    SERIAL#    INST_ID
---------- ---------- ----------
        80          4          2
SQL> ALTER SYSTEM KILL SESSION '80, 4, @2';
alter system kill session '80, 4, @2'
*
ERROR at line 1:
ORA-00031: session marked for kill
SQL>
```

例3-2 アイドル状態のインスタンスでのセッションの特定および終了

この例で、実行しているセッションがインスタンスINST_ID=1のSYSDBAであるとします。インスタンスINST_ID=2のセッションは、Oracle Databaseが60秒以内に文を実行すると、ただちに終了されます。

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';

       SID    SERIAL#    INST_ID
---------- ---------- ----------
        80          6          2

SQL> ALTER SYSTEM KILL SESSION '80, 6, @2';
System altered.
SQL>
```

例3-3 IMMEDIATEパラメータの使用

次の例には、未完了のアクティビティが完了するのを待機せずにただちにセッションを終了する、オプションのIMMEDIATE句が含まれています。

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';

       SID    SERIAL#    INST_ID
---------- ---------- ----------
        80          8          2

SQL> ALTER SYSTEM KILL SESSION '80, 8, @2' IMMEDIATE;

System altered.

SQL>
```

関連項目

- Oracle Database管理者ガイド
- Oracle Database 2日でパフォーマンス・チューニング・ガイド
- アプリケーション・コンティニュイティについて

Oracle RACでの初期化パラメータ・ファイルの概要

Oracle RACデータベースの初期化パラメータは、SPFILEに格納されます。

データベースを作成すると、Oracle Databaseにより、指定したファイルの場所にSPFILEが作成されます。この場所には、Oracle Automatic Storage Management (Oracle ASM)ディスク・グループまたはクラスタ・ファイル・システムを指定できます。手動でデータベースを作成する場合は、初期化パラメータ・ファイル(PFILE)からSPFILEを作成することをお薦めします。

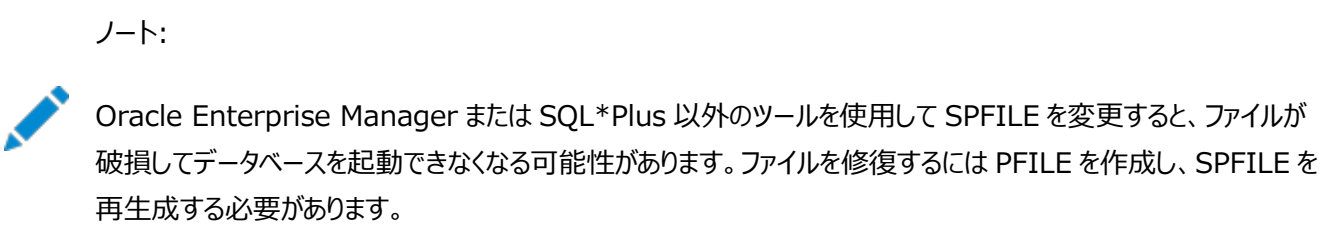
ノート:

Oracle RAC で従来の PFILE が使用されるのは、SPFILE が存在しないか、STARTUP コマンドで PFILE を指定した場合のみです。管理の単純化、パラメータ設定の一貫性の維持、データベースの停止および起動イベント全体にわたるパラメータ設定の永続性の保証のために、SPFILE を使用することをお薦めします。また、SPFILE をバックアップするように Oracle Recovery Manager (RMAN)を構成することもできます。

クラスタ・データベース内のインスタンスはすべて、起動時に同じSPFILEを使用します。SPFILEはバイナリ・ファイルであるため、エディタを使用して直接編集しないでください。かわりに、Oracle Enterprise ManagerまたはALTER SYSTEM SQL文を使用して、SPFILEパラメータ設定を変更します。

Oracle RACのSPFILEパラメータ値の設定

SPFILEの設定は、Oracle Enterprise ManagerまたはALTER SYSTEM文のSET句を使用して変更できます。



ノート:

Oracle Enterprise Manager または SQL*Plus 以外のツールを使用して SPFILE を変更すると、ファイルが破損してデータベースを起動できなくなる可能性があります。ファイルを修復するには PFILE を作成し、SPFILE を再生成する必要があります。

SPFILEはバイナリ・ファイルですが、この項の例では、ASCIIテキストとして記述してあります。次のエントリを含むSPFILEでインスタンスを起動するとします。

```
*.OPEN_CURSORS=500
prod1.OPEN_CURSORS=1000
```

SPFILEエントリのピリオド(.)の前にある値は、特定のパラメータの値が適用されるインスタンスを識別します。ピリオドの前にアスタリスク(*)がある場合、その値は、SPFILEに後続の個別の値が示されていないすべてのインスタンスに適用されます。

Oracleシステム識別子(SID)がprod1のインスタンスでは、データベース全体のパラメータが500に設定されていても、OPEN_CURSORSパラメータの設定は1000です。ワイルドカード文字のアスタリスク(*)が使用されたパラメータ・ファイルのエントリは、インスタンス固有のエントリがないインスタンスのみに適用されます。したがって、データベース管理者は、インスタンスprod1のパラメータ設定を制御できます。この2種類の設定は、パラメータ・ファイル内でいずれの順序でも指定できます。

別のDBAが次の文を実行した場合、SIDがprod1以外のすべてのインスタンスの設定がOracle Databaseによって更新されます。

```
ALTER SYSTEM SET OPEN_CURSORS=1500 sid='*' SCOPE=SPFILE;
```

これで、SPFILEにはOPEN_CURSORSの次のエントリが含まれます。

```
*.OPEN_CURSORS=1500
prod1.OPEN_CURSORS=1000
```

次の文を実行して、prod1を除くすべてのインスタンスのOPEN_CURSORSをデフォルト値にリセットします。

```
ALTER SYSTEM RESET OPEN_CURSORS SCOPE=SPFILE;
```

これで、SPFILEにはprod1の次のエントリのみが含まれます。

```
prod1.OPEN_CURSORS=1000
```

次の文を実行して、インスタンスprod1のみのOPEN_CURSORSパラメータをデフォルト値にリセットします。

```
ALTER SYSTEM RESET OPEN_CURSORS SCOPE=SPFILE SID='prod1';
```

Oracle RACでのパラメータ・ファイルの検索順序

Oracle Databaseでは、プラットフォームに応じてパラメータ・ファイルが特定の順序で検索されます。Oracle RACデータベースの場合、srvctl config databaseコマンドを使用すると、パラメータ・ファイルの場所を簡単に調べることができます。

LinuxおよびUNIXプラットフォームでは、検索順序は次のとおりです。

1. $ORACLE_HOME/dbs/spfilesid.ora
2. $ORACLE_HOME/dbs/spfile.ora
3. $ORACLE_HOME/dbs/initsid.ora

Windowsプラットフォームでは、検索順序は次のとおりです。

1. %ORACLE_HOME%¥database¥spfilesid.ora
2. %ORACLE_HOME%¥database¥spfile.ora
3. %ORACLE_HOME%¥database¥initsid.ora

ノート:

すべてのインスタンスで同じファイルを使用する必要があり、すべてのインスタンスの SID が異なっているため、デフォルトの SPFILE 名は使用しないことをお薦めします。かわりに、Oracle ASM に SPFILE を格納します。SPFILE をクラスタ・ファイル・システムに格納する場合は、SPFILE に$ORACLE_HOME/dbs/spfiledb_unique_name.ora というネーミング規則を使用してください。SPFILE=ORACLE_HOME/dbs/spfiledb_unique_name.ora という名前を含む、$ORACLE_HOME/dbs/initsid.ora という名前の PFILE を作成してください。

関連項目

- srvctl config database

サーバー・パラメータ・ファイルのバックアップ

リカバリのために、サーバー・パラメータ・ファイルを定期的にバックアップすることをお薦めします。

Oracle Enterprise Managerを使用してこれを実行するか、CREATE PFILE文を使用します。たとえば:

```
CREATE PFILE='/u01/oracle/dbs/test_init.ora'
FROM SPFILE='/u01/oracle/dbs/test_spfile.ora';
```

Recovery Manager (RMAN)を使用して、サーバー・パラメータ・ファイルのバックアップを作成できます。SPFILEは、クライアント側の初期化パラメータ・ファイルを使用してインスタンスを起動することによって、リカバリすることもできます。その後、CREATE SPFILE文を使用して、サーバー・パラメータ・ファイルを再生成します。この操作に使用されるパラメータ・ファイルがシングル・インスタンス用である場合、Oracle RACインスタンスで一意であっても、パラメータ・ファイルにはインスタンス固有の値は含まれません。したがって、前述のようにパラメータ・ファイルに適切な設定が含まれていることを確認してください。

通常のバックアップ操作の実行中にSPFILE(および制御ファイル)が自動的にRMANによってバックアップされるようにするには、Oracle Enterprise ManagerまたはRMANのCONTROLFILE AUTOBACKUP文を使用して、RMANの自動バックアップ機能を使用可能にします。

関連項目

- CREATE SPFILE
- 『Oracle Database Recovery Managerリファレンス』

Oracle RACでの初期化パラメータの使用

デフォルトでは、ほとんどのパラメータがデフォルト値に設定されていて、すべてのインスタンスで同じ値です。

ただし、多くの初期化パラメータに対しては、表3-3に記載されているとおり、各インスタンスで別々の値も設定できます。これ以外のパラメータは、次の項で説明されているように、一意または同一である必要があります。

- すべてのインスタンスで同じ値を設定する必要があるパラメータ
- すべてのインスタンスで一意の値を設定するパラメータ
- すべてのインスタンスで同じ値を設定する必要があるパラメータ

表3-3に、Oracle RACデータベースで特に使用される初期化パラメータのサマリーを示します。

表3-3 Oracle RACに固有の初期化パラメータ

パラメータ	説明
ACTIVE_INSTANCE_COUNT	この初期化パラメータは、Oracle RAC 11g リリース 2 (11.2)で非推奨になりました。かわりに、1 つの優先インスタンスと 1 つの使用可能なインスタンスを伴うサービスを使用します。
ASM_PREFERRED_READ_FAILURE_GROUPS	ミラー・データのコピーの読取り元の優先ディスクにする一連のディスクを指定します。このパラメータに設定する値はインスタンス固有で、すべてのインスタンスで同じにする必要はありません。
CLUSTER_DATABASE	クラスタ・モードで起動するデータベースを使用可能にするパラメータです。このパラメータを TRUE に設定します。
CLUSTER_DATABASE_INSTANCES	Oracle RAC はこのパラメータを使用して、十分なメモリー・リソースを割り当てます。すべてのインスタンスに同じ値を設定する必要があります。 ● ポリシー管理データベースの場合、16 が内部的に設定されます。 ● 管理者管理データベースの場合、構成済の Oracle RAC インスタンスの数に内部的に設定されます。 また、インスタンスを追加する場合、現行のインスタンスの数より大きい値をこのパラメータに設定できます。ポリシー管理データベースで、このパラメータにより大きい値を設定する必要があるのは、16 を超えるインスタンスでデータベースを実行する場合のみです。この場合、パラメータには、このデータベースを実行するインスタンスの予想最大数を設定します。
CLUSTER_INTERCONNECTS	複数のインターコネクトが存在する場合、プライベート・ネットワークの代替クラスタ・インターコネクトを指定します。

パラメータ	説明
	ノート: ● すべての Oracle Database と Oracle Clusterware では同じインターコネクト・ネットワークを使用することをお薦めします。 ● 特別な場合を除き、CLUSTER_INTERCONNECTS パラメータを設定することはお薦めしません。詳細は、「Linux および UNIX プラットフォームでの複数のクラスタ・インターコネクトの管理」を参照してください。 ● このパラメータは、グリッドのプラグ・アンド・プレイ環境のグリッド・プラグ・アンド・プレイ・プロファイルに格納されます。
DB_NAME	インスタンス固有のパラメータ・ファイルで DB_NAME の値を設定する場合は、すべてのインスタンスに同じ値を設定する必要があります。
DISPATCHERS	DISPATCHERS パラメータは、共有サーバー構成(多数のユーザー・プロセスが、非常に少数のサーバー・プロセスを共有できるように構成されたサーバー)を使用可能にするために設定します。共有サーバー構成では、多数のユーザー・プロセスがディスパッチャに接続します。DISPATCHERS パラメータには、多くの属性を含めることができます。 少なくとも、PROTOCOL 属性および LISTENER 属性を構成することをお薦めします。PROTOCOL には、ディスパッチャ・プロセスがリスニングのエンド・ポイントを生成するネットワーク・プロトコルを指定します。LISTENER には、Oracle Net Services リスナーの別名を指定します。別名には、tnsnames.ora ファイルなどのネーミング・メソッドを介して解決される名前を設定します。tnsnames.ora ファイルには、ネット・サービス名が記述されています。このファイルは、クライアント、ノードおよび Oracle Performance Manager ノード上で必要になります。Oracle Enterprise Manager では、Cloud Control のクライアントで tnsnames.ora エントリは必要ありません。 DISPATCHERS パラメータとその属性の構成、および共有サーバーの構成の詳細は、『Oracle Database Net Services 管理者ガイド』も参照してください。
GCS_SERVER_PROCESSES	この静的パラメータでは、Oracle RAC インスタンスのグローバル・キャッシュ・サービス(GCS)のサーバー・プロセスの初期数を指定します。GCS プロセスでは、Oracle RAC インスタンスのインスタンス間トラフィックのルーティングが管理されます。GCS サーバー・プロセスのデフォルト数は、最小が 2 で、システム・リソースに基づいて計算されます。CPU が 1 つのシステムでは、1 つの GCS サーバー・プロセスがあります。CPU が 2 つから 8 つのシステムでは、2 つの GCS サーバー・プロセスがあります。CPU が 9 つ以上あるシステムでは、GCS サーバー・プロセスの数は、CPU の数を 4 で割り、端数を切り捨てた数と同じになります。たとえば、CPU が 10 ある場合は 10 を 4 で割るため、システムには 2 つの GCS プロセスがあることになります。異なるインスタンスで、このパラメータを異なる値に設

パラメータ	説明
	定できます。
INSTANCE_NAME	一意のインスタンス名を指定します。クライアントは、この名前を使用して、セッションをクラスタ内の特定のインスタンスに強制的に接続します。通常、INSTANCE_NAME パラメータは db_unique_name_instance_number(orcldb_2 など)という形式になります。 ノート: グリッドのプラグ・アンド・プレイ環境では、INSTANCE_NAME パラメータは必須ではなく、指定しない場合は db_unique_name_instance_number がデフォルトになります。
RESULT_CACHE_MAX_SIZE	クラスタ化されたデータベースでは、すべてのインスタンスで RESULT_CACHE_MAX_SIZE=0 を設定して結果キャッシュを無効にするか、またはすべてのインスタンスで 0(ゼロ)以外の値を使用して結果キャッシュを有効にできます。結果キャッシュの有効と無効を切り替えるには、すべてのインスタンスを再起動する必要があります。 ● 結果キャッシュの有効化: RESULT_CACHE_MAX_SIZE を 0 より大きい値に設定するか、またはパラメータを未設定のままにします。個々のインスタンスで異なるキャッシュのサイズを指定できます。 ● 結果キャッシュの無効化: すべてのインスタンスで RESULT_CACHE_MAX_SIZE=0 を設定すると、結果キャッシュが無効になります。結果キャッシュの無効化はクラスタ全体で行う必要があるため、いずれか 1 つのインスタンスの起動時に RESULT_CACHE_MAX_SIZE=0 を設定した場合は、すべてのインスタンスで起動時にパラメータを 0(ゼロ)に設定する必要があります。一部のインスタンスで結果キャッシュを無効にすると、誤った結果になる場合があります。 RESULT_CACHE_MAX_SIZE パラメータを設定しない場合、パラメータはデフォルトの 0(ゼロ)以外の値に解決されます。
SERVICE_NAMES	サービスを使用する場合は、SERVICE_NAMES パラメータに値を設定せずに、かわりに、Oracle Enterprise Manager Cloud Control のクラスタ管理サービス・ページで、クラスタ管理されるサービスを作成する必要があります。これは、Oracle Clusterware が、ユーザーが作成したサービスおよびデフォルトのデータベース・サービスのこのパラメータの設定を制御するためです。「動的データベース・サービスによるワークロード管理」で説明するサービスの機能は、SERVICE_NAMES を設定する場合に使用可能な機能とは直接関係ありません。また、このパラメータに値を設定した場合、サービスの使用によって得られるメリットが失われてしまう可能性があります。 ノート: クライアント接続ではインスタンス名ではなくサービスを使用することをお薦めします。SERVICE_NAMES パラメータのエントリは、INSTANCE_NAME パラメータの値ではなく、クライアント接続で使用される場合があります。SERVICE_NAMES パラメータに 1 つ以上の名前が含まれ、異なるインスタンスで 1 つ以上の名前を共有する場合があるため、クライアントは、接続文字列で選択されたサービス名に応じて、特定のインスタンスまたは一連の

パラメータ	説明
	いずれかのインスタンスに接続できます。
SPFILE	SPFILE を使用する場合は、Oracle RAC データベースのすべてのインスタンスが SPFILE を使用し、このファイルが共有記憶域に存在する必要があります。
THREAD	インスタンスで使用される REDO スレッドの数を指定します。使用可能な未使用の REDO スレッド番号であれば、どれでも指定できます。指定する場合、このパラメータの値はすべてのインスタンスに対して一意である必要があります。INSTANCE_NAME パラメータを使用して REDO ログ・グループを指定することをお薦めします。

関連項目

- Oracle Databaseリファレンス

すべてのインスタンスで同じ値を設定する必要があるパラメータ

データベースの作成に重要な特定のパラメータ、または特定のデータベース操作に影響する特定のパラメータには、Oracle RACデータベースの各インスタンスで同じ値を設定する必要があります。

これらの初期化パラメータ値は、SPFILEに指定するか、または各インスタンスの個々のPFILEに指定します。次のリストに、すべてのインスタンスで同一である必要があるパラメータを示します。

- COMPATIBLE
- CLUSTER_DATABASE
- CONTROL_FILES
- DB_BLOCK_SIZE
- DB_DOMAIN
- DB_FILES
- DB_NAME
- DB_RECOVERY_FILE_DEST
- DB_RECOVERY_FILE_DEST_SIZE
- DB_UNIQUE_NAME
- INSTANCE_TYPE (RDBMSまたはASM)
- PARALLEL_EXECUTION_MESSAGE_SIZE
- REMOTE_LOGIN_PASSWORDFILE
- UNDO_MANAGEMENT

次のパラメータは、パラメータの値を0 (ゼロ)に設定する場合のみ、すべてのインスタンスで同じにする必要があります。

- DML_LOCKS
- RESULT_CACHE_MAX_SIZE

すべてのインスタンスで一意の値を設定するパラメータ

INSTANCE_NUMBERパラメータなど、特定のパラメータは各インスタンスに固有です。

Oracle Grid Infrastructure 21c以降、ポリシー管理データベースは非推奨です。

ポリシー管理データベースで一意の設定を持つパラメータを設定する必要がある場合は、データベースのサーバー・プールに割り当てられる各サーバーに対してsrvctl modify instance -n node_name -i instance_nameコマンドを実行することにより、インスタンスが特定のノードで常に同じ名前を使用するようにできます。その後、パラメータの一意の値をinstance_nameに指定できます(この値は、node_name上でデータベースが実行されるときに使用されます)。

データベース名と、インスタンスに割り当てられたINSTANCE_NAME番号で構成される環境変数ORACLE_SIDを指定します。

CLUSTER_INTERCONNECTS初期化パラメータを使用して、Oracle Clusterwareがプライベート・ネットワークに使用している代替インターコネクトを指定します。CLUSTER_INTERCONNECTS初期化パラメータを設定すると、Oracle RACデータベースの各インスタンスに対して一意の値が使用されます。

Oracle Databaseは、INSTANCE_NUMBERパラメータを使用して起動時にインスタンスを識別し、INSTANCE_NAMEパラメータを使用して特定のインスタンスにREDOログ・グループを割り当てます。インスタンス名はdb_unique_name_instance_numberの形式にすることが可能で、名前と番号がアンダースコアで区切られたこの形式の場合、アンダースコアの後の番号がINSTANCE_NUMBERとして使用されます。グリッドのプラグ・アンド・プレイを使用するOracle Database 11.2では、ポリシー管理データベースのインスタンス番号を明示的に割り当てる必要がなくなり、インスタンス名はデフォルトのdb_unique_name_instance_numberに設定され、Oracle Databaseによってインスタンス番号が割り当てられます。

自動UNDO管理を使用可能にしてUNDO_TABLESPACEを指定する場合、各インスタンスでこのパラメータに一意のUNDO表領域名を設定します。

ROLLBACK_SEGMENTSパラメータを使用する場合は、SPFILEでSID識別子を使用して、これらのパラメータに一意の値を設定することをお薦めします。ただし、各インスタンスのINSTANCE_NUMBERに一意の値を設定する必要があり、デフォルト値は使用できません。

ASM_PREFERRED_READ_FAILURE_GROUPS初期化パラメータを使用すると、優先読取り障害グループ名のリストを指定できます。これらの障害グループのディスクは、優先読取りディスクになります。したがって、すべてのノードはそのローカル・ディスクから読み取ることができます。この結果、効率およびパフォーマンスが向上し、ネットワーク・トラフィックが削減されます。このパラメータの設定はインスタンス固有で、すべてのインスタンスで同じにする必要はありません。

関連項目

- LinuxおよびUNIXプラットフォームでの複数のクラスタ・インターコネクトの管理

すべてのインスタンスで同じ値を設定する必要があるパラメータ

ここにリストされているパラメータをすべてのインスタンスで同じ設定にすることをお薦めします。

表3-4のパラメータには、すべてのインスタンスで同じ値を設定することをお薦めします。これらのパラメータにはインスタンスごとに異なる値を設定できますが、すべてのインスタンスでパラメータに同じ値を設定すると管理が簡単です。

表3-4 すべてのインスタンスで同じ値を設定する必要があるパラメータ

パラメータ	説明

パラメータ	説明
ARCHIVE_LAG_TARGET	Oracle RAC データベースのインスタンスごとに異なる値を設定すると、データベース処理によって追加の自動同期化が実行されるため、多くの場合、オーバーヘッドが増加します。 Oracle RAC データベースのダウンストリーム取得構成で、Oracle GoldenGate ダウンストリーム取得または Oracle GoldenGate 統合取得モードを使用する場合、値は 0 (ゼロ)より大きい必要があります。
CLUSTER_DATABASE_INSTANCES	このパラメータについては、すべての Oracle RAC データベース・インスタンスで同一の設定であることが望まれますが、これは必須ではありません。
LICENSE_MAX_USERS	このパラメータでは、データベースに定義されるユーザー数のデータベース全体における制限が決定されるため、このパラメータにはデータベースのすべてのインスタンスに同じ値を指定して、どのインスタンスの使用時にも現在の値を確認できるようにすると便利です。異なる値を設定すると、インスタンスの起動時に Oracle Database によって追加で警告メッセージが生成されたり、データベース・ユーザーの管理に関連するコマンドが一部のインスタンスで失敗する可能性があります。
LOG_ARCHIVE_FORMAT	すべてのインスタンスで同じ値を使用しない場合、メディア・リカバリが複雑になります。アーカイブ・ログ・ファイルを作成したインスタンスにかかわらず、リカバリを行うインスタンスでは、必要なアーカイブ・ログ・ファイル名のフォーマットがそのインスタンス自体の LOG_ARCHIVE_FORMAT の値で定義されていると想定します。 Oracle Data Guard をサポートするデータベースでは、アーカイブ REDO ログ・ファイルの送受信を行うために、すべてのインスタンスで LOG_ARCHIVE_FORMAT に同じ値を使用する必要があります。
SPFILE	すべてのインスタンスでこのパラメータに同じファイルを指定しない場合、各インスタンスは、フェイルオーバー、ロード・バランシングおよび通常の操作中に、異なる動作または予測できない動作を行う場合があります。また、ALTER SYSTEM SET または ALTER SYSTEM RESET コマンドで SPFILE に行う変更は、コマンドを実行したインスタンスで使用される SPFILE のみに保存されます。加えた変更は、別の SPFILE が使用されるインスタンスには反映されません。 サーバーによって値が設定されているインスタンスで SPFILE の値が異なる場合、デフォルトの SPFILE を使用していないインスタンスを再起動する必要があります。
TRACE_ENABLED	診断トレース情報を Oracle RAC データベースで常に使用可能にするには、すべてのデータベース・インスタンスで TRACE_ENABLED を TRUE に設定する必要があります。一部のインスタンスのみのトレースを行う場合、TRACE_ENABLED が FALSE に設定されているインスタンスのみにアクセス可能なとき、必要な診断情報を使用できない場合があります。

パラメータ	説明
UNDO_RETENTION	各インスタンスで UNDO_RETENTION に異なる値を設定すると、スケーラビリティが低下し、フェイルオーバー後に予測できない動作が行われる場合があります。したがって、このパラメータに Oracle RAC データベースのインスタンス間で異なる値を割り当てる前に、メリットがあるかどうかを慎重に考慮する必要があります。

管理者管理データベースのポリシー管理データベースへの変換

管理者管理データベースをポリシー管理データベースに変換できます。

ノート:

管理者管理データベースが権限の弱いユーザー用に構成されており、このデータベースをポリシー管理データベースに変換しようとする場合は、この権限の弱いユーザーにウォレットを(まだ存在しない場合)手動で追加し、Oracle Database 用の Windows サービスを作成できるようにする必要があります。

管理者管理データベースを変換するには:

1. すべてのサービスとデータベースの現在の構成を確認します(間違いがあったためにリカバリが必要な場合、元の構成がどうであったかを確認できます)。

```
srvctl config database -db db_unique_name
srvctl config service -db db_unique_name
```

2. ポリシー管理データベース用のサーバー・プールを作成します(これを実行するには、クラスタ管理者である必要があります)。

```
srvctl add srvpool -serverpool server_pool -min 0 -max n
```

前述のコマンドでは、0 (ゼロ)はサーバー・プールで希望するサーバーの最小数で、nは最大数です。

ノート:

このステップでは、必ずしも新しく作成したサーバー・プールにサーバーを配置するとはかぎりません。たとえば、新しいサーバー・プールがサーバーを割り当てることができる元の空きプールにサーバーがない場合は、変換が完了したときに、srvctl relocate server コマンドを使用して別のサーバー・プールからサーバーを再配置する必要がある可能性があります。

3. 次のようにOracle Enterprise ManagerまたはSRVCTLを使用して、データベースを停止します。

```
srvctl stop database -db db_unique_name
```

4. 新しいサーバー・プールに存在するようにデータベースを変更します。

```
srvctl modify database -db db_unique_name -serverpool server_pool
```

5. 次のように、サービス・ユーザーをウォレットに追加します。

```
crsctl add wallet -type OSUSER -user user_name -passwd
```

6. ステップ1のコマンドを繰り返してデータベースのステータスを確認し、データベースがポリシー管理になったことを確認します。

次のように、前述の手順で行った変更を認識させるために、Oracle Enterprise Managerを構成します。

1. Oracle Enterprise Manager Cloud Controlに新しいデータベース・インスタンスを認識させるために、インスタンス名をdb_unique_name#からdb_unique_name_#に変更(シャープ記号(#)の前にアンダースコア(_)を追加)する必要があります。
2. dbs/databaseディレクトリ内のorapwdファイルの名前を変更します(または、orapwdコマンドを実行して、新しいorapwdファイルを作成します)。

   デフォルトのorapwdファイルには、orapwdORCL1などのようにインスタンス名が付加されています。前述のステップで変更したインスタンス名に対応するように、ファイル名を変更する必要があります。たとえば、orapwdORCL1をorapwdORCL_1に変更するか、または新しいorapwdファイルを作成する必要があります。

ポリシー管理データベースを管理者管理データベースに直接変換することはできません。その場合は、srvctl remove databaseコマンドおよびsrvctl remove serviceコマンドでポリシー管理構成を削除してから、srvctl add databaseコマンドおよびsrvctl add instanceコマンドで同じデータベースを管理者管理データベースとして登録します。データベースおよびインスタンスの登録後に、srvctl add serviceコマンドを使用して、削除したサービスを元どおりに追加する必要があります。

管理者管理データベースのサービスは、引き続きPREFERREDおよびAVAILABLE定義によって定義されます。ポリシー管理データベースの場合、サービスはデータベース・サーバー・プールに対して定義され、uniform(サーバー・プール内のすべてのインスタンスで実行)またはsingleton(サーバー・プール内の単一インスタンスでのみ実行)のいずれかになります。データベースの管理ポリシーを変更した場合、選択したデータベースの管理ポリシーに応じて、UNIFORMかSINGLETON、またはPREFERRED/AVAILABLEのいずれかになるように、データベース・サービスを再作成する必要があります。

関連項目

- サーバー制御ユーティリティのリファレンス
- サービスのデプロイメント・オプション

データベース・サーバーのメモリー不足の管理

Memory Guardは、サーバー上のメモリー不足をリアルタイムで検出し、新しいセッションを他のサーバーにリダイレクトして、メモリーが不足しているサーバー上のすべての使用可能メモリーが使用されることを防ぎます。

オープン・セッションまたはランナウェイ・ワークロードが多すぎるため、エンタープライズ・データベース・サーバーが使用可能メモリーをすべて使用することがあります。メモリーが不足すると、トランザクションが失敗することがあります。極端な場合には、サーバーが再起動し、アプリケーションの貴重なリソースが失われることもあります。Memory Guardは、サーバー上のメモリー不足をリアルタイムで検出し、新しいセッションを他のサーバーにリダイレクトして、メモリーが不足しているサーバー上のすべての使用可能メモリーが使用されることを防ぎます。

新しいセッションを別のサーバーに再ルートすると、メモリーが不足しているサーバー上の既存のワークロードが保護され、サーバーを使用可能な状態に保つことができます。Memory Guardはサーバーのメモリー不足を管理するOracle RACの機能で、Oracle RACデータベースにホストされているアプリケーションのサービス・レベルの管理に新しいリソース保護機能を追加します。

Oracle Database Quality of Service Managementが有効になっていると、クラスタ・ヘルス・モニターは、クラスタ・サーバーのメモリー・リソースに関するリアルタイム情報を提供するメトリック・ストリームをメモリー・ガードに送信します。この情報の内容は次のとおりです。

- 使用可能メモリーの量
- 現在使用されているメモリーの量

ノードのメモリーが不足しているとMemory Guardが判断した場合は、新しい接続が作成されないように、そのノード上のOracle Clusterwareで管理されているデータベース・サービスが停止されます。メモリー不足が緩和されると、そのノードのサービスが自動的に再開し、リスナーはそのサーバーへの新規接続の送信を開始します。メモリー不足は、いくつかの方法で緩和できます(たとえば、既存のセッションの終了やユーザーの介入など)。

Oracle RACデータベースの静止

Oracle RACデータベースを静止する手順は、非クラスタ・データベースを静止する場合と同じです。

1つのインスタンスからALTER SYSTEM QUIESCE RESTRICTED文を使用します。データベースの静止処理中は、どのインスタンスからもデータベースを開くことはできません。DBA以外のすべてのセッションが非アクティブになると、ALTER SYSTEM QUIESCE RESTRICTED文は終了し、データベースは静止状態にあるとみなされます。Oracle RAC環境では、この文を発行したインスタンスのみでなく、すべてのインスタンスにこの文が適用されます。

Oracle RAC環境でALTER SYSTEM QUIESCE RESTRICTED文を正しく発行するには、データベース・リソース・マネージャ機能をアクティブにしておくだけでなく、その機能がクラスタ・データベースのすべてのインスタンスに対しても、インスタンス起動時からアクティブになっている必要があります。DBA以外のセッションがアクティブにならないようにするのは、データベース・リソース・マネージャの機能です。また、この文が適用されている間に現行のリソース・プランを変更しようとすると、システムが静止状態でなくなるまで、その変更はキューに入れられます。

次の条件がOracle RACに適用されます。

- ALTER SYSTEM QUIESCE RESTRICTED文を発行しても、Oracle Databaseがその処理を完了していない場合、データベースを開くことはできません。
- データベースが静止状態にある場合、そのデータベースを開くことはできません。
- ALTER SYSTEM QUIESCE RESTRICTED文およびALTER SYSTEM UNQUIESCE文は、このコマンドを発行したインスタンスのみでなく、Oracle RAC環境のすべてのインスタンスに適用されます。

ノート:

コールド・バックアップを実行するために静止状態を使用することはできません。データベースが静止状態にある場合でも、Oracle Database のバックグラウンド・プロセスが Oracle Database の内部処理のために更新を実行していることがあるためです。また、オンライン・データファイルのヘッダーは、引き続きアクセス中であるかのように見えます。このファイル・ヘッダーの状態は、データベースが正しく停止された場合とは異なります。データベースが静止状態の間でも、オンライン・バックアップ操作は実行できます。

関連項目

- Oracle Database管理者ガイド

- ALTER SYSTEM

LinuxおよびUNIXプラットフォームでの複数のクラスタ・インターコネクトの管理

LinuxおよびUNIXプラットフォームで実行されるOracle RAC環境では、CLUSTER_INTERCONNECTS初期化パラメータを使用して、Oracle Clusterwareがプライベート・ネットワーク用に使用しているものの代替インターコネクトを指定できます。

ノート:

CLUSTER_INTERCONNECTS 初期化パラメータは、冗長インターコネクトの使用によって提供されている高可用性IP (HAIP)アドレスに設定しないでください。HAIP は自動的に認識されます。

CLUSTER_INTERCONNECTSに複数の値を設定する場合、Oracle Databaseは、インターコネクトに指定するすべてのネットワーク・インタフェースを使用し、表示されているすべてのインターコネクトが動作している場合はロード・バランシングを提供します。このパラメータで複数のインターコネクトを定義する場合、データベースのすべてのインスタンスで同じ値(インターコネクトをリストする順序を含む)を使用する必要があります。

ノート:

オペレーティング・システム・レベルでデフォルトのインターコネクト設定を上書きする CLUSTER_INTERCONNECTS 初期化パラメータを設定することはお薦めしません。

かわりに、ベスト・プラクティスは、Oracle RACおよびOracle Real Application Clusters One Node 11gリリース2 (11.2)データベース以上向けのOracle Grid Infrastructure 11gリリース2 (11.2)で使用可能な冗長インターコネクトの使用を使用することです。Oracle Database 11gリリース2 (11.2)より前のデータベースの場合、オペレーティング・システムベースのネットワーク・ボンディング・テクノロジを使用して、クラスタ・インターコネクトとして使用するつもりのネットワーク・インタフェース・カード向けの高可用性(およびロード・バランシング)を可能にします。1つのクラスタ内で複数のデータベース・バージョンを使用する場合は、両方の手法を組み合せることができます。冗長インターコネクトの使用では、結合にかかわらず、オペレーティング・システム・レベルで提示されるインタフェースを使用します。結合テクノロジの詳細は、オペレーティング・システムのベンダーに問い合せてください。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

CLUSTER_INTERCONNECTSパラメータを設定するためのユース・ケース

CLUSTER_INTERCONNECTS初期化パラメータでは、IPアドレスが必要です。コロン(:)で区切って、複数のIPアドレスを指定できます。Oracle RACネットワーク・トラフィックは、指定されたIPアドレス間で分散されます。

ノート:

- ポリシー管理データベースを使用する場合、CLUSTER_INTERCONNECTS パラメータは設定しないことをお薦めします。
- すべてのデータベースと Oracle Clusterware で同じインターコネクト・ネットワークを使用することをお薦めします。

一般に、CLUSTER_INTERCONNECTSパラメータは、次の場合にのみ設定します。

- クラスタで複数のデータベースが実行され、インターコネクト・トラフィックを分離する必要があり、冗長インターコネクトの使用を使用しない場合。
- オペレーティング・システムによって高可用性が実現された単一のIPアドレスがあり、そのIPアドレスに安定したインタフェース名がない(再起動時に名前を変更できるなど)場合。

次の一般的な構成では、CLUSTER_INTERCONNECTSパラメータは設定しないでください。

- 冗長インターコネクトの使用を使用する場合。
- クラスタ・インターコネクトが1つのみ存在する場合。
- デフォルトのクラスタ・インターコネクトが、Oracle RACデータベースの帯域幅の要件を満たしている場合(通常は満たしています)。

CLUSTER_INTERCONNECTS初期化パラメータの指定時は、次のことに注意してください。

- CLUSTER_INTERCONNECTS初期化パラメータは、UDP IPCが使用可能なLinuxおよびUNIX環境でのみ有効です。
- パラメータ・ファイルでCLUSTER_INTERCONNECTS初期化パラメータを設定する場合は、Oracle RACデータベースの各インスタンスに対して異なる値を指定します。
- 異なるノードで、同じデータベースの異なるインスタンスに指定されたIPアドレスは、同一のインターコネクト・ネットワークに接続するネットワーク・アダプタに属する必要があります。
- このパラメータに対して複数のIPアドレスを指定する場合、同一データベースのすべてのインスタンスに対して、同じ順序でIPアドレスを指定します。たとえば、node1の最初のインスタンスのパラメータで、alt0:、fta0:およびics0:デバイスのIPアドレスをその順に指定する場合、node2の2つ目のインスタンスのパラメータでも同等のネットワーク・アダプタのIPアドレスをその順に指定する必要があります。
- CLUSTER_INTERCONNECTSパラメータに指定したインターコネクトへの書込み中にオペレーティング・システム・エラーが発生した場合は、他のインタフェースが使用可能な場合でも、Oracle Databaseによってエラーが戻されます。これは、Oracle Databaseとインターコネクトの間の通信プロトコルが、使用しているプラットフォームに大きく依存する場合があるためです。詳細は、ご使用のOracle Databaseのプラットフォーム固有のマニュアルを参照してください。

例

単一のクラスタ・インターコネクトで帯域幅の要件を満たすことができない場合は、CLUSTER_INTERCONNECTSの設定を考慮します。冗長インターコネクトの使用を使用できない1つ以上のデータベースから高いインターコネクト帯域幅を要求されているデータ・ウェアハウス環境では、このパラメータの設定が必要な場合があります。

たとえば、高いインターコネクト帯域幅の要件を持つ2つのデータベースがある場合は、オペレーティング・システムが提供するデフォルトのインターコネクトを無効にし、各サーバー・パラメータ・ファイルで次の構文を使用して、各データベースに異なるインター

コネクトを指定できます。ipnは、ドットで区切られた標準的な10進形式のIPアドレス(たとえば、144.25.16.214)です。

```
Database One: crm1.CLUSTER_INTERCONNECTS = ip1
Database Two: ext1.CLUSTER_INTERCONNECTS = ip2
```

高い帯域幅を必要とするデータベースがある場合は、次の構文を使用して複数のインターコネクトを指定できます。

```
CLUSTER_INTERCONNECTS = ip1:ip2:...:ipn
```

関連項目

- Oracle Databaseリファレンス

Oracle ClusterwareでのOracle RACデータベースの管理方法のカスタマイズ

これらの例は、Oracle ClusterwareによるOracle RACデータベースに対する制御を最小化するために使用します(アップグレード中に必要となる可能性があります)。

デフォルトでは、Oracle ClusterwareによってOracle RAC環境のデータベースの再起動が制御されます。たとえば、データベースのアップグレード中など、場合によっては、Oracle ClusterwareのOracle RACデータベースに対する制御レベルを最小限に抑える必要があることがあります。

ノート:

サード・パーティのクラスタウェアを使用する場合は、Oracle Clusterware を使用して Oracle RAC インスタンスを管理することをお薦めします。インスタンスを手動に設定し、そのインスタンスをサード・パーティのクラスタウェアで起動する場合は、データベース・インスタンスの監視および再起動にサード・パーティのクラスタウェアを使用しないでください。Oracle Clusterware がこれを行う必要があるためです。

システムの再起動時にOracle ClusterwareによるOracle RACデータベースの再起動を防止する場合、または障害が発生したインスタンスの2回目以降の再起動を回避する場合、制御の程度を定義する管理ポリシーを構成します。管理ポリシーには、AUTOMATIC (デフォルト)とMANUALの2つがあります。管理ポリシーをAUTOMATICに設定すると、データベースは、データベース・ホスト・コンピュータの再起動時に、前回の実行状態(起動または停止)に自動的にリストアされます。MANUALの場合、データベースは、データベース・ホスト・コンピュータの再起動時に、自動的に再起動されることはありません。MANUALに設定しても、Oracle Restartは、実行中のデータベースを監視し、障害発生時にデータベースを再起動します。

SRVCTLコマンドを使用して、次の例に示すとおり、Oracle Clusterwareの管理ポリシーを表示および変更します。

例1: 現行管理ポリシーの表示

次のコマンド構文を使用して、現行の管理ポリシーを表示します(ここで、db_unique_nameは、管理ポリシーを変更するデータベースの名前です)。

```
srvctl config database -db db_unique_name -all
```

例2: 現行の管理ポリシーの別の管理ポリシーへの変更

次のSRVCTLコマンド構文を使用して、現行の管理ポリシーをAUTOMATIC、MANUALまたはNORESTARTに変更します。

```
srvctl modify database -db db_unique_name -policy [AUTOMATIC | MANUAL | NORESTART]
```

このコマンド構文は、データベース・リソースのリソース属性を設定します。

例3: 新規データベース用の管理ポリシーの指定

srvctl add databaseコマンドを使用して新しいデータベースを追加する場合、次の例のように-policyパラメータを使用して管理ポリシーをAUTOMATIC、MANUALまたはNORESTARTのいずれかに指定できます(ここで、db_unique_nameはデータベース名です)。

```
srvctl add database -db db_unique_name -policy [AUTOMATIC | MANUAL | NORESTART]
   -oraclehome $ORACLE_HOME -dbname DATA
```

このコマンド構文によって、新しいデータベースがOracle Clusterwareに制御されるようになります。管理ポリシー・オプションを指定しない場合、Oracle Databaseによってデフォルト値automaticが使用されます。管理ポリシーを変更した後、Oracle Clusterwareリソースは、影響を受けたデータベースの新しい値を記録します。

関連項目

- srvctl config database
- srvctl modify database
- srvctl add database

Oracle Enterprise Managerの高度な管理

Oracle Enterprise Manager Cloud Controlを使用すると、Oracle RACデータベースのインストール、構成および監視を単一の場所で実行できます。

この項では、『*Oracle Database 2*日で*Real Application Clusters*ガイド』や「Oracle RACデータベースの監視およびチューニングの概要」で取り上げられていない高度な管理タスクについて説明します。

この項には次のトピックが含まれます:

- ノードおよびインスタンスの検出のためのOracle Enterprise Manager Cloud Controlの使用
- Oracle Enterprise Managerのその他の機能
- Oracle RACでのジョブおよびアラートの管理

ノードおよびインスタンスの検出のためのOracle Enterprise Manager Cloud Controlの使用

Oracle Enterprise ManagerでOracle RACデータベースおよびインスタンス・ターゲットを検出すると、監視と管理が可能になります。

Oracle Enterprise Manager Cloud Controlでは、Oracle Enterprise Managerコンソール・インタフェースを使用し

てOracle Real Application Clusters (Oracle RAC)データベースおよびインスタンス・ターゲットを検出できます。

Oracle RACデータベースが存在するクラスタにOracle Enterprise Manager Cloud Controlエージェントをインストールすると、Oracle RACデータベース・ターゲットがインストール時に検出されます。エージェントのインストール後にデータベースが作成される場合またはエージェントのインストール時にデータベースが自動的に検出されない場合は、コンソール・インタフェースを使用してターゲットを検出できます。

ノードおよびインスタンスを検出するには、次のようにOracle Enterprise Manager Cloud Controlを使用します。

1. Oracle Enterprise Managerにログインし、「ターゲット」タブをクリックします。
2. 「データベース」タブをクリックすると、使用可能なターゲットがすべて表示されます。「タイプ」列に、「クラスタ・データベース」というエントリを使用するOracle RACデータベースが表示されます。
3. ターゲット名を選択して「追加」をクリックし、このデータベース・ターゲットを追加します。「データベース・ターゲットの追加: ホストの指定」ページが表示され、このページではデータベース、リスナーおよびOracle Automatic Storage Management (Oracle ASM)を監視ターゲットとして追加できます。
4. 懐中電灯のアイコンをクリックして使用可能なホスト名を表示し、ホストを選択して「続行」をクリックします。「データベースの追加: ソースの指定」ページが表示されます。
5. 非クラスタ・データベースおよびリスナーのみを検出するか、またはすべてのクラスタ・データベース、非クラスタ・データベースおよびクラスタのリスナーを検出するようにOracle Enterprise Managerにリクエストし、「続行」をクリックします。
6. この手順で、再構成したクラスタ・データベースおよびそのすべてのインスタンスが検出されなかった場合は、「クラスタでターゲットが検出されました」ページでクラスタ・データベースおよび非クラスタ・データベースを手動で構成できます。

Oracle Enterprise Managerのその他の機能

Oracle Enterprise Manager 12c以降、Oracle Enterprise Managerには様々な管理機能が用意されています。

- Oracle Grid Infrastructure/Oracle RACプロビジョニング・デプロイメント・プロシージャでは、Oracle RAC 12cおよびOracle Grid Infrastructureをプロビジョニングします。また、この手順では、プロファイルと呼ばれる機能があり、入力を記録しておいて、その後に繰り返されるデプロイメントでこの記録を使用できます。
- 新しいプロシージャの動的前提条件により、My Oracle Supportに接続されていれば、Oracle Enterprise ManagerではOracle RACプロビジョニング用の最新の前提条件およびツールをダウンロードできます。
- 既存のクラスタ・データベースのワンクリック拡張機能では、Oracle RAC 12cスタックがサポートされるようになりました。
- 既存の「Oracle Real Application Clustersの削除/縮小」機能は、Oracle RAC 12cのクラスタで動作が保証されています。
- 既存の「Oracleデータベースのプロビジョニング」プロシージャでは、シングル・インスタンスのOracle Database 12cのプロビジョニングがサポートされるようになりました。
- 非クラスタ・データベース用のOracle Grid Infrastructure 12cをプロビジョニングするために、新しいデプロイメント・プロシージャ「スタンドアロン・サーバー用のOracle Grid Infrastructureのプロビジョニング」が導入されました。

Oracle RACでのジョブおよびアラートの管理

Oracle Enterprise Managerの「管理」タブは、Oracle RACデータベースに使用できます。

クラスタ・データベースの「ホーム」ページには、Oracle Real Application Clusters (Oracle RAC)データベースのすべてのインスタンスが表示され、サーバー管理のために自動ワークロード・リポジトリ(AWR)によって収集されたいくつかのOracle RAC固有の統計の集計が示されます。

その詳細を表示するために、インスタンス固有のページに移動する必要はありません。ただし、クラスタ・データベースの「ホーム」ページでは、稼働しているはずのインスタンスが停止したり、インスタンスで多数のアラートが発生している場合は、各アラートについてインスタンス固有のページにドリルダウンできます。

この項で後述する特定の管理タスクを実行するには、ターゲットOracle RACデータベースにログインし、クラスタ・データベースの「ホーム」ページに移動し、「管理」タブをクリックします。

Oracle RACでのジョブの管理

Oracle Enterprise Managerジョブは、データベース・レベルでもインスタンス・レベルでも管理できます。

たとえば、クラスタ・データベース・レベルでジョブを作成し、そのジョブをターゲットOracle Real Application Clusters (Oracle RAC)データベースのアクティブな任意のインスタンスで実行できます。または、インスタンス・レベルでジョブを作成し、それを作成した特定のインスタンスでのみ実行することもできます。障害が発生した場合、再起ジョブは残りのインスタンスで実行できます。

ジョブは、インスタンス・レベル、クラスタ・レベルまたはクラスタ・データベース・レベルで作成できるため、クラスタ・データベース内の使用可能ないずれのホストでもジョブを実行できます。これはスケジュールされるジョブにも適用されます。Oracle Enterprise Managerでは、ジョブ・アクティビティがActive、History、Libraryなどのカテゴリに分類されて表示されます。

オペレーティング・システムのスクリプトやSQLスクリプトの送信およびスケジュールされたジョブの調査には、「ジョブ」タブを使用します。たとえば、特定のOracle RACデータベースのためのバックアップ・ジョブを作成するには、次のようにします。

1. 「ターゲット」をクリックし、ジョブを作成するデータベースをクリックします。
2. ターゲット・データベースにログインします。
3. Oracle Enterprise Managerに「データベース・ホーム」ページが表示されたら、「メンテナンス」をクリックします。
4. Enterprise Managerのジョブ・ウィザードの各ページに入力し、ジョブを作成します。

Oracle Enterprise Managerを使用したOracle RACでのアラートの管理

Oracle Enterprise Managerを使用して、Oracle RAC環境のアラートを構成できます。

また、グローバル・キャッシュ変換、読取り一貫性要求など、Oracle RACデータベースの特殊なテストも構成できます。

Oracle Enterprise Managerでは、Oracle RAC環境のデータベース・レベルとインスタンス・レベルのアラートは区別されます。アーカイブ・ログ・アラートなど、インスタンス・レベル・アラートのアラートしきい値は、インスタンスのターゲット・レベルで設定できます。この機能により、パフォーマンスがしきい値を超えた場合、特定のインスタンスに関するアラートを受信できます。また、表領域に関するアラートの設定など、データベース・レベルでアラートを構成することもでき、各インスタンスで重複するアラートの受信を回避できます。

関連項目

- Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス

関連項目:

Oracle RACでのアラートの構成の例は、Oracle Technology Networkを参照してください。パッケージを使用してしきい値を構成する方法は、『*Oracle Database PL/SQL*パッケージおよびタイプ・リファレンス』を参照してください

Oracle Enterprise Managerでの定義済一時停止の使用

Oracle Real Application Clusters (Oracle RAC)データベースのすべての管理対象ターゲットについて一時停止(メンテナンス操作によって監視データに偏りが発生したり、不要なアラートが生成されないように、データベース監視を一時停止する期間のこと)を定義できます。

一時停止を定義すると、メンテナンスの実行中にアラートが発生しないようにできます。一時停止は、クラスタ・データベース全体について定義することも、クラスタ・データベースの特定のインスタンスについて定義することもできます。

4 Oracle RAC One Nodeの管理

Oracle Real Application Clusters One Node(Oracle RAC One Node)は、クラスタ内の1つのノードで実行されるOracle Real Application Clusters(Oracle RAC)データベースのシングル・インスタンスです。このオプションによって、Oracleでのデータベース統合の柔軟性が向上します。フェイルオーバーによる保護で高可用性を実現しながら、多くのデータベースを最小限のオーバーヘッドで1つのクラスタに統合でき、オンラインでのローリング・パッチ適用、オペレーティング・システムおよびOracle Clusterwareのローリング・アップグレードも可能になります。

この章の内容は次のとおりです。

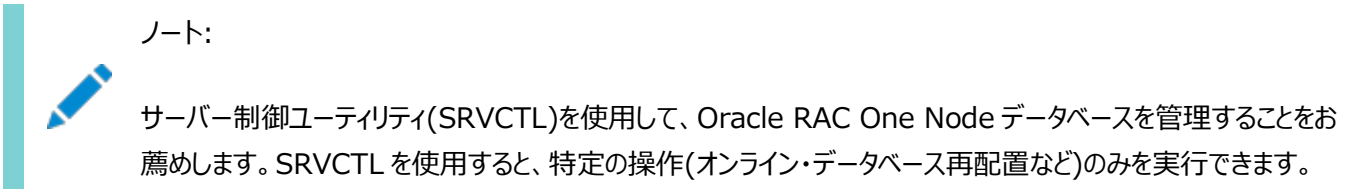
- Oracle RAC One Nodeデータベースの作成
- データベースの変換
- オンライン・データベース再配置

Oracle RAC One Nodeデータベースの作成

その他のOracle Databaseと同様に、フリート・パッチ適用およびプロビジョニングまたはDatabase Configuration Assistant (DBCA)を使用すると、Oracle RAC One Nodeデータベースを作成できます(手動で作成したスクリプトも有効な代替手段です)。

Oracle RAC One Nodeデータベースは、フリート・パッチ適用およびプロビジョニングと-dbtype RACONENODEパラメータを指定したrhpctl add databaseコマンドを使用することで作成できます。また、Oracle RAC One Nodeデータベースは、rhpctl add workingcopyコマンドを使用して組み込むこともできます。

Oracle RAC One Nodeデータベースは、単一インスタンスOracle DatabaseまたはOracle RACデータベースからの変換の結果になることもあります。通常、オラクル社提供のツールは、Oracle RAC One Nodeデータベースをを Oracle Clusterwareに登録します。構成が原因で、Oracle RAC One NodeデータベースのOracle Clusterwareへの自動登録は行われなかった可能性があります。この場合は、この項のステップに従って、Oracle RAC One Nodeデータベースを Oracle Clusterwareに登録してください。



ノート:

サーバー制御ユーティリティ(SRVCTL)を使用して、Oracle RAC One Node データベースを管理することをお薦めします。SRVCTL を使用すると、特定の操作(オンライン・データベース再配置など)のみを実行できます。

Oracle RAC One NodeデータベースがOracle Clusterwareに自動的に登録されなかった場合は、srvctl add databaseコマンドを使用してOracle RAC One Nodeデータベースをクラスタに追加します。たとえば:

```
$ srvctl add database -dbtype  RACONENODE [-server server_list]
  [-instance instance_name] [-timeout timeout]
```

管理者管理Oracle RAC One Nodeデータベースを追加する場合は、-serverオプションおよび-instanceオプションを使用します。

Oracle RAC One Nodeデータベースの場合、少なくとも1つの動的データベース・サービスを構成する必要があります(デフォルトのデータベース・サービスに加えて、かつ、これと反対に)。管理者管理Oracle RAC One Nodeデータベースを使用する場合は、他のOracle RACデータベースと同様、サービスの登録が実行されます。サービスをポリシー管理Oracle RAC One Nodeデータベースに追加する場合、SRVCTLは配置情報を受け入れませんが、かわりにSERVER_POOLS属性の値を使用してこれらのサービスを構成します。

ノート:

管理者管理 Oracle RAC One Node データベースを追加する場合は、オプションで、srvctl add database コマンドの-instance instance_name オプションを使用してインスタンス接頭辞を指定できます。こうすると、インスタンスの名前は、prefix_1 になります。インスタンス接頭辞を指定しないと、データベースの一意の名前の最初の 12 文字が接頭辞になります。インスタンス名は、オンライン・データベース再配置時に prefix_2 に変更され、後続のオンライン・データベース再配置時に prefix_1 に戻されます。フェイルオーバーでは、同じインスタンス名が使用されます。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- srvctl add database
- 対話モードでのDBCAによるターゲット・ノードへのデータベース・インスタンスの追加

データベースの変換

SRVCTLを使用して、インスタンスが1つのOracle RACデータベースをOracle RAC One Nodeデータベースに変換できます。その逆も同様に可能です。

この項には次のトピックが含まれます:

- Oracle RACからOracle RAC One Nodeへのデータベースの変換
- Oracle RAC One NodeからOracle RACへのデータベースの変換

Oracle RACからOracle RAC One Nodeへのデータベースの変換

SRVCTLを使用して、Oracle RACデータベースをOracle RAC One Nodeデータベースに変換します。

Oracle RACデータベースをOracle RAC One Nodeデータベースに変換する前に、Oracle RACデータベースのインスタンスが1つのみであることを確認する必要があります。Oracle RACデータベースが管理者管理であり、かつ複数のインスタンスがある場合は、srvctl remove instanceコマンドを使用して、1つを除くすべてのインスタンスを削除する必要があります。Oracle RACデータベースがポリシー管理であり、かつ複数のインスタンスがある場合は、srvctl stop instanceコマンドを使用して、1つを除くすべてのインスタンスを停止する必要があります。

Oracle RACデータベースが管理者管理である場合は、すべてのサービスの構成を変更して、変換後でもOracle RAC One Nodeデータベースであるようなインスタンスに優先インスタンスを設定する必要があります。サービスにPRECONNECT TAFポリシーがある場合は、変換プロセスを開始する前に、そのTAFポリシーをBASICまたはNONEに更新する必要があります。これらのサービスでは、使用可能インスタンスが不要になります。

Oracle RACデータベースがポリシー管理である場合は、すべてのサービスの構成を変更して、Oracle RACデータベースをOracle RAC One Nodeデータベースに変換する前に、すべてのサービスで同じサーバー・プールが使用されるようにしておく必要があります。

次のように、srvctl convert databaseコマンドを使用して、インスタンスが1つのOracle RACデータベースをOracle RAC One Nodeデータベースに変換できます。

```
$ srvctl convert database -db db_unique_name -dbtype RACONENODE
  [-instance instance_name -timeout timeout]
  -w timeout]
```

ノート:

Oracle RAC One Node に変換する Oracle RAC データベースでは、Oracle Managed Files を使用しているか(自動スレッド割当てを有効にするため)、または 2 つ以上の REDO スレッドを保持している必要があります。

関連項目

- srvctl remove instance
- srvctl stop instance
- srvctl convert database

Oracle RAC One NodeからOracle RACへのデータベースの変換

Oracle RAC One Nodeデータベース所有者としてログインして、次のSRVCTLコマンドを入力すると、Oracle RAC One NodeデータベースをOracle RACデータベースに変換できます。

```
srvctl convert database -db db_unique_name -dbtype RAC
```

オンライン・データベース再配置を使用してOracle RACに変換するデータベースを再配置している場合、またはオンライン・データベース再配置が失敗した場合は、srvctl convert databaseコマンドを実行する前に、再配置を終了するか、完了させる必要があります。

このコマンドを実行した後に、データベース・サーバー・プールの他に、各データベース・サービスのサーバー・プールを作成する必要があります。このデータベース・サービスが使用するサーバー・プールのSERVER_NAMESの値に、Oracle RAC One NodeからOracle RACノードに変換したノードが設定されている必要があります。CRSCTLユーティリティまたはOracle Enterprise Managerを使用すると、サーバー・プールを作成して構成できます。

管理者管理Oracle RAC One Nodeデータベースをを Oracle RACデータベースに変換すると、シングル・インスタンス・データベースがデータベース・サービスの優先インスタンスになるようにすべてのデータベース・サービスが構成されます。データベースを変換した後は、srvctl add instanceコマンドを実行してインスタンスをデータベースに追加できます。

ポリシー型管理のOracle RAC One Nodeデータベースをを Oracle RACデータベースに変換すると、すべてのデータベース・サービスのカーディナリティがUNIFORMに設定されます。また、このデータベースが現在実行されているサーバー・プールが再利用されることになります。変換では、このデータベースがサーバー・プール内のすべてのノードで実行されるように再構成されます。

このコマンドでは、追加のインスタンスは起動されませんが、srvctl start databaseコマンドを実行すると、サーバー・プール内のすべてノードでこのデータベースが起動します。

関連項目

- srvctl convert database

オンライン・データベース再配置

オンライン・データベース再配置機能を使用して、サービスの可用性を維持したまま、Oracle RAC One Nodeデータベースを別のノードに再配置できます。

データベースの新規ノードへの再配置中に、データベース・セッションを継続できるように、計画的なオンライン・データベース再配置中にのみOracle RAC One Nodeデータベースの第2インスタンスが作成されます。Oracle RAC One Nodeデータベースではオンライン・データベース再配置のみを使用できますが、管理スタイルにかかわらず(管理者管理またはポリシー管理)、Oracle RACデータベースではオンライン・データベース再配置は使用できません。

srvctl relocate databaseコマンドを使用して、再配置されるデータベースの開始およびサービスの移行後、データベースの以前のインスタンスが停止するまでの時間を構成できます。この構成時間は、操作全体にかかる時間の上限ではなく、再配置されるデータベースが以前のインスタンスから新しいインスタンスに接続が移行するのを待機した後、以前のインスタンスが停止するまでの時間を制御するだけです。

オンライン・データベース再配置は、次のように行われます。

1. 新しいデータベース・インスタンスを別の場所で開始します。
2. 再配置するインスタンスにすべてのサービスを移動します。
3. 再配置するインスタンスにすべての接続が移行するまで待機します。
4. 以前のデータベース・インスタンスを停止し、再配置するインスタンスに残りのすべての接続を強制移動します。

オンライン再配置のタイムアウトは、ステップ3を実行するために構成する時間です。

データベース・インスタンスのオンライン再配置を開始する前に、次のタスクを実行します。

- データベースの候補サーバー・リストに現在存在していないターゲット・ノードにデータベース・インスタンスを再配置する場合は、Oracle ASMに格納されている共有パスワード・ファイルを使用しないかぎり、パスワード・ファイル(構成されている場合)をターゲット・ノードにコピーする必要があります。
- 共有パスワード・ファイルを使用せずにOracle RAC One Nodeデータベースのリモート管理に対してパスワード・ファイルベースの認証を使用する場合、データベースを実行できるノードごとに、SID_prefix_1およびSID_prefix_2という名前の2つのパスワード・ファイルが必要です。パスワード・ファイルを更新するたびに、これらの両方のファイルをすべての候補ノードに再コピーする必要があります。これは、ポリシー管理型と管理者管理型の両方のデータベースに当てはまります。

  Oracle Clusterwareを使用してデータベースの起動および停止を行うこと、および他の管理用にデータ・ディクショナリにユーザーを定義することをお薦めします。
- オペレーティング・システムがMicrosoft Windowsの場合、データベース・インスタンスを再配置する前に、データベース・サービス・ユーザーがウォレットに追加されていることを確認する必要があります。crsctl query wallet -type

OSUSER -allを実行して、データベース・サービス・ユーザーがウォレットに存在するかどうかを確認します。存在しない場合、crsctl add wallet -type OSUSER -user user_name -passwdを実行して、データベース・サービス・ユーザーをウォレットに追加します。

srvctl relocate databaseコマンドを使用して、Oracle RAC One Nodeデータベースの再配置を実行します。たとえば:

```
$ srvctl relocate database -d rac1 -n node7
```

関連項目

- データベース・パスワード・ファイルの作成とメンテナンス
- srvctl relocate database

5 動的データベース・サービスによるワークロード管理

ワークロード管理には、ロード・バランシング、Oracle Real Application Clusters (Oracle RAC)のクライアントの有効化、分散トランザクション処理、およびサービスが含まれます。

この章の内容は次のとおりです。

- 接続ロード・バランシング
- ロード・バランシング・アドバイザ
- Oracle RACのクライアントの有効化
- Oracle RACの分散トランザクション処理
- 自動ワークロード・リポジトリ
- 自動ワークロード・リポジトリを使用したサービスのパフォーマンスの測定
- 自動ワークロード・リポジトリ・サービスのしきい値とアラート
- Oracleサービスの使用方法
- サービスのデプロイメント・オプション
- サービスの管理
- グローバル・サービス
- サービスへの接続: 例

接続ロード・バランシング

Oracle Net Servicesでは、Oracle RAC構成内のインスタンス間でクライアント接続を分散する機能を使用できます。

実装可能なロード・バランシングには、クライアント側とサーバー側の2種類のロード・バランシングがあります。クライアント側のロード・バランシングでは、接続要求は各クライアントから独立してリスナーをまたいで分散されます。サーバー側のロード・バランシングの場合、SCANリスナーはサービスの-clbgoalおよび-rlbgoal設定に基づいて、現在サービスを提供している最適なインスタンスに接続要求を送ります。

SCANリスナーはHTTPプロトコルを認識するため、HTTPクライアントを適切なハンドラ(クラスタ内のSCANリスナーが存在するノードとは別のノードに存在する可能性がある)にリダイレクトできます。

Oracle RACデータベースのクライアント接続では、両方のタイプの接続ロード・バランシングを使用する必要があります。

- サーバー側のロード・バランシング
- 一般的なデータベース・クライアント
- クライアント側のロード・バランシング
- 古いクライアント用のクライアント側の接続構成

関連項目

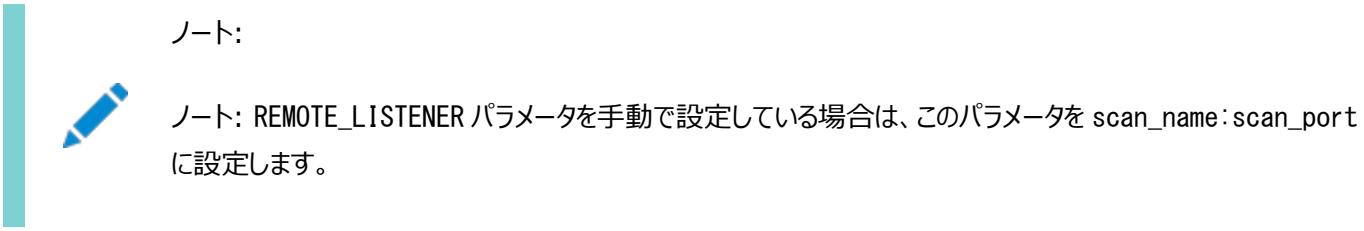
- Oracle Database Net Services管理者ガイド

サーバー側のロード・バランシング

DBCAを使用してOracle RACデータベースを作成すると、次の処理が自動的に実行されます。

- サーバー側のロード・バランシングの構成および有効化
- サーバー上のtnsnames.oraファイルにおけるクライアント側のロード・バランシング接続定義のサンプルの作成

Oracle Clusterwareデータベース・エージェントの役割は、LISTENER_NETWORKSパラメータの管理です。



ノート:

ノート: REMOTE_LISTENER パラメータを手動で設定している場合は、このパラメータを scan_name:scan_port に設定します。

FAN、高速接続フェイルオーバーおよびロード・バランシング・アドバイザは、正確な接続時ロード・バランシング構成(サービスに対する接続時ロード・バランシングの目標の設定など)に基づいて処理を実行します。接続時ロード・バランシングでは、LONGまたはSHORTのいずれかの目標を使用できます。これらの目標の特性は次のとおりです。

- SHORT: SHORT接続時ロード・バランシング方式は、ランタイム・ロード・バランシングを使用するアプリケーションに使用します。ロード・バランシング・アドバイザに統合されている接続プールを使用する場合は、CLB_GOALをSHORTに設定します。次の例では、SRVCTLを使用して、サービスoltpappを変更し、接続時ロード・バランシングの目標にSHORTを設定しています。

  ```
  $ srvctl modify service -db db_unique_name -service oltpapp -clbgoal SHORT
  ```

- LONG: LONG接続時ロード・バランシング方式は、ランタイム・ロード・バランシングが必要でない場合に使用します。このことは、バッチ操作の場合に一般的です。LONGは、デフォルトの接続時ロード・バランシングの目標です。次の例では、SRVCTLを使用してサービスbatchconnを変更し、長時間セッション用の接続時ロード・バランシングの目標を定義しています。

  ```
  $ srvctl modify service -db db_unique_name -service batchconn -clbgoal LONG
  ```

一般的なデータベース・クライアント

Oracle Net Servicesを使用すると、CONNECT_TIMEOUT、RETRY_COUNT、およびTRANSPORT_CONNECT_TIMEOUTパラメータをtnsnames.ora接続文字列に追加できます。

たとえば、SCANアドレスをデータベースのリモート・リスナーに使用する場合は次のようになります。

```
jdbc:oracle:thin:@(DESCRIPTION =
 (TRANSPORT_CONNECT_TIMEOUT=3)(CONNECT_TIMEOUT=60)
 (RETRY_COUNT=3)(FAILOVER=ON)
 (ADDRESS_LIST =(ADDRESS=(PROTOCOL=tcp)
  (HOST=CLOUD-SCANVIP.example.com)(PORT=5221))
 (CONNECT_DATA=(SERVICE_NAME=orcl)))
Remote_listeners=CLOUD-SCANVIP.example.com:5221
```

たとえば、データベースのVIPを指すリモート・リスナーを使用する場合は次のようになります。

```
jdbc:oracle:thin:@(DESCRIPTION =
 (TRANSPORT_CONNECT_TIMEOUT=3)
 (CONNECT_TIMEOUT=60)(RETRY_COUNT=20)
 (RETRY_DELAY=3)(FAILOVER=ON)
 (ADDRESS_LIST=
 (ADDRESS=(PROTOCOL=tcp)(HOST=CLOUD-VIP1)(PORT=1521) )
 (ADDRESS=(PROTOCOL=tcp)(HOST=CLOUD-VIP2)(PORT=1521) )
 (ADDRESS=(PROTOCOL=tcp)(HOST=CLOUD-VIP3)(PORT=1521) ))
 (CONNECT_DATA=(SERVICE_NAME=GOLD)))
```

これらのパラメータの値を表す単位は秒です。前述の例では、Oracle Netは各完全接続が応答を受信するのを60秒待機した後、障害が発生したと想定して、ADDRESS_LISTの次のリストを再試行します。Oracle Netは、アドレス・リストを3回試行すると、クライアントに失敗メッセージを返します。TRANSPORT_CONNECT_TIMEOUTパラメータは、データベース・サーバーへのTCP接続の確立を待機する時間を設定します。

SCANの場合、クライアントに失敗を返す前に、(SCANにより返される) 3つのアドレスすべてがOracle Net Servicesによって試行されます。EZConnectとSCANを併用した場合、この接続のフェイルオーバー機能が使用できます。

この動作は、Oracle Net接続フェイルオーバーと呼ばれます。リスト内の選択されたアドレスからエラーが戻されると、Oracle Net Servicesによってリストの次のアドレスが試行され、接続が成功するか、またはリスト内に試行するアドレスがなくなるまで続けられます。

古いクライアント用のクライアント側の接続構成

クライアント側のロード・バランシングに加えて、Oracle Net Servicesには接続フェイルオーバーが含まれています。リスト内の選択されたアドレスからエラーが戻されると、Oracle Net Servicesによってリストの次のアドレスが試行され、接続が成功するか、またはリスト内に試行するアドレスがなくなるまで続けられます。SCANの場合、クライアントに失敗を返す前に、Oracle Net Servicesによって3つのアドレスすべてが試行されます。EZConnectとSCANを併用した場合、この接続のフェイルオーバー機能が使用できます。

可用性を高めるために、Oracle Netがエラーを戻すまでにリスナーからの応答を待機する時間のタイムアウトを指定できます。このタイムアウト・パラメータを設定する方法は、クライアント・アクセスのタイプによって異なります。Oracle Netは、下位互換性のためにこれらのパラメータを維持します。

この項には次のトピックが含まれます:

- JDBC-Thinクライアント
- OCIクライアント

JDBC-Thinクライアント

次のようにoracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STRプロパティを設定することによって、遅延を回避できます。

```
Properties prop = new Properties ();
prop.put (oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR,
"" + (1 * 1000)); // 1 second
dbPools[ poolIndex ].setConnectionProperties ( prop );
```

パラメータの値は、ミリ秒単位で指定します。このため、アプリケーションが再度接続を試みた場合のタイムアウトを500Msに短

縮できます。

OCIクライアント

OCIクライアントには、クライアント側でローカルのsqlnet.oraファイルを作成します。

次の行を追加してこのファイルに接続タイムアウトを構成します。

```
sqlnet.outbound_connect_timeout = number_of_seconds
```

OCIクライアントのタイムアウト値の粒度は秒単位です。sqlnet.oraファイルは、このクライアントを使用するすべての接続に適用されます。

ノート:

サーバーの sqlnet.ora ファイルには接続タイムアウトを構成しないでください。

関連項目

- Oracle Call Interfaceプログラマーズ・ガイド

クライアント側のロード・バランシング

クライアント側のロード・バランシングは、パラメータLOAD_BALANCE=ONを設定して、クライアントの接続定義(tnsnames.oraファイルなど)に定義します。このパラメータをONに設定した場合は、Oracle Databaseによってアドレス・リストから無作為にアドレスが選択されて、そのノードのリスナーに接続されます。これによって、クラスタ内で使用可能なSCANリスナー間で、クライアント接続が均等に分散されます。

接続要求用にSCANを構成した場合、クライアント側ロード・バランシングは、SCANアクセスをサポートするクライアントには関係しません。クライアントがSCANを使用して接続する場合、EZConnectを使用していないかぎり、Oracle NetはSCANに対して定義された3つのIPアドレスの間でクライアント接続要求の負荷を自動的に均等に分散します。

SCANリスナーは、(-clbgoalがSHORTに設定されている場合は)最もロードされていないインスタンスのローカル・リスナーに接続要求をリダイレクトし、要求されたサービスを提供します。接続要求を受信したリスナーは、要求されたサービスを提供するとリスナーが認識しているインスタンスにユーザーを接続します。リスナーがサポートしているサービスを確認するには、lsnrctl servicesコマンドを実行します。

クライアントがSCANを使用して接続する場合、Oracle NetはSCANに対して定義された3つのIPアドレスの間でクライアント接続要求のロード・バランシングを自動的に行います。ただし、EZConnectを使用している場合は行いません。

SCANをサポートしていないクライアントを使用している場合(クライアント・バージョンがOracle Database 11gリリース2(11.2)よりも前の場合など)、SCANを使用するには、SCAN VIPを含めるようにクライアントtnsnames.oraを変更し、LOAD_BALANCE=ONを設定してVIP間の要求を均等に分散するようにする必要があります。たとえば:

```
Sales.example.com=(DESCRIPTION=
  (ADDRESS_LIST=(LOAD_BALANCE=ON)(FAILOVER=ON)
    (ADDRESS=(PROTOCOL=TCP)(HOST=172.22.67.192)(PORT=1521))
```

```
    (ADDRESS=(PROTOCOL=TCP)(HOST=172.22.67.193)(PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=172.22.67.194)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=salesservice.example.com))
  )
```

ノート:

データベースがOracle Database 11gリリース2 (11.2)以上で、SCANを使用する場合は、SCAN VIPをREMOTE_LISTENERパラメータに追加して、適切なリスナー・クロス登録を有効にします。

関連項目

- Oracle Databaseリファレンス

ロード・バランシング・アドバイザ

この項では、ロード・バランシング・アドバイザについて説明します。内容は次のとおりです。

- ロード・バランシング・アドバイザの概要
- ロード・バランシング・アドバイザを使用する環境の構成
- ロード・バランシング・アドバイザのFANイベント
- ロード・バランシング・アドバイザのFANイベントの監視

ロード・バランシング・アドバイザの概要

ロード・バランシングは、使用可能なすべてのOracle RACデータベース・インスタンス間で作業を分散します。アプリケーションでは、特定のサービスを提供するインスタンス間をまたがって実行される、接続が永続的な接続プールを使用することをお薦めします。永続接続を使用すると、接続が作成される頻度は低く、長期間存在します。作業は頻繁にシステムに送られ、この接続を利用し、比較的短時間存続します。ロード・バランシング・アドバイザは、受信した作業に対して最適なサービス・クオリティを提供するインスタンスにその作業を転送する方法についてのアドバイスを提供します。これにより、後で作業を再配置する必要性が最小化されます。

ロード・バランシング・アドバイザおよびランタイム接続ロード・バランシングの目標を使用することで、フィードバックはシステムに組み込まれます。作業は、システム全体で最適なサービス時間が実現されるようにルーティングされ、システムの状態変化に透過的に対応します。安定した状態のシステムでは、Oracle RACのすべてのインスタンスでスループットが向上した状態が維持されるようになります。

ロード・バランシング・アドバイザを使用できる標準アーキテクチャには、接続時ロード・バランシング、トランザクション処理モニター、アプリケーション・サーバー、接続コンセントレータ、ハードウェアおよびソフトウェアのロード・バランサ、ジョブ・スケジューラ、バッチ・スケジューラおよびメッセージ・キューイング・システムが含まれます。これらすべてのアプリケーションでは、作業を割り当てることができます。

ロード・バランシング・アドバイザは、リスナー、JDBCユニバーサル接続プール、OCIセッション・プール、Oracle WebLogic Server Active GridLink for Oracle RAC、ODP.NET接続プールなどの主要なOracleクライアントとともにデプロイされま

す。また、サードパーティ・アプリケーションは、JDBCとOracle RAC FAN APIを使用するか、またはOCIでコールバックを使用して、ロード・バランシング・アドバイザ・イベントをサブスクライブすることもできます。

ロード・バランシング・アドバイザを使用する環境の構成

ロード・バランシングを有効にする各サービスにサービス・レベルの目標を定義して、ロード・バランシング・アドバイザを使用するように環境を構成できます。

サービス・レベルの目標を構成すると、ロード・バランシング・アドバイザが有効になり、そのサービスのFANロード・バランシング・イベントのパブリッシュが可能になります。ランタイム接続のロード・バランシングにおけるサービス・レベルの目標値には、次の2つのタイプがあります。

- SERVICE_TIME: 応答時間に基づいて、作業要求をインスタンスに割り当てます。ロード・バランシング・アドバイザのデータは、サービスで完了した作業の経過時間およびサービスに対して使用可能な帯域幅に基づきます。SERVICE_TIMEの使用例としては、需要が変動するインターネット・ショッピングなどのワークロードがあります。次の例は、onlineサービスを使用して、目標を接続のSERVICE_TIMEに設定する方法を示します。

```
$ srvctl modify service -db db_unique_name -service online
  -rlbgoal SERVICE_TIME -clbgoal SHORT
```

- THROUGHPUT: スループットに基づいて、作業要求をインスタンスに割り当てます。ロード・バランシング・アドバイザのデータは、サービスで完了した作業の処理速度およびサービスに対して使用可能な帯域幅に基づきます。THROUGHPUTの使用例としては、前のジョブが完了してから次のジョブを開始するバッチ処理などのワークロードがあります。次の例は、sjobサービスを使用して、目標を接続のTHROUGHPUTに設定する方法を示します。

```
$ srvctl modify service -db db_unique_name -service sjob
  -rlbgoal THROUGHPUT -clbgoal LONG
```

ランタイム接続ロード・バランシングの目標をNONEに設定すると、サービスのロード・バランシングが無効になります。データ・ディクショナリのサービスの目標設定は、DBA_SERVICESビュー、V$SERVICESビューおよびV$ACTIVE_SERVICESビューを問い合せて確認できます。また、Oracle Enterprise Managerを使用して、サービスのロード・バランシング設定を確認することもできます。

関連項目

- サービスの管理

ロード・バランシング・アドバイザのFANイベント

ロード・バランシング・アドバイザのFANイベントでは、ロード・バランシング・アルゴリズムのメトリックが提供されます。

このイベントを使用する最も簡単な方法は、JDBC、Universal Connection Pool (または非推奨の暗黙的な接続キャッシュ)、ODP.NET接続プール、OCIセッション・プール、Oracle WebLogic Server Active GridLink for Oracle RACなどのOracle統合クライアントのランタイム接続ロード・バランシング機能を使用することです。他のクライアント・アプリケーションは、Oracle RAC FAN APIを使用することでFANをプログラムで利用し、FANイベントをサブスクライブしたり、受信時にイベント処理アクションを実行できます。表5-1に、ロード・バランシング・アドバイザのFANイベント・パラメータを示します。

関連項目:

Oracle RAC FAN APIの詳細は、『*Oracle Database JDBC*開発者ガイド』を参照してください。

表5-1 ロード・バランシング・アドバイザのFANイベント

パラメータ	説明
VERSION	イベント・レコードのバージョン。リリースの変更を識別するために使用されます。
EVENT_TYPE	ロード・バランシング・アドバイザ・イベントは、常に SERVICEMETRICS イベント・タイプです。
SERVICE	サービス名。DBA_SERVICES の NAME の値に該当します。
DATABASE	サービスをサポートしている一意のデータベースを示し、DB_UNIQUE_NAME 初期化パラメータの値と一致します。このパラメータのデフォルト値は、初期化パラメータ DB_NAME の値です。
INSTANCE	サービスをサポートしているインスタンスの名前であり、ORACLE_SID の値と一致します。
PERCENT	そのデータベース・インスタンスに送信する作業要求の割合。
FLAG	サービスの目標を基準としたサービス品質。有効な値は、GOOD、VIOLATING、NO DATA および BLOCKED です。
TIMESTAMP	通知イベントをオーダーする場合に使用するローカル・タイム・ゾーン。

ノート:

INSTANCE、PERCENT および FLAG イベント・パラメータが、サービスを提供する各インスタンスに生成されます。インスタンス・データの各セットは、中カッコ({})で囲まれます。

関連項目

- 『Oracle Database JDBC開発者ガイド』

ロード・バランシング・アドバイザのFANイベントの監視

ロード・バランシング・アドバイザのFANイベントの内部キュー表に対して次の問合せを使用すると、インスタンス用に生成されたロード・バランシング・アドバイザ・イベントを監視できます。

```
SET PAGES 60 COLSEP '|' LINES 132 NUM 8 VERIFY OFF FEEDBACK OFF
COLUMN user_data HEADING "AQ Service Metrics" FORMAT A60 WRAP
BREAK ON service_name SKIP 1
SELECT
 TO_CHAR(enq_time, 'HH:MI:SS') Enq_time, user_data
 FROM sys.sys$service_metrics_tab
 ORDER BY 1 ;
```

この問合せの結果には、次のような行が含まれます。

```
02:56:05|SYS$RLBTYP('hr', 'VERSION=1.0 database=sales service=hr
   { {instance=sales_4 percent=38 flag=GOOD aff=TRUE}{instance=sales_1
   percent=62 flag=GOOD aff=TRUE} } timestamp=2012-07-16 07:56:05')
```

次に、Oracle RAC FAN APIを使用してOracle Notification Serviceから取得された、2つのインスタンス(orcl1およびorcl2)で提供されるlba_servサービスのロード・バランシング・アドバイザ・イベントの例を示します。

```
Notification Type: database/event/servicemetrics/lba_serv.example.com
   VERSION=1.0 database=orcl service=lba_serv.example.com { {instance=orcl2
   percent=50 flag=UNKNOWN aff=FALSE}{instance=orcl1 percent=50 flag=UNKNOWN
   aff=FALSE} } timestamp=2012-07-06 13:19:12
```

ノート:

SERVICMETRICS イベントは、FAN コールアウト・メカニズムを介しては認識されません。

Oracle RACのクライアントの有効化

Oracle RACデータベースへの接続に使用される多くの一般的なクライアント・アプリケーション環境は、FANと統合されています。そのため、FANを使用する最も簡単な方法は、Oracle統合クライアントを使用することです。

次の項では、FANをOracleクライアントと統合する方法と、いくつかの特定のクライアント開発環境でFANイベントを有効にする方法について説明します。

- Oracle統合クライアントとFANの概要
- JDBC-Thinクライアントでの高速接続フェイルオーバーの有効化
- JDBCクライアントでのランタイム接続ロード・バランシングの有効化
- JavaのアプリケーションコンティニュイティのためのJDBC-Thinクライアントの構成
- トランザクション・ガード用のJDBC-Thinクライアントの構成
- OCIクライアントでの高速接続フェイルオーバーの有効化
- OCIクライアントでのランタイム接続ロード・バランシングの有効化
- トランザクション・ガードを使用するためのOCIクライアントの構成
- ODP.NETクライアントを有効化してFAN高可用性イベントを受信する方法
- ODP.NETクライアントを有効化してFANロード・バランシング・アドバイザのイベントを受信する方法
- トランザクション・ガードを使用するためのODP.NETクライアントの構成

Oracle統合クライアントとFANの概要

FANの全体的な目的は、アプリケーションのエンドツーエンドの完全自動リカバリ、および実際のトランザクション・パフォーマンスに基づいたロード・バランシングを可能にすることです。

アプリケーションは、FAN高可用性(HA)イベントを使用して、失敗を高速に検出し、失敗後の接続プールを均等に分散し、失敗したコンポーネントが修復されると接続を再度分散します。

ロード・バランシング・アドバイス・ヘルプ接続プールを持つFANイベントは、最適なサービスを提供する使用可能なインスタンスに接続を一貫して配信します。FAN HAは、JDBC-thinドライバおよびOCIドライバと統合されています。FAN HAおよびFANロード・バランシングの両方が、JDBC Universal Connection Pool (および非推奨の暗黙的な接続キャッシュ)、OCIセッション・プール、ODP.NET接続プールおよびOracle WebLogic Server Active GridLink for Oracle RACと統合されています。

FANとの統合によって、Oracle統合クライアントはOracle RACクラスタの最新のステータスをより迅速に認識します。これによって、クライアント接続が使用できなくなったインスタンスやサービスを待機したり接続試行することがなくなります。インスタンスが起動すると、Oracle RACでは、最近起動されたインスタンスへの接続を接続プールで作成し、このインスタンスで提供される追加のリソースを接続プールで使用できるように、FANを使用して接続プールに通知します。

FANと統合されたOracle クライアント・ドライバでは次のことが適用されます。

- 終了した接続を削除すると同時に、サービスがインスタンスでDOWNとして宣言され、ノードも同時にDOWNとして宣言されます。
- サービスが再起動を繰り返し試行する間クライアントを待機させるかわりに、NOT RESTARTING状態がOracle Databaseで検出されるとすぐにエラーをクライアントにレポートします。

FANと統合されたOracle接続プールでは、次の処理を実行できます。

- サービスの起動時に、Oracle RACのすべてのインスタンス間で接続を均等に分散します。この方法は、接続プールで定義されているセッションを、サービスがサポートされている最初のOracle RACインスタンスに割り当てるよりも有効です。
- ロード・バランシング・アドバイザのイベントを使用して、実行時の作業要求を均等に分散します。

クライアント・ドライバや接続プールとFANを使用する場合、FANイベントをクライアントに配信するようにOracle Notification Serviceを適切に構成する必要があります。また、ロード・バランシングの場合は、接続プールで使用されるサービスを提供するすべてのインスタンスへのデータベース接続のロード・バランシングを構成する必要があります。Oracle Net Servicesでクライアント側とサーバー側のロード・バランシングを構成することをお薦めします。Oracle DBCAを使用してデータベースを作成すると、デフォルトで、クライアント側とサーバー側の両方のロード・バランシングが構成されます。

関連項目

- 接続ロード・バランシング
- 高速アプリケーション通知

JDBC-Thinクライアントでの高速接続フェイルオーバーの有効化

Universal Connection PoolおよびOracle WebLogic Server Active GridLink for Oracle RACで高速接続フェイルオーバー(FCF)を有効化すると、FAN HAおよびロード・バランシング・アドバイザ・イベントを使用できるようになります。

Universal Connection PoolでFANを使用する場合、アプリケーションは、JDBC OCIクライアントまたはJDBC Thinクライアントのどちらの JDBC開発環境も使用できます。Java Database Connectivity Oracle Call Interface (JDBC/OCI)ドライバ接続プール機能は、JDBC-thinクライアントの一部です。この機能は、OracleOCIConnectionPoolク

ラスによって提供されます。

JDBC-thinクライアント用のFCFを有効にするには、最初のgetConnection()リクエストを行う前に、oracle.jdbc.poolパッケージのOracleDataSourceクラスのメソッドsetFastConnectionFailoverEnabled(true)を呼び出します。JDBC-thinクライアント用のFCFを有効にすると、フェイルオーバー・プロパティは接続プール内のすべての接続に適用されます。JDBC-thinドライバまたはJDBC/OCIクライアントでFCFを有効にすると、接続プールですべてのFANイベントを受信して、これらのイベントを処理できます。

JDBCアプリケーションの開発者は、Oracle Database 11gリリース2 (11.2)で導入された一連のAPIを使用して、プログラムでFANと統合できます。Oracle RAC FAN APIを使用すると、Oracle RACによって送信されるFANイベント通知の、アプリケーション・コードによる受信と応答が次の方法で可能になります。

- Oracle RACサービスの停止イベント、サービスの起動イベントおよびノードの停止イベントのリスニング
- ロード・バランシング・アドバイザ・イベントのリスニングと、それに対する応答

関連項目

- 『Oracle Database JDBC開発者ガイド』

JDBC-Thinクライアント用のOracle Notification Service

FCFは、Oracle Notification Serviceを利用して、接続プールとOracle RACデータベース間でデータベース・イベントを伝播します。実行時、接続プールは、Oracle Notification Service環境を設定できる必要があります。Oracle Notification Service(ons.jar)は、Oracleクライアント・ソフトウェアの一部として含まれています。Oracle Notification Serviceは、リモート構成またはクライアント側のOracle Notification Serviceデーモン構成のいずれかを使用して構成できます。リモートOracle Notification Serviceサブスクリプションを使用すると、次のメリットがあります。

- すべてのJava中間層ソフトウェアがサポートされます。
- クライアント・システムではOracle Notification Serviceデーモンが不要なため、このプロセスを管理する必要はありません。
- データ ソース・プロパティを使用して、構成作業を簡単に行うことができます。

JDBC/OCIおよびJDBC Thinドライバ・クライアント用のFCFの構成

Universal Connection Poolまたは暗黙接続キャッシュ用のFCFを有効にできます。

暗黙接続キャッシュは非推奨であるため、Universal Connection PoolをJavaに使用することをお薦めします。Oracle WebLogic Server Active GridLink for Oracle RACを使用することもできます。

この項では、JDBC用のFCFを有効にする方法について説明します。JDBC/OCIクライアントでFCFを有効にしている場合は、Oracle Database 11gリリース2 (11.2)で使用されているOCI クライアント用にFANを有効化する方法(サービスでnotificationをTRUEに設定)は使用せず、クライアントまたはサーバーのいずれにもTAFを構成しないでください。アプリケーション・コンティニュイティおよびトランザクション・ガードを構成することもできます。

FCFを有効にするには、次の手順で説明するように、最初にUniversal Connection Poolを有効にする必要があります。

1. 接続プールを作成し、setFastConnectionFailoverEnabled(true)を設定します。

次の例では、接続プールを作成し、FCFを有効にします。この例を使用する場合、ucp.jarライブラリはアプリケーションのCLASSPATHに含まれる必要があります。

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setFastConnectionFailoverEnabled(true);
```

2. Oracle Notification Serviceリモート・サブスクリプションに使用するポートを特定します。

   次の例に示すように、Oracle Clusterwareを実行している各ノードで、次のコマンドを使用してOracle Notification Service構成を表示します。

   ```
   srvctl config nodeapps -onsonly
   ```

   このコマンドの出力では、Oracle Notification Service用に構成されているローカル・ポートおよびリモート・ポートがリストされます。

   ノート:

   Oracle Notification Service 構成は、Oracle Clusterware のインストール時に自動的に完了しているはずです。

3. リモートOracle Notification Serviceサブスクリプションを構成します。

   ユニバーサル接続プールを使用する場合、アプリケーションはOracleDataSourceインスタンスのsetONSConfigurationをコールして、使用するノード番号とポート番号を指定します。次の例に示すように、各ノードで使用されるポート番号は、ステップ2の各ノードで表示されるリモート・ポートと同じです。この例を使用する場合、ons.jarライブラリはアプリケーションのCLASSPATHに含まれる必要があります。

   ```
   pds.setONSConfiguration("nodes=racnode1:6200,racnode2:6200");
   ```

   リモートOracle Notification Service構成を使用するアプリケーションでは、アプリケーションを起動する前に、oracle.ons.oraclehomeシステム・プロパティにORACLE_HOMEの場所を設定する必要があります。たとえば:

   ```
   java -Doracle.ons.oraclehome=$ORACLE_HOME ...
   ```

4. 接続URLを構成します。

   FCFを使用する場合、コネクション・ファクトリの接続URLはサービス名構文を使用する必要があります。サービス名は、接続プールをサービスにマップするために使用されます。次の例では、接続URLの構成を示しています。

   ```
   pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
   pds.setURL("jdbc:oracle:thin@//SCAN_name:service_name");...
   ```

関連項目

- 『Oracle Database JDBC開発者ガイド』
- Oracle Universal Connection Pool開発者ガイド

JDBCクライアントでのランタイム接続ロード・バランシングの有効化

実行時接続ロード・バランシングには、Oracle JDBCドライバおよびOracle RACデータベースを使用する必要があります。

Oracle JDBC Universal Connection PoolおよびOracle WebLogic Server Active GridLink for Oracle RAC

では、Oracle RACデータベースによって提供されるロード・バランシング機能が利用されます。

ロード・バランシング・アドバイザの情報を利用するために、Universal Connection PoolおよびOracle WebLogic Server Active GridLink for Oracle RACが統合されています。Oracle Database 11g リリース11.1.0.7.0では、JDBCのユニバーサル接続プールが導入されました。そのため、Oracle RACデータベースで使用するためにOracle Database 10g リリース1で導入された既存のJDBC接続プール(暗黙接続キャッシュ)が非推奨になりました。Oracle Database 12cに加えて、Oracle Database 10gまたはOracle Database 11gでもUniversal Connection Poolを使用できます。

実行時接続ロード・バランシングには、FCFが有効であり、適切に構成されていることが必要です。また、Oracle RACロード・バランシング・アドバイザは、接続プールで使用されるサービスごとにサービス・レベルの目標で構成される必要があります。接続時ロード・バランシングの目標は、SHORTに設定される必要があります。たとえば:

```
srvctl modify service -db db_unique_name -service service_name
   -rlbgoal SERVICE_TIME -clbgoal SHORT
```

関連項目

- JDBC/OCIおよびJDBC Thinドライバ・クライアント用のFCFの構成
- Oracle Universal Connection Pool開発者ガイド

Javaのアプリケーション・コンティニュイティのためのJDBC-Thinクライアントの構成

リプレイ・データ・ソース(oracle.jdbc.replay.OracleDataSource)は、アプリケーション・コンティニュイティがJavaで必要とするJDBC-thinデータ・ソースです。

このデータ・ソースは、新しい物理JDBC接続をUniversal Connection PoolおよびOracle WebLogic Server Active GridLink for Oracle RACデータ・ソースの両方に生成するコネクション・ファクトリとして機能します。JDBCリプレイ・ドライバは、Oracle Database 12cとのクライアント対話中のコールの履歴を、Oracle Databaseと協力して保持します。データベース・サービスの欠落で生じるセッションの停止(計画済または計画外)に続いて、データベースの指示のもとで、JDBCリプレイ・ドライバは、非トランザクションおよびトランザクションのデータベース・セッション状態の再構築を試行し、これによって停止は遅れた実行として示されます。

Javaのアプリケーション・コンティニュイティおよびJDBCリプレイ・ドライバを使用するには、Oracle Database 12cクライアントを使用してOracle Database 12cデータベースに接続する必要があります。Javaのアプリケーション・コンティニュイティは次の構成でサポートされています:

- Oracle JDBCリプレイ・データ・ソースを使用し、Universal Connection PoolまたはOracle WebLogic Server Active GridLink(典型的なサード・パーティ製JDBCベースの接続プール)を使用しないJDBCアプリケーション
- Universal Connection Poolデータ・ソースを使用するJDBCアプリケーション(Universal Connection Poolデータ・ソースを使用するように構成されたスタンドアロンまたはサード・パーティ製のアプリケーション・サーバー)
- Oracle WebLogic Server Active GridLinkのみを使用して、Universal Connection Poolデータ・ソース(典型的なOracle WebLogic Server J2EEケース)を使用しないJDBCアプリケーション

JDBCリプレイ・ドライバを使用するようにJDBC-thinクライアントを構成するには:

1. リプレイ用に認証されているアプリケーションを使用してください。
2. アプリケーションで使用するサービスがまだ存在しない場合は、SRVCTLを使用してそのサービスを作成します。このサービスで-failovertypeパラメータをTRANSACTIONに設定し、-commit_outcomeパラメータをTRUEに設定します。
3. 次の例に示すとおり、PoolDataSourceオブジェクトを使用して接続要素を構成します。

```
PoolDataSource rds = PoolDataSourceFactory.getPoolDataSource();
rds.setConnnectionPoolName("replayExample");
rds.setONSConfiguration("nodes=racnode1:4200,racnode2:4200");
rds.setFastConnectionFailoverEnabled(true);
rds.setConnectionFactoryClassName("oracle.jdbc.replay.OracleDataSourceImpl");
Connection conn = rds.getConnection();
```

4. データベースへの接続時に、サービスを提供するすべてのインスタンスにアクセス可能なURLを使用します。

関連項目

- アプリケーション・コンティニュイティについて
- アプリケーション・コンティニュイティおよびトランザクション・ガードのサービスの作成
- JDBC/OCIおよびJDBC Thinドライバ・クライアント用のFCFの構成
- 『Oracle Database JDBC開発者ガイド』
- Oracle Universal Connection Pool開発者ガイド

関連項目:

アプリケーション・コンティニュイティを有効化しないトランザクション・ガードの構成の詳細は、*Oracle Database JDBC*開発者ガイドを参照

トランザクション・ガード用のJDBC-Thinクライアントの構成

トランザクション・ガードは、計画および計画外の停止の場合に、処理を1回以下にするためにアプリケーションで使用されるプロトコルおよび汎用ツールを提供します。

アプリケーションでは論理トランザクションIDを使用して、停止に続くデータベース・セッション内でオープンになっている最終トランザクションの結果が判断されます。トランザクション・ガードを使用しないと、停止の後にエンド・ユーザーまたはアプリケーションが操作を再試行しようとして、トランザクションが重複してコミットされたり、順序が不適切にトランザクションがコミットされることで、論理破損が発生する可能性があります。

関連項目

- 『Oracle Database JDBC開発者ガイド』
- Oracle Database開発ガイド
- Oracle Call Interfaceプログラマーズ・ガイド

OCIクライアントでの高速接続フェイルオーバーの有効化

OCIクライアントは、Oracle RAC高可用性FANイベントの通知を受信するように登録し、イベント発生時に応答することに

よって、FCFを有効にできます。FCFを使用すると、OCIアプリケーションでのセッション・フェイルオーバー応答時間が向上し、接続プールおよびセッション・プールから機能していないインスタンスへの接続も削除されます。FCFはOCIアプリケーションで使用でき、このアプリケーションはTAF、OCIドライバ(独自の接続プールを含む)、OCI接続プールおよびOCIセッション・プールも使用します。FANは、高可用性およびロード・バランシング・イベントのためにOracle Notification Serviceでポストされます。

FCFを使用するには、FANが有効なサービスを使用する必要があります。FANは、Oracle Notification Service経由で公開されます。クライアント・アプリケーションには、イベント発生時に使用するコールバックを登録することもできます。これによって、接続障害が検出されるまでの時間が短縮されます。

DOWNイベントの処理中に、OCIでは次の処理が実行されます。

- クライアントで影響を受ける接続が終了し、エラーが戻されます。
- OCI接続プールおよびOCIセッション・プールから接続を削除します。OCIセッション・プールでは、各セッションが接続プールの物理的な接続とマッピングされています。1つの接続に複数のセッションが存在する場合もあります。
- TAFが構成されている場合、接続をフェイルオーバーします。TAFが構成されていない場合、接続先のインスタンスに障害が発生しても、クライアントはエラーを受信するだけです。

アプリケーションでTAFを使用している場合は、SRVCTLまたはOracle Enterprise Managerを使用して、サービスのTAFプロパティを有効にする必要があります。構成済のサービスを使用して、Oracle RACデータベースに接続するようにOCIクライアント・アプリケーションを構成します。

ノート:

OCI は UP イベントを管理しません。

OCIクライアント用のFCFの構成

OCIアプリケーションは、Oracle RACインスタンスに接続して、HAイベント通知を有効にする必要があります。さらに、これらのアプリケーションは、次のステップを実行してOCIクライアント用のFCFを構成する必要があります。

1. 次の例に示すとおり、FAN、接続時ロード・バランシングおよびランタイム接続ロード・バランシングを有効にするように、OCI接続プールのサービスを構成します。

```
$ srvctl modify service -db crm -service ociapp.example.com -notification TRUE
```

2. アプリケーションをスレッド・ライブラリにリンクします。
3. スレッド・ライブラリにリンクした後、アプリケーションは、FANイベントが発生すると常に起動されるコールバックを登録できます。

関連項目

- Oracle Database Net Services管理者ガイド
- Oracle Call Interfaceプログラマーズ・ガイド

OCIクライアントでのランタイム接続ロード・バランシングの有効化

Oracle Database 12cでは、OCIセッション・プールによって、動的に管理される事前作成済のデータベース・セッションのセッ

トをアプリケーションの複数スレッドで使用できます。

接続プーリングではプール要素は接続ですが、セッション・プーリングではプール要素はセッションとなります。Oracle Databaseでは、セッション・プール内のセッションを継続的に再利用して、インスタンスへのほぼ永続的なチャネルを形成することによって、アプリケーションでセッションが必要になるたびにセッションを作成してクローズするオーバーヘッドを削減できます。

ランタイム接続ロード・バランシングは、デフォルトでOracle Database 11gリリース11.1以上、Oracle Database 10gリリース10.2以上のサーバーとのクライアント通信で有効になっています。Oracle RAC環境の場合、アプリケーションのセッション要求を均等に分散させるために、セッション・プールは、高速アプリケーション通知(FAN)イベントを介してOracle RACロード・バランシング・アドバイザ脚注1から受信されるサービス・メトリックを使用します。セッション・プールに入ってくる作業要求は、現在のサービス・パフォーマンスを使用して、サービスを提供しているOracle RACのインスタンス全体で分散できます。

OCIクライアントを構成してロード・バランシング・アドバイザのFANイベントを受信する方法

Oracle RAC環境の場合、アプリケーションのセッション要求を均等に分散させるために、セッション・プールは、高速アプリケーション通知(FAN)イベントを介してOracle RACロード・バランシング・アドバイザから受信されるサービス・メトリックを使用します。次の例に示すとおり、アプリケーションがサービス時間に基づいてサービス・メトリックを受信できるようにするには、FAN、ロード・バランシング・アドバイザの目標(-rlbgoalパラメータ)および接続ロード・バランシングの目標(-clbgoalパラメータ)を、セッション・プールで使用されるサービスに構成していることを確認します。

```
$ srvctl modify service -db crm -service ociapp.example.com -rlbgoal SERVICE_TIME
  -clbgoal SHORT -notification TRUE
```

関連項目

- Oracle Call Interfaceプログラマーズ・ガイド

トランザクション・ガードを使用するためのOCIクライアントの構成

OCIは、FANメッセージおよびトランザクション・ガードをサポートしています。FANは、ノード、データベース、インスタンス、サービスおよびパブリック・ネットワーク・レベルでの停止をOCIベースのアプリケーションに迅速に通知するように設計されています。

障害の通知があると、アプリケーションはトランザクション・ガードを利用して、最後の処理中のトランザクションの結果を確実に判別できます。

トランザクション・ガードによって、ユーザーが不満を抱く不明なエラー、カスタマ・サポート・コールおよび機会損失のコストを削減します。トランザクション・ガードは、既知の結果に対する自社製のソリューションと比べて、より安全でパフォーマンスが良く、オーバーヘッドがより少なくなっています。

関連項目

- 高速アプリケーション通知
- Oracle RACのクライアントの有効化
- Oracle Call Interfaceプログラマーズ・ガイド

ODP.NETクライアントを有効化してFAN高可用性イベントを受信する方法

ODP.NETの接続プールでは、ノード、サービスおよびサービス・メンバーが停止したことを示すFAN HA通知をサブスクライブできます。

DOWNイベントが発生すると、そのインスタンスに送られる接続プール内のセッションはOracle Databaseによって削除され、ODP.NETは無効になった接続を事前対応的に削除します。無効な接続が削除されたことで、接続合計数がMin Pool Sizeパラメータの値を下回った場合、ODP.NETは、既存のOracle RACインスタンスへの追加の接続を確立します。

Oracle Database 12c以上に接続する場合、ODP.NETは、Advanced Queuingではなく、Oracle Notification Serviceを使用します。

FAN高可用性イベントをサブスクライブすることによって、ODP.NET接続プールの高速接続フェイルオーバーを有効にします。高速接続フェイルオーバーを有効にするには、次の例に示すとおり、HA Events=trueおよびpooling=true (デフォルト値)を接続文字列に含めますが、ここで、user_nameはデータベース・ユーザーの名前、passwordはそのユーザーのパスワードです。

```
con.ConnectionString =
   "User Id=user_name;Password=password;Data Source=odpnet;" +
   "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
   "HA Events=true;Incr Pool Size=5;Decr Pool Size=2";
```

関連項目

- Oracle Data Provider for .NETの開発者ガイドfor Microsoft Windows
- 高速アプリケーション通知

ODP.NETクライアントを有効化してFANロード・バランシング・アドバイザのイベントを受信する方法

Oracle Database 12c以上に接続する場合、ODP.NETは、Advanced Queuingではなく、Oracle Notification Serviceを使用します。

ODP.NETクライアントまたはアプリケーションを有効化して、FANロード・バランシング・アドバイザのイベントを受信するには、次の手順を実行します。

1. 次の例に示すとおり、SRVCTLを使用してOracle Notification Serviceの通知を有効にし、実行時ロード・バランシングの目標を設定します。

   ```
   $ srvctl modify service -db crm -service odpapp.example.com
     -notification TRUE -clbgoal LONG -rlbgoal SERVICE_TIME
   ```

2. Oracle Notification Service (ONS)をFANイベント(実行時ロード・バランシング・アドバイスなど)用に構成します。
3. ConnectionStringのロード・バランシング属性にTRUEを設定して、ODP.NET接続プールでロード・バランシング・イベントを利用するように構成します(デフォルトはFALSE)。この処理は、接続時に実行できます。この処理は、接続プールを使用している場合、またはプーリング属性がTRUE(デフォルト)設定されている場合にのみ実行できます。

   次の例では、ロード・バランシングが有効になるようにConnectionStringを構成する方法を示します。user_nameはユーザー名、passwordはパスワードです。

```
con.ConnectionString =
  "User Id=user_name;Password=password;Data Source=odpapp;" +
  "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
  "Load Balancing=true;Incr Pool Size=5;Decr Pool Size=2";
```

ノート:

ODP.NET では、ノード起動時(UP イベント)の接続の再分散はサポートしていません。ただし、サーバー側でフェイルオーバーが有効になっている場合は、ODP.NET で、新しく使用可能になったインスタンスに接続を移行できます。

関連項目

- srvctl modify service
- Oracle Data Provider for .NETの開発者ガイドfor Microsoft Windows
- 高速アプリケーション通知

トランザクション・ガードを使用するためのODP.NETクライアントの構成

ODP.NETは、FANメッセージおよびトランザクション・ガードをサポートしています。FANは、ノード、データベース、インスタンス、サービスおよびパブリック・ネットワーク・レベルでの停止をODP.NETベースのアプリケーションに迅速に通知するように設計されています。

障害の通知があると、アプリケーションはトランザクション・ガードを利用して、最後の処理中のトランザクションの結果を確実に判別できます。

トランザクション・ガードによって、ユーザーが不満を抱く不明なエラー、カスタマ・サポート・コールおよび機会損失のコストを削減します。トランザクション・ガードは、既知の結果に対する自社製のソリューションと比べて、より安全でパフォーマンスが良く、オーバーヘッドがより少なくなっています。

関連項目

- 高速アプリケーション通知
- サービスの管理
- アプリケーション・コンティニュイティおよびトランザクション・ガードのサービスの作成
- Oracle Data Provider for .NETの開発者ガイドfor Microsoft Windows

Oracle RACの分散トランザクション処理

X/Open Distributed Transaction Processing(DTP)アーキテクチャは、複数のアプリケーション・プログラム(AP)が複数の異なるリソース・マネージャ(RM)から提供されるリソースを共有できるようにするための、標準のアーキテクチャまたはインタフェースを定義しています。APとRM間の作業を調整し、グローバル・トランザクションを実現します。

次の項では、Oracle RACがグローバル(XA)・トランザクションおよびDTP処理をサポートする方法について説明します。

- XAトランザクションとOracle RACの概要
- XAトランザクションのためのグローバル・トランザクションとXAアフィニティの使用

- Oracle RACのXAトランザクションによるサービスの使用
- XAアプリケーションのサービスの構成
- 管理者管理データベースのサービスの再配置

XAトランザクションとOracle RACの概要

デフォルトでグローバル(XA)・トランザクションは、Oracle RACインスタンスにまたがることができ、Oracle XAライブラリを使用する任意のアプリケーションが、Oracle RAC環境を十分に利用してアプリケーションの可用性およびスケーラビリティを向上させるようにすることができます。

GTXnバックグラウンド・プロセスは、Oracle RAC環境でXAトランザクションをサポートしています。GLOBAL_TXN_PROCESSES初期化パラメータ(デフォルトで1に設定)は、各Oracle RACインスタンスのGTXnバックグラウンド・プロセスの初期数を指定します。クラスタ全体でこのパラメータのデフォルト値を使用し、複数のOracle RACインスタンス間にわたる分散トランザクションを可能にします。デフォルト値の使用により、Oracle RACインスタンス全体にわたって実行される作業単位は、リソースを共有し、単一のトランザクションとして機能します(つまり、作業単位は密結合となります)。また、クラスタ内の任意のノードへの2フェーズ・コミット要求の送信も可能となります。

Oracle RAC 11gリリース1 (11.1)より前では、Oracle RACで密結合を実現する方法として分散トランザクション処理(DTP)サービス、つまりカーディナリティ(1)によって、ロード・バランシングが有効かどうかに関係なく、すべての密結合ブランチが確実に同じインスタンスに配置されるサービスを使用していました。XAアプリケーションが同じトランザクション・ブランチで一時停止および再開を使用せず、かつ、ブランチにまたがるセーブポイントを発行しない場合、密結合されたXAトランザクションは、Oracle RACデータベースにデプロイするための特別なタイプのシングルトン・サービスを必要としません。トランザクション・ブランチが一時停止されたか再開されたかをアプリケーションが判断できない場合は、アプリケーションは、DTPサービスを、または、できればXAアフィニティを引き続き使用する必要があります。

同じXAブランチを一時停止または再開する場合、あるいはブランチにわたってセーブポイントを使用する場合、XAアフィニティ(同じXAトランザクションのすべてのブランチを同じOracle RACインスタンスに配置すること)が要件になります。様々なトランザクションのバランスを取ることができるため、はるかに優れたパフォーマンスももたらされます。XAアフィニティは、Oracle WebLogic Server Active GridLink for Oracle RAC、JDBC Universal Connection PoolおよびOracle Tuxedoで使用できます。XAアフィニティは、RedHat JBoss、IBM WebSphereおよびIBM Libertyに対しても標準です。

ノート:

1 つの優先インスタンスと多くの使用可能なインスタンスでサービスを使用する場合、XA でのトランザクション処理モニターが最適に動作します。Oracle Database 11g リリース 1 (11.1)以降の DTP 設定の使用はお薦めしません。

関連項目

- XAトランザクションのためのグローバル・トランザクションとXAアフィニティの使用
- Oracle Databaseリファレンス

XAトランザクションのためのグローバル・トランザクションとXAアフィニティの使用

Oracle RACの分散トランザクション処理(DTP)を使用してアプリケーションのパフォーマンスを向上させるには、XAアフィニティを利用します。

XAアフィニティを使用すると、分散トランザクションのすべてのブランチをクラスタ内のシングル・インスタンスに割り当てることができます。XAアフィニティを実装するために、WebLogic Serverやユニバーサル接続プールなど、XAアフィニティを提供するアプリケーション・サーバーを使用できます。アプリケーション・サーバーにXAアフィニティがない場合は、Oracle RAC全体でシングルトン・サービスを使用することもできます。

Oracle RACデータベースの複数の接続にわたりロード・バランスを実行するアプリケーション・サーバー層の接続プールでは、XAアフィニティを使用して、1つのグローバル分散トランザクションのすべての密結合ブランチが、1つのOracle RACインスタンスのみで実行されるようにします。XAアフィニティを備えた接続プールを使用すると、XAを使用するサービスをOracle RACに広げられるようになります。これは、X/Open分散トランザクション処理やMicrosoft分散トランザクション・コーディネータなどのプロトコルを使用した分散トランザクション環境にも当てはまります。

分散トランザクションのパフォーマンスを向上させるには、優先インスタンスが1つのサービスを使用します。シングルトン・サービスは、Oracle RACデータベースの1つのOracle RACインスタンスで1つずつ実行されます。このサービスもメンテナンス目的の排出が可能であるため、従来のDTPサービスよりも優れた高可用性の特性があります。クラスタ全体でロード・バランスを実行するには、1つまたは2つの大規模アプリケーション・サーバーを使用するよりも、小規模アプリケーション・サーバーのグループをいくつか用意して、各グループ内でトランザクションを単一または一連のサービスに割り当てる方が効果的です。シングルトン・サービスを使用すると、サービスを通じて実行されるグローバル分散トランザクションは、単一のOracle RACインスタンスで実行する専用の密結合ブランチを持ちます。これには、次のような利点があります。

- 密結合ブランチで相互に行った変更が必要である場合に、1つのOracle RACインスタンス内で変更をローカルに参照できます。
- サービスの再配置とフェイルオーバーは、グローバル・トランザクションを使用することで完全にサポートされます。
- Oracle Databaseでは、Oracle RACインスタンスより多くのシングルトン・サービスを使用することによって、Oracle RACデータベースのすべてのインスタンスのサービスで負荷を均等に分散できます。

ノート:

Oracle Database 11g リリース 1 (11.1)以降の DTP 設定の使用はお薦めしません。

Oracle RACのXAトランザクションによるサービスの使用

Oracle RACでXAを使用するほとんどのアプリケーションは、接続プールまたはトランザクション処理モニターによって提供されるXAアフィニティで、均一サービス(またはすべての優先サービス)を使用できます。

XAアフィニティを提供するために、アプリケーションでもシングルトン・サービスを使用できます。

シングルトン・サービスを使用しているときに、クラスタ内のすべてのインスタンスを利用するには、分散トランザクションをホストするOracle RACインスタンスごとに1つ以上のシングルトン・サービスを作成します。各アプリケーション・サーバーの別々のサービスを選択して、Oracle RACデータベース・インスタンス間でワークロードを均等に分散します。1つの分散トランザクションのすべての

ブランチが1つのインスタンスで実行されるため、複数のシングルトン・サービスを介して、多数の分散トランザクション処理(DTP)トランザクションの負荷を均等に分散するために、すべてのインスタンスを利用できるようになり、その結果、アプリケーションのスループットを最大限にできます。

クラスタ・データベースのノードを追加または削除した場合は、最適なパフォーマンス・レベルを維持するために、サービスの確認と再配置が必要になることがあります。シングルトン・サービスを使用すると、現在の作業を完了できます。DTPサービスを使用すると、現在の作業は終了します。

DTPサービスは、同一ブランチを一時停止して再開するXAアプリケーションにのみ使用する必要があります。DTPを使用しているときには、シングルトンの場合と同じアプローチを適用しますが、サービスの再配置時に作業を排出できなくなります。

XAアプリケーションのサービスの構成

分散トランザクション処理用の分散トランザクション処理(DTP)サービスを作成するには、次のステップを実行します。

1. Oracle Enterprise ManagerまたはSRVCTLを使用して単一のサービスを作成します。

   管理者管理データベースの場合は、優先インスタンスとして1つのインスタンスのみを定義します。必要な数の使用可能なインスタンスを指定できます。たとえば:

   ```
   $ srvctl add service -db crm -service xa_01.example.com -preferred RAC01
     -available RAC02,RAC03
   ```

   ポリシー管理データベースの場合は、使用するサーバー・プールを指定して、サービスのカーディナリティをSINGLETONに設定します。たとえば:

   ```
   $ srvctl add service -db crm -service xa_01.example.com -serverpool mypool
     -cardinality SINGLETON
   ```

2. サービスのDTPパラメータ(-dtp)をTRUEに設定します(デフォルト値はFALSEです)。Oracle Enterprise ManagerまたはSRVCTLを使用して、単一のサービスのDTPプロパティを変更できます。次の例では、SRVCTLを使用してxa_01.example.comサービスを変更する方法を示しています。

   ```
   $ srvctl modify service -db crm -service xa_01.example.com -dtp TRUE
   ```

ノート:

アプリケーションで DTP サービスを必要する場合は、-dtp パラメータを使用してください。それ以外の場合は、-dtp パラメータを指定していない前述の例を使用してください。

関連項目

- srvctl add service
- srvctl modify service

管理者管理データベースのサービスの再配置

Oracle Real Application Clusters 11g リリース1 (11.1)以降では、グローバル・トランザクションとXAアフィニティが分散トランザクション処理(DTP)サービスの必要性に取ってかわります。

XAデプロイメントのほとんどは、ロード・バランシングと柔軟性が向上するように、DTP属性ではなくグローバル・トランザクションとXAアフィニティを使用するようになっています。

サービスが他のインスタンスに移行したら、使用可能なすべてのハードウェアで均等に負荷を再分散させるために、優先インスタンスにサービスを強制的に再配置する必要がある場合があります。GV$ACTIVE_SERVICESビューのデータを使用して、DTPサービスを再配置する必要があるかどうかを判断できます。

Oracle RACシャーディング

Oracle RACシャーディングは、表パーティションとOracle RACインスタンスの間にアフィニティを作成し、対応するパーティションを論理的に保持するインスタンスにパーティション化キーを指定するデータベース・リクエストをルーティングします。

Oracleでは、行のアフィニティをインスタンスで作成するデータベースで行の非結合行のサブセットに対して、各インスタンスがリクエストを常に取得するように、Oracle RACインスタンスにデータベース・リクエストをルーティングします。アフィニティにより、キャッシュ・ローカリティが改善され、ノード間の同期およびブロックのpingが低減されるため、Oracle RACの高いパフォーマンスおよびスケーラビリティが実現されます。

Oracle RACアフィニティのシャーディングでは、クライアントとサーバー側のサポートを使用して、Oracle Databaseのシャーディングに含まれているキーベースのルーティングを行います。Oracle接続プール(ユニバーサル接続プール、OCIなど)でのシャーディングのサポート用に実装されているものと同じAPIを使用して、データベースのシャーディング・キーを提供するアプリケーションは、シャーディングに対する実行と同じ方法で、キーベース・ルーティングを使用します。これにより、Oracle RACアフィニティが有効化されます。

シャーディング・キーの提供に必要なアプリケーションの変更は、アプリケーションのすべてのモジュールに影響を与える必要がありません。変更は、頻繁に処理されないデータベース・リクエストにのみ適用できます。接続文字列にシャーディング・キーを提供しないリクエストは、ロード・バランシング・ポリシーに基づいてルーティングされます。インスタンスに対してデータ・オブジェクトの明示的な所有権割当てを行うため、キーのないリクエストはデータ・アフィニティに悪影響を及ぼしません。

ノート:

Oracle では、パーティション化された表にのみ Oracle RAC アフィニティがサポートされます。データベース・スキーマを変更せずに、サポートされている方法を使用して表をパーティション化すると、この機能を有効にして ALTER SYSTEM ENABLE AFFINITY コマンドを実行できます。

アフィニティが有効なルーティングを利用するためにアプリケーションに変更を加える場合、独立した複数のデータベース間にデータが分散されているときは、シャーディングを利用することもできます。最大のスケーラビリティおよび障害の分離が必要な場合、後で分散シャーディングに移行できます。

関連項目

- Oracle Database SQL言語リファレンス
- Oracle Database Net Services管理者ガイド
- Oracleシャーディングの使用
- 『Oracle Database JDBC開発者ガイド』
- Oracle Call Interfaceプログラマーズ・ガイド

自動ワークロード・リポジトリ

自動ワークロード・リポジトリ(AWR)は、データベースのパフォーマンス統計値を収集、処理および保持します。

収集されたデータは、レポートとビューに表示できます。データベースでサービスを使用すると、AWRはサービス・レベルでメトリックを追跡します。

メトリックは、時間、トランザクション、データベース・コールなど、様々な単位に対して測定できます。たとえば、データベース・コール/秒はメトリックです。サーバーによって生成されるアラートは、ユーザー指定のしきい値を超えたり、必要なしきい値に達しない場合、これらのメトリックに基づいて発行されます。その後で、データベースまたはシステム管理者は、次のように応答できます。

- Oracle Databaseリソース・マネージャを使用して、あるサービスのサービス・レベルの優先順位を他のサービスよりも高くする
- オーバーロード状態になったプロセスを停止する
- サービス・レベル要件を変更する
- サービス品質の変更に応答するためにリカバリ例を実施する

AWRメトリックおよびパフォーマンス・アラートを使用すると、サービス・レベルが変更されても、継続的なサービスの可用性を維持できます。また、データベース・サービスによって提供されるサービスの品質を測定できます。

AWRによって、Oracle Clusterwareのワークロード管理フレームワークおよびデータベース・リソース・マネージャにおけるパフォーマンス・データ表現の永続性とグローバル性が保証されます。この情報によって、Oracle Databaseはサービス別にジョブ・クラスをスケジュールしたり、コンシューマ・グループに優先順位を割り当てることができます。必要に応じて、Oracle Enterprise ManagerまたはSRVCTLを使用して、手動でワークロードを再度均等に分散させることができます。一連のセッションを切断しても、サービスを続行させておくこともできます。

ノート:

DBMS_SERVICE パッケージを、Oracle RAC データベースが使用するサービスに使用することはお薦めしません。SRVCTL または Oracle Enterprise Manager を使用して Oracle RAC 用のデータベース・サービスを作成します。

関連項目

- Oracle Database 2日でパフォーマンス・チューニング・ガイド
- Oracle Databaseパフォーマンス・チューニング・ガイド
- Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス

自動ワークロード・リポジトリを使用したサービスのパフォーマンスの測定

サービスによってパフォーマンス・チューニングに新たな局面が加わります。サービスにより、ワークロードの視覚化と測定が可能になり、リソース使用量と待機時間はアプリケーションに起因すると考えられるようになるためです。

すべてのセッションが匿名で共有されている多くのシステムでは、セッションおよびSQLを使用したチューニングのかわりに、サービスおよびSQLを使用したチューニングを実行します。

AWRでは、データベースで実行されているすべてのサービスと作業の応答時間、スループット、リソース使用量および待機イベントの情報など、パフォーマンス統計が保持されます。また、サービスのメトリック、統計、待機イベント、待機クラスおよびSQLレベルのトレースも保持されます。さらに、オプションとして、特定の統計を監視するためのモジュールをアプリケーションで定義して、これらの統計をカスタマイズできます。また、このモジュール内に、重要なビジネス・トランザクションで特定の統計値に応答して実行されるアクションを定義することもできます。

モジュールおよびアクションの監視を有効にするには、DBMS_MONITOR PL/SQLパッケージを使用します。たとえば、erpサービスを使用する接続の場合、次のコマンドを使用すると、payrollモジュールのexceptions payアクションを監視できます。

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP',
   MODULE_NAME=> 'PAYROLL', ACTION_NAME => 'EXCEPTIONS PAY');
```

erpサービスを使用する接続の場合、次のコマンドを使用すると、payrollモジュールのすべてのアクションを監視できます。

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP',
   MODULE_NAME=> 'PAYROLL', ACTION_NAME => NULL);
```

アプリケーション・モジュールとアクションの監視が有効化されたことを確認するには、DBA_ENABLED_AGGREGATIONSビューを使用します。

サービスによる統計の収集およびトレースは、Oracle RACデータベース全体を対象とします。また、Oracle RACデータベースと非クラスタのOracle Databaseのどちらの場合も、インスタンスの再起動やサービスの再配置が行われても統計の集計は失われません。

サービス名、モジュール名およびアクション名は、V$SESSION、V$ACTIVE_SESSION_HISTORYおよびV$SQLビューで確認できます。コール回数およびパフォーマンス統計は、V$SERVICE_STATS、V$SERVICE_EVENT、V$SERVICE_WAIT_CLASS、V$SERVICEMETRICおよびV$SERVICEMETRIC_HISTORYで確認できます。重要なトランザクションで統計の収集を有効にしている場合、V$SERV_MOD_ACT_STATSビューを使用すると、各データベース・インスタンスのそれぞれのサービス名、モジュール名およびアクション名に対して、コール速度を確認できます。

SQL*Plusスクリプトの次のサンプルを実行すると、5秒間隔でサービス品質の統計が収集されます。これらのサービス品質の統計を使用して、サービスの品質を監視したり、作業を割り当てたり、サービスをOracle RAC全体のインスタンス間で均等に分散できます。

```
SET PAGESIZE 60 COLSEP '|' NUMWIDTH 8 LINESIZE 132 VERIFY OFF FEEDBACK OFF
COLUMN service_name FORMAT A20 TRUNCATED HEADING 'Service'
COLUMN begin_time HEADING 'Begin Time' FORMAT A10
COLUMN end_time HEADING 'End Time' FORMAT A10
COLUMN instance_name HEADING 'Instance' FORMAT A10
COLUMN service_time HEADING 'Service Time|mSec/Call' FORMAT 999999999
COLUMN throughput HEADING 'Calls/sec'FORMAT 99.99
BREAK ON service_name SKIP 1
SELECT
    service_name
  , TO_CHAR(begin_time, 'HH:MI:SS') begin_time
  , TO_CHAR(end_time, 'HH:MI:SS') end_time
  , instance_name
  , elapsedpercall  service_time
  ,  callspersec  throughput
FROM
```

```
    gv$instance i
  , gv$active_services s
  , gv$servicemetric m
WHERE s.inst_id = m.inst_id
  AND s.name_hash = m.service_name_hash
  AND i.inst_id = m.inst_id
  AND m.group_id = 10
ORDER BY
   service_name
 , i.inst_id
 , begin_time ;
```

自動ワークロード・リポジトリ・サービスのしきい値とアラート

サービス・レベルのしきい値を使用すると、実際のサービス・レベルとサービスの必須レベルを比較できます。これによって、満足するサービス・レベルが提供されているかどうかを認識できます。最終的な目標は、基準を満たしたサービス・レベルを提供する、予測可能なシステムを構成することです。最小限のリソース使用量で可能なかぎり速く実行することは必要ではなく、必要なのは、サービスの品質を満たすことです。

AWRを使用すると、コールの応答時間(ELAPSED_TIME_PER_CALL)およびコールのCPU時間(CPU_TIME_PER_CALL)の2つのパフォーマンスしきい値を、サービスごとに明示的に指定できます。応答時間のしきい値は、各サービスの各ユーザー・コールの経過時間が特定の値を超えないようにすることを示すもので、コールのCPU時間のしきい値は、各サービスの各コールによるCPUの使用時間が特定の値を超えないようにすることを示すものです。応答時間は、ユーザーのためのコールの実行に支障を与える可能性があるすべての遅延および障害を反映する基本的な測定単位です。また、応答時間では、Oracle RACデータベースのノード間における、ノード別の処理能力の差異も確認できます。

Oracle RACデータベースの各インスタンスに、これらのしきい値を設定する必要があります。経過時間およびCPU時間は、サーバー側のコールの経過時間を移動平均法で算出した時間です。AWRは経過時間およびCPU時間を監視し、パフォーマンスがしきい値を超えた場合には、AWRアラートを発行します。これらのアラートに対してOracle Enterprise Managerのジョブを使用してアクションをスケジュールすることも、アラート受信時にプログラムによってアクションが発生するようにアクションをスケジュールすることもできます。アラートが発行された場合は、ジョブの優先順位を変更したり、オーバーロード状態になったプロセスを停止したり、サービスを再配置、起動または停止して応答します。これによって、需要量に変動が発生した場合でも、サービスの可用性を維持できます。

この項には次のトピックが含まれます:

- サービスおよびしきい値のアラートの例
- サービス、モジュールおよびアクション監視の有効化

サービスおよびしきい値のアラートの例

この例では、payrollサービスのしきい値を確認する必要があります。この情報は、AWRレポートを使用して取得できます。システムが最適な状態で稼働しているときに、いくつかの連続した時間間隔で実行したレポートの結果を比較する必要があります。たとえば、payrollアプリケーションがアクセスするサーバーで、毎週木曜日に使用率がピークに達する午後1時から午後5時までの間にAWRレポートを実行するとします。AWRレポートには、payrollサービスを含む各サーバーのコールについて、応答時間

(経過データベース時間)およびCPUの使用時間(CPU時間)が含まれます。また、AWRレポートには、完了した作業のブレークダウンおよび応答時間に影響を与えている待機時間も記録されます。

DBMS_MONITORを使用して、payrollサービスのコールごとの経過時間に対する警告しきい値に0.5秒(500000マイクロ秒)を設定します。また、payrollサービスのコールごとの経過時間の重要な警告のしきい値を0.75秒(750000マイクロ秒)に設定します。

この例では、payrollサービスのしきい値を次のように追加します。

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
METRICS_ID => DBMS_SERVER_ALERT.ELAPSED_TIME_PER_CALL
, warning_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, warning_value => '500000'
, critical_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, critical_value => '750000'
, observation_period => 30
, consecutive_occurrences => 5
, instance_name => NULL
, object_type => DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE
, object_name => 'payroll');
```

次のSELECT文を使用すると、しきい値の構成がすべてのインスタンスに設定されていることを確認できます。

```
SELECT METRICS_NAME, INSTANCE_NAME, WARNING_VALUE, CRITICAL_VALUE,
OBSERVATION_PERIOD FROM dba_thresholds ;
```

サービス、モジュールおよびアクション監視の有効化

各サービス内の重要なモジュールおよびアクションに対するパフォーマンス・データのトレース機能を有効にできます。パフォーマンス統計は、V$SERV_MOD_ACT_STATSビューで参照できます。たとえば、次のように設定できます。

- ERPサービスで、payrollモジュール内のexceptions payアクションを監視します。
- ERPサービスで、payrollモジュール内のすべてのアクションを監視します。
- HOT_BATCHサービスで、postingモジュール内のすべてのアクションを監視します。

次のコマンドは、サービスのモジュールとアクションの監視を有効化する方法を示します。

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'erp', module_name=>
 'payroll', action_name => 'exceptions pay');
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'erp', module_name=>
 'payroll');
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'hot_batch',
module_name =>'posting');
```

サービス、モジュールおよびアクションの監視が有効化されいてることを確認するには、次のSELECT文を使用します。

```
COLUMN AGGREGATION_TYPE FORMAT A21 TRUNCATED HEADING 'AGGREGATION'
COLUMN PRIMARY_ID FORMAT A20 TRUNCATED HEADING 'SERVICE'
COLUMN QUALIFIER_ID1 FORMAT A20 TRUNCATED HEADING 'MODULE'
COLUMN QUALIFIER_ID2 FORMAT A20 TRUNCATED HEADING 'ACTION'
SELECT * FROM DBA_ENABLED_AGGREGATIONS ;
```

出力は、次のようなものです。

```
AGGREGATION             SERVICE                MODULE        ACTION
------------            --------------------   ----------    -------------
SERVICE_MODULE_ACTION   erp                    payroll       exceptions pay
SERVICE_MODULE          erp                    payroll
SERVICE_MODULE          hot_batch              posting
```

Oracleサービスの使用方法

ワークロードまたはアプリケーションのグループを管理するために、特定のアプリケーションまたはアプリケーションの一部の操作に割り当てるサービスを定義できます。また、作業の種類ごとにサービスとしてグループ化することもできます。たとえば、オンライン・ユーザーはあるサービスを使用し、バッチ処理は別のサービスを使用し、レポートはまた別のサービスを使用してデータベースに接続することができます。

1つのサービスを共有するすべてのユーザーで、サービス・レベルの要件を同じにすることをお薦めします。サービスには個別の特性を定義できるため、各サービスはそれぞれ別々の作業単位にできます。サービスの使用時には、多数のオプションを使用できます。これらのオプションを実装する必要はありませんが、それらを使用すると、アプリケーションのパフォーマンスが最適化できます。

サービスのデプロイメント・オプション

この項では、次のサービスのデプロイメントについて説明します。

- Oracle RACデータベースにおけるサービスの使用
- サービスの特性
- デフォルトのサービス接続
- サービス登録の制限

Oracle RACデータベースにおけるサービスの使用

サービスによって、位置の透過性がもたらされます。サービス名は複数のデータベース・インスタンスを識別することができ、インスタンスは複数のサービスに属することができます。複数のデータベース機能で、Oracle RACデータベース用のサービスを使用します。

この項には次のトピックが含まれます:

- サービスのOracle Clusterwareリソース
- サービスのデータベース・リソース・マネージャ・コンシューマ・グループのマッピング
- AWRによるサービスごとのパフォーマンス監視
- パラレル操作とサービス
- Oracle GoldenGateおよびOracle RAC

サービスのOracle Clusterwareリソース

サービスを定義すると、リソース・プロファイルは自動的に作成されます。リソース・ファイルには、Oracle Clusterwareによるサービスの管理方法と、PREFERREDインスタンスが停止した場合にサービスがフェイルオーバーされるインスタンスが定義されて

います。また、リソース・プロファイルでは、インスタンスおよびデータベースに対するサービスの依存性も定義します。この依存性の情報によって、データベースが停止した場合に、インスタンスおよびサービスが自動的に正しい順序で停止されます。

サービスのデータベース・リソース・マネージャ・コンシューマ・グループのマッピング

サービスはOracle Resource Managerと統合されており、リソース・マネージャでは、サービスを使用してインスタンスに接続するユーザーが使用するリソースを制限できます。Oracle Resource Managerでは、コンシューマ・グループをサービスにマッピングできるため、そのサービスを使用してインスタンスに接続しているユーザーは、指定されたコンシューマ・グループのメンバーになります。Oracle Resource Managerは、インスタンス・レベルで動作します。

AWRによるサービスごとのパフォーマンス監視

自動ワークロード・リポジトリ(AWR)によって生成されるメトリック・データは、様々なグループ(イベント、イベント・クラス、セッション、サービス、表領域メトリックなど)に編成されます。通常、Oracle Enterprise ManagerまたはAWRレポートを使用してAWRデータを表示します。

関連項目

- Oracle Databaseパフォーマンス・チューニング・ガイド

パラレル操作とサービス

デフォルトでは、Oracle RAC環境で、パラレルで処理されるSQL文はクラスタ内のすべてのノードで実行できます。

このクロスノードまたはノード間パラレル処理を適切に実現するには、ノード間パラレル処理によってインターコネクト・トラフィックが増大する可能性があるため、Oracle RAC環境でのインターコネクトのサイズが適切である必要があります。ノード間パラレル処理を制限するには、初期化パラメータPARALLEL_FORCE_LOCALを使用してOracle RAC環境でパラレル処理を制御します。このパラメータをTRUEに設定すると、パラレル処理サーバーは、SQL文が開始されたのと同じOracle RACノードからのみ実行できます。

サービスを使用して、パラレルSQL操作に使用できるインスタンスの数を制限します。デフォルトのデータベース・サービスを使用すると、パラレルSQL操作は、使用可能なすべてのインスタンスで実行できます。それぞれが1つ以上のインスタンスを含むサービスを必要な数のみ作成できます。パラレルSQL操作が開始されると、パラレル処理サーバーは、最初のデータベース接続で使用された特定のサービスを提供するインスタンス上でのみ起動されます。

PARALLEL_INSTANCE_GROUPは、Oracle RACパラメータで、サービスとともに使用すると、パラレル問合せ操作を、限られた数のインスタンスに制限できます。パラレル問合せ操作を、限られた数のインスタンスに制限するには、PARALLEL_INSTANCE_GROUP初期化パラメータにサービスの名前を設定します。これは、パラレル・リカバリやGV$問合せの処理などの他のパラレル操作には影響しません。

Oracle GoldenGateおよびOracle RAC

Oracle GoldenGateはOracle RAC機能を利用します。

Oracle GoldenGateがOracle RAC環境で構成されている場合、各キュー表には所有するインスタンスがあります。キュー表をホストするインスタンスに障害が発生しても、Oracle RACデータベースの別のインスタンスがキュー表の所有インスタンスにな

るため、Oracle GoldenGateは継続して稼働できます。

また、Oracle RACデータベースでは、バッファ・キューごとにサービスが作成されます。インスタンスの起動や停止などのために所有権が切り替わる場合、このサービスは常に、宛先キューの所有者インスタンスで実行され、このキューの所有権に従います。このサービスは、キューからキューへの伝播で使用されます。

サービスの特性

データベースに新しいサービスを作成した場合は、サービスごとに自動ワークロード管理の特性を定義する必要があります。サービスの特性には、次のものが含まれます。

- サービス名
- サービス・エディション
- サービス管理ポリシー
- サービスのデータベース・ロール
- インスタンスのプリファレンス
- サーバー・プールの割当て
- ランタイム接続ロード・バランシングのロード・バランシング・アドバイザの目標
- 接続時ロード・バランシングの目標
- 分散トランザクション処理
- TAFによるOCIクライアントのフェイルオーバー

サービス名

クライアントは、サービス名を使用して1つ以上のインスタンスに接続します。

各サービスにはサービス名があります。サービス名は、システム全体で一意である必要があります。

サービス名は、次の資格を満たしている必要があります。

- 名前は、英数字(a-z、A-Z、0-9)、アンダースコア(_)およびハイフン(-)で構成されている必要があります。
- 名前のサービス・ドメイン部分は、英数字(a-z、A-Z、0-9)、アンダースコア(_)、ドル記号($)、シャープ記号(#)、ピリオド(.)およびハイフン(-)で構成されている必要があります
- ドメイン修飾サービス名の形式は、service_name.service_domainです。
- データベースのデフォルト・サービスと同じ名前(db_unique_name.db_domain)でサービスを作成することはできません。

サービス・エディション

データベース・オブジェクトのエディションベースの再定義を使用すると、アプリケーションのオブジェクトが使用中であってもそれらのオブジェクトをアップグレードできます。データベース・サービスの作成時にそのエディション属性を設定したり、既存のサービスを変更してエディションを設定できます。サービス・エディションを設定すると、そのサービスを使用する接続は、このエディションを初期セッション・エディションとして使用します。サービスでエディション名が指定されていない場合、初期セッション・エディションはデータベースのデフォルト・エディションです。

次のように、SRVCTLを使用してサービス・エディションを設定できます。

```
$ srvctl modify service -db hr -s crmsrv -edition e2
```

サービス管理ポリシー

Oracle Clusterwareを使用してデータベースを管理する場合、-policyパラメータを指定してsrvctl add serviceコマンドを使用してサービスを追加するときに、個々のデータベース・サービス用に起動オプションを構成できます。

サービスの管理ポリシーをAUTOMATIC(デフォルト)に設定した場合は、SRVCTLを使用してデータベースを起動するとサービスが自動的に起動されます。管理ポリシーをMANUALに設定した場合、サービスは自動起動されず、SRVCTLを使用して手動で起動する必要があります。MANUALに設定しても、Oracle Clusterwareは、実行中のサービスを監視し、障害が発生すると再起動されます。Oracle RAC 11gリリース2(11.2)より前は、すべてのサービスが、MANUAL管理ポリシーで定義されているかのように動作していました。

CRSCTLを使用したOracle Clusterwareの停止および再起動は障害として扱われ、サービスが実行中の場合は再起動されます。

ノート:

管理者管理データベースの自動サービスを使用する場合、計画的なデータベースの起動中に、優先インスタンスではなく最初のインスタンスでサービスが起動することがあります(開始されたインスタンスが優先サービスと使用可能なサービスを組み合せたリストに含まれている場合)。

関連項目

- srvctl add service

サービスのデータベース・ロール

目的の環境でOracle Data Guardを構成してある場合は、サービスの追加時または変更時に、SRVCTLを使用して該当するコマンドに-roleパラメータを指定することでサービスのロールを定義できます。

サービスにロールを指定すると、Oracle Clusterwareは、データベース・ロールがそのサービスに指定したロールに一致した場合にのみ自動的にサービスを起動します。有効なロールは、PRIMARY、PHYSICAL_STANDBY、LOGICAL_STANDBYおよびSNAPSHOT_STANDBYで、1つのサービスに複数のロールを指定できます。

ノート:

サービス・ロールのみがサービスの自動開始を制御します。手動で開始する SRVCTL を使用することで、ロールが一致しない場合でもサービスは正常に実行されます。

REDO Apply (フィジカル・スタンバイ・データベース)は、ユーザーが構成できるすべてまたは一部のスタンバイ・インスタンス上で実行できます。別のスタンバイ・インスタンスを追加することで、必要に応じてREDO Applyのパフォーマンスをスケールできます。

クラスタ内の複数のデータベースが同じサービス名を提供すると、Oracle RACは、該当するすべてのデータベースにわたってそのサービスへの接続を均等に分散します。これはOracle Data Guardのスタンバイ・データベースおよびアクティブ・データベースに役に立ちますが、サービスへのクライアント接続を特定のデータベースに割り当てる必要がある場合、サービス名はクラスタ内で一意である(他のデータベースによって提供されない)必要があります。

関連項目

- 『Oracle Data Guard概要および管理』

インスタンスのプリファレンス

管理者管理データベースに対してサービスを定義する場合は、SRVCTLに-preferredパラメータを使用して、そのサービスを通常サポートするインスタンスを定義します。

このようなインスタンスを、優先インスタンスといいます。サービスの優先インスタンスが失敗した場合に備えて、-availableパラメータを指定してSRVCTLを使用し、サービスをサポートするその他のインスタンスを定義することもできます。このようなインスタンスを、使用可能インスタンスといいます。

優先インスタンスを指定する場合は、サービスが通常実行されるインスタンスの数を指定します。これは、サービスの最大カーディナリティです。Oracle Clusterwareは、サービスを構成したインスタンス数で常にサービスが実行されることを確認しようとします。その後は、インスタンス障害またはサービスの計画的な再配置のために、使用可能インスタンスでサービスが実行される場合があります。

インスタンスが失敗した場合、Oracle Clusterwareは優先リストおよび使用可能なリストを順序付けられたリストと解釈するため、リストに複数のインスタンスがあると、Oracle Clusterwareによってサービスがどの使用可能インスタンスに再配置されるかをある程度制御できます。ただし、計画済操作時は、サービスを現在提供していない、優先リストまたは使用可能リスト内のインスタンスにサービスを手動で転送できます。

Oracle Databaseでは、使用可能インスタンスに移動されたサービスは、優先インスタンスを再起動しても、優先インスタンスには自動的に戻りません。これには、次の理由があります。

- サービスが、指定した数のインスタンスで実行されている。
- 現在のインスタンスでサービスを維持することによって、より高度なサービス可用性が提供される。
- サービスを最初の優先インスタンスに戻さないことで、2回目の機能停止が回避される。

Oracle Databaseリリース19.3以降では、サービスの-failback属性にyesを指定すると、最後の優先インスタンスが停止して使用可能なインスタンスにフェイルオーバーした後、優先インスタンスが使用可能になったときに、サービスがそのインスタンスにフェイルバックされます。以前のリリースでは、FANコールアウトを使用して、優先インスタンスへのフェイルバックを自動化できます。

関連項目

- Oracle RACの管理ツール

サービスの関連付け

Oracle RACは、可能な場合、同じCOLOCATION_TAGを持つクライアントを同じデータベース・インスタンスにルーティングします。

同じインスタンスのセッションを関連付けると、インスタンス間通信が減少するため、同じインスタンスで実行することでメリットがあ

るワークロードのパフォーマンスが向上します。COLOCATION_TAGは、*Oracle Database Net Services*リファレンスの説明に従い、サービスによって使用されるTNS接続文字列のCONNECT_DATAパラメータで構成します。

関連項目

- COLOCATION_TAG

サーバー・プールの割当て

ポリシー管理データベースのサービスを定義する場合、-serverpoolパラメータを指定したSRVCTLを使用して、データベースがホストされているサーバー・プールにサービスを割り当てます。

サービスは、-cardinalityパラメータを使用して、UNIFORM (サーバー・プール内のすべてのインスタンスで実行)またはSINGLETON (サーバー・プール内の単一インスタンスでのみ実行)のいずれかとして定義できます。singletonサービスの場合、Oracle RACはそのサービスがアクティブなサーバー・プール内でインスタンスを選択します。そのインスタンスで障害が発生すると、サービスはサーバー・プール内の別のインスタンスにフェイルオーバーします。サービスは1つのサーバー・プールでのみ実行でき、すべてのサーバー・プールに少なくとも1つのサービスを含めることをお薦めします。

ノート:

Oracle Database Quality of Service Management (Oracle Database QoS Management)では、サーバー・プールの最大サイズが 1 である場合、そのサーバー・プールの singleton サービスを管理します。

関連項目

- Oracle RACの管理ツール

ランタイム接続ロード・バランシングのロード・バランシング・アドバイザの目標

ランタイム接続ロード・バランシングを使用すると、アプリケーションは、ロード・バランシング・アドバイザ・イベントを使用して、ユーザーにより適切なサービスを提供できます。Oracle JDBC、Oracle Universal Connection Pool for Java、OCIセッション・プール、ODP.NETおよびOracle WebLogic Server Active GridLink for Oracle RACクライアントは、ロード・バランシング・アドバイザリ・イベントを利用するために自動的に統合されます。ロード・バランシング・アドバイザは、サービスに対してインスタンスで提供されている現在のサービス・レベルをクライアントに通知します。ロード・バランシング・アドバイザを有効にするには、サービスを作成または変更するときに、SRVCTLに-rlbgoalパラメータを使用します。

また、そのインスタンスに送るワークロードの推奨量を提示します。最適なサービス品質(単一のトランザクションが完了した効率)または最適なスループット(完了ジョブまたは長時間の問合せが完了した効率)のどちらに基づいてサービスに接続するかは、目標によって決定されます。

接続時ロード・バランシングの目標

Oracle Net Servicesでは、接続ロード・バランシングにより、サービスをサポートしているすべてのインスタンスにユーザー接続を分散できます。

各サービスに対して、-clbgoalパラメータを指定したSRVCTLを使用して、接続時ロード・バランシングの目標を設定し、リスナーにロード・バランシングを使用させる方法を定義できます。接続は、リスナーにセッション数を使用するように指定するLONG

(接続プールやSQL*FORMSなど)、またはリスナーに応答時間やスループットの統計を使用するように指定するSHORTに分類されます。

ロード・バランシング・アドバイザが有効化されていると(-rlbgoalパラメータがNONEに設定されていない)、接続ロード・バランシングはロード・バランシング・アドバイザを使用しようとします(ロード・バランシングの目標がSHORTまたはLONGのどちらに設定されているかは関係ありません)。ロード・バランシングがSHORTに設定されている場合は、サービスのGOODNESS値を使用して、すべての接続リクエストが1つのインスタンスに集中しないようにします。ロード・バランシングがLONGに設定されている場合は、run queue length (サービスがシングルトンの場合)、またはsession count (サービスが均一の場合)を使用します。シングルトン・サービスはサーバー・プール内の1つのサーバー・インスタンスでのみ実行されますが、均一サービスはサーバー・プール内のすべてのサーバー・インスタンスで実行されます。

分散トランザクション処理

Oracle XAアプリケーションには固有の要件があります。Oracleでは、Oracle RAC全体にわたるグローバル・トランザクションを使用できます。最適なパフォーマンスを得るには、ほとんどのトランザクションにXAアフィニティ(同一インスタンスのすべてのブランチ)を使用して、必要に応じてグローバル・トランザクションを使用します。接続プール(ユニバーサル接続プールやWebLogic Serverなど)でXAアフィニティを使用できます。SRVCTLを使用して作成したシングルトン・サービスを使用することもできます。さらに、SRVCTLを使用して分散トランザクション処理パラメータ(-dtp)をTRUEに設定します(同一のOracle XAブランチを一時停止および再開する場合)。ただし、これは計画メンテナンスのローリングを提供しないため、通常は使用しないでください。

関連項目

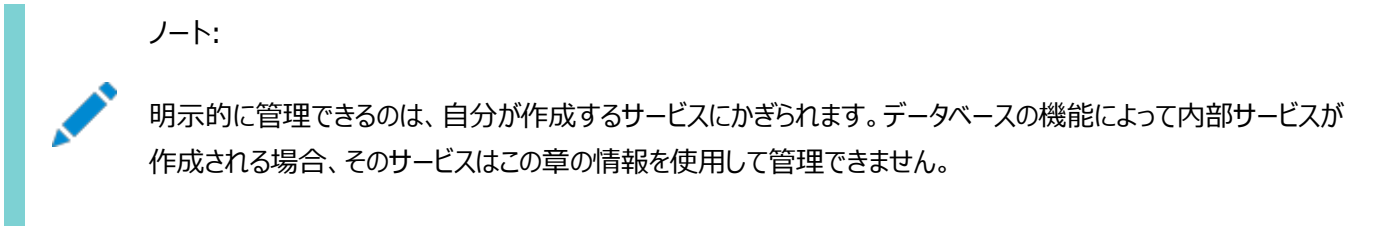
- Oracle RACの分散トランザクション処理
- srvctl add service

デフォルトのサービス接続

Oracle RAC Databaseには、DB_UNIQUE_NAMEが設定されている場合はこれにより識別され、設定されていない場合はDB_NAMEまたはPDB_NAMEで識別されるOracle Databaseサービスが含まれます。このデフォルトのサービスは、Oracle RAC環境のすべてのインスタンスで常に使用可能です(制限モードのインスタンスを除く)。このサービスまたはサービスのプロパティは、変更できません。また、データベースでは、次の2つの内部サービスがサポートされています。

- SYS$BACKGROUND(バックグラウンド・プロセスのみで使用されるサービス)
- SYS$USERS(どのアプリケーション・サービスとも関連付けられていないユーザー・セッションのデフォルト・サービス)

これらのサービスはすべて、内部管理に使用されます。計画済停止やOracle Data Guardへのフェイルオーバーを実行するためにこれらの内部サービスを停止したり無効にすることはできません。これらのサービスをクライアント接続に使用しないでください。



ノート:

明示的に管理できるのは、自分が作成するサービスにかぎられます。データベースの機能によって内部サービスが作成される場合、そのサービスはこの章の情報を使用して管理できません。

制限されたサービス登録

この機能により、デフォルトでローカルIPアドレスからのリスナー登録のみが許可され、登録要求がリスナーにより許可される一連のIPアドレスまたはサブネットを構成および動的に更新する機能が提供されます。

セキュリティはすべての企業で高い優先度を持ち、ネットワーク・セキュリティおよびデータベースへのアクセスの制御は、セキュリティ確保全体における不可欠な要素です。リスナーへのデータベース・インスタンスの登録は、要求元が有効なノードである場合にのみ成功します。ネットワーク管理者は、有効なノードおよび除外ノードのリストを指定したり、有効なノードの確認を無効にすることができます。有効なノードのリストでは、データベースに登録できるノードやサブネットを明示的にリストします。除外ノードのリストでは、データベースに登録できないノードを明示的にリストします。動的登録を制御することによって、Oracle RACデプロイメントの管理性およびセキュリティが向上します。

登録のための有効なノードの確認(VNCR)はデフォルトで有効になっています。デフォルト構成では、リスナーは、SCANリスナーのサブネット内にあり、プライベート・ネットワークにアクセスできるノードからの登録要求のみを受け入れます。非SCANリスナーでは、ローカル・ノード上のインスタンスからの登録のみが受け入れられます。registration_invited_nodes_aliasパラメータをlistener.oraファイルで使用するか、次のようにSRVCTLを使用してSCANリスナーを変更することによって、リモート・ノード、または有効なノードのリスト上のSCANリスナーのサブネット外のノードを手動で含める必要があります。

```
$ srvctl modify scan_listener -invitednodes node_list -invitedsubnets subnet_list
```

> ノート:
>
> Oracle Grid Infrastructure 12c 以降、SCAN リスナーについて、VALID_NODE_CHECKING_REGISTRATION_listener_name および REGISTRATION_INVITED_NODES_listener_name パラメータが listener.ora ファイルに設定されている場合、リスナー・エージェントはこれらのパラメータを上書きします。

SRVCTLユーティリティを使用してinvitednodes値とinvitedsubnets値を設定すると、リスナー・エージェントは自動的にVALID_NODE_CHECKING_REGISTRATION_listener_nameをSUBNETに設定し、REGISTRATION_INVITED_NODES_listener_nameをlistener.oraファイルで指定されたリストに設定します。

CRSによって管理されるその他のリスナーの場合、リスナー・エージェントは、listener.oraファイルでまだ設定されていない場合にのみ、listener.oraファイルでVALID_NODE_CHECKING_REGISTRATION_listener_nameを設定します。SRVCTLユーティリティでは、SCAN以外のリスナーについてinvitednodes値とinvitedsubnets値の設定はサポートされていません。リスナー・エージェントは、SCAN以外のリスナーについてlistener.oraファイルのREGISTRATION_INVITED_NODES_listener_nameを更新しません。

関連項目

- Oracle Database Net Services管理者ガイド

サービスの管理

Oracle Enterprise ManagerおよびSRVCTLユーティリティを使用して、管理サービスを作成および管理できます。次の項では、これらのツールを使用して、サービスに関連するタスクを実行する方法について説明します。

この項には次のトピックが含まれます:

- サービスの管理の概要
- Oracle Enterprise Managerを使用したサービスの管理
- SRVCTLを使用したサービスの管理

ノート:

DBMS_SERVICE パッケージを使用して、サービスとサービス属性を作成または変更することもできますが、このパッケージを使用して行われた設定は、SRVCTL によって上書きされます。DBMS_SERVICE パッケージは、Oracle RAC データベースが使用するサービスで使用したり、Oracle Restart の使用時や Oracle Clusterware による単一インスタンス・データベースの管理時に使用することはお薦めしません。

サービスの管理の概要

サービスを作成して管理する場合、データベースで実行する作業を管理しやすい単位に分割します。

サービスを使用する目的は、データベース・インフラストラクチャを最大限有効に利用することです。ビジネス要件に基づいて、サービスを作成およびデプロイできます。Oracle Databaseは各サービスのパフォーマンスを計測できます。DBMS_MONITORパッケージを使用すると、サービス内のアプリケーション・モジュールおよびモジュールの個別のアクションを両方定義して、これらのアクションのしきい値を監視できるようになり、これによってワークロードを管理して必要に応じて容量を供給することができます。

データベースに新しいサービスを作成した場合は、サービスごとに自動ワークロード管理の特性を定義する必要があります(「サービスの特性」を参照)。

関連項目:

OracleクラスタでOracle Database QoS Managementを使用している場合、データベース・サービスの構成方法の詳細は、『*Oracle Database Quality of Service Management*ユーザーズ・ガイド』を参照してください。

サービスの作成に加えて、次の作業を実行できます。

- サービスの削除。作成したサービスは削除できます。ただし、Oracle Databaseで作成されたデフォルトのデータベース・サービスのプロパティは、削除したり、変更することはできません。
- サービスのステータスの確認。サービスは、使用可能インスタンスごとに別々のロールが割り当てられる場合があります。多くのサービスを使用する複雑なデータベースでは、すべてのサービスの詳細を把握しておくことは困難な場合があります。そのため、インスタンスごとまたはサービスごとにステータスの確認が必要になる場合があります。たとえば、特定のインスタンス、または特定のインスタンスを実行するOracleホームに変更を加える前に、このインスタンスのサービスのステータスの確認が必要な場合があります。
- データベースまたはインスタンスのサービスの起動および停止。インスタンスへのクライアント接続に使用するサービスは、事前に起動されている必要があります。たとえば、SRVCTLコマンド`srvctl stop database -db db_unique_name`を実行してデータベースを停止した場合(`db_unique_name`は停止するデータベース名)、そのデータベースへのすべて

サービスが停止されます。サービス管理ポリシーによっては、データベースの起動時にサービスを手動で再起動する必要がある場合があります。srvctl stop databaseおよびsrvctl stop serviceの両方のコマンドは、接続を強制的に切断する-forceオプションを受け入れます。計画済停止のセッションを排出する場合、-forceオプションは使用しません。

ノート:

Oracle RAC データベースで Oracle Database QoS Management が有効になっている場合、サービスは、停止後に自動的に再起動されます。

- サービスのコンシューマ・グループへのマッピング。サービスをリソース・マネージャのコンシューマ・グループにマッピングして、サービスがインスタンスで使用可能なリソースの量を制限することができます。コンシューマ・グループを作成し、サービスをこのグループにマッピングする必要があります。
- データベースまたはインスタンスのサービスの有効化および無効化。デフォルトでは、障害が発生すると、Oracle Clusterwareはサービスの再起動を自動的に試みます。サービスを無効化すると、この動作を回避できます。サービスの無効化は、データベースまたはインスタンスのメンテナンスを実行する必要がある場合、たとえばアップグレードの実行中に接続要求の成功を回避する必要がある場合などに便利です。
- 別のインスタンスへのサービスの再配置。たとえば、クラスタ・ノードを追加または削除した後に、あるインスタンスから別のインスタンスにサービスを移動して、ワークロードを再分散させることができます。

ノート:

- サービスを使用する場合は、SERVICE_NAMES パラメータに値を設定しないでください(作成したサービスおよびデフォルトのデータベース・サービスに対するこのパラメータの設定値は Oracle Database によって制御されます)。この章で説明しているサービスの機能は、SERVICE_NAMES を設定した場合に Oracle Database で使用可能な機能とは直接関係ありません。また、このパラメータに値を設定した場合、サービスの使用によって得られるメリットが失われてしまう可能性があります。
- サービス・ステータス情報は、SRVCTL から、またはサービス関連データベース・ビュー(dba_services など)から取得する必要があります。
- DISPATCHERS 初期化パラメータを使用してサービスを指定する場合は、SERVICE_NAMES パラメータ内のすべてのサービスが無効になり、管理できなくなります。(たとえば、SRVCTL コマンドでサービスを停止すると、サービスに接続しているユーザーは停止されません。)

関連項目

- Oracle RACのクライアントの有効化
- Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス

Oracle Enterprise Managerを使用したサービスの管理

「クラスタ管理データベース・サービス」ページは、サービスに関連するすべてのタスクを開始するためのマスター・ページです。

次のように、このページにアクセスします。

1. Oracle Enterprise Managerで、クラスタ・データベースのホームページに移動します。

   関連項目:

   Oracle Enterprise Managerへのログインの詳細は、『*Oracle Database 2*日でデータベース管理者』を参照してください。

2. 「可用性」メニューから、「クラスタ管理データベース・サービス」を選択して、「クラスタ管理データベース・サービス」ページを表示します。
3. Oracle RACデータベースおよびホスト・オペレーティング・システムに対する資格証明を入力または確認し、「続行」をクリックして、「クラスタ管理データベース・サービス」ページを表示します。

「クラスタ管理データベース・サービス」ページで、ドリルダウンして次のタスクを実行できます。

- クラスタのサービス・リストの表示
- 現在、各サービスを実行しているインスタンスの表示
- ポリシー管理環境でサービスを提供するサーバー・プールおよびノードの表示
- 各サービスのステータスの表示
- サービスの作成および編集
- サービスの起動および停止
- サービスの有効化および無効化
- サービスのインスタンスレベルのタスクの実行
- サービスの削除

ノート:

クラスタ・データベースへアクセスするには、SYSDBA 資格証明を所有している必要があります。「クラスタ管理データベース・サービス」では、SYSDBA 以外での接続は許可されません。

関連項目

- Oracle Database 2日でデータベース管理者

関連項目:

Oracle Enterprise Managerを使用したサービスの管理の詳細は、Oracle Enterprise Managerのオンライン・ヘルプを参照してください。

SRVCTLを使用したサービスの管理

SRVCTLを使用してサービスを作成した場合、別のSRVCTLコマンドでそのサービスを起動する必要があります。

一方、後で手動でサービスを停止または再起動する必要がある場合もあります。さらに、自動的な再起動が実行されないようにサービスを無効化したり、サービスを手動で再配置したり、サービスに関するステータス情報を取得することもあります。次の項では、SRVCTLを使用して次の管理タスクを実行する方法について説明します。

- SRVCTLを使用したサービスの作成
- アプリケーション・コンティニュイティおよびトランザクション・ガードのサービスの作成
- SRVCTLを使用したサービスの起動および停止
- SRVCTLを使用したサービスの有効化および無効化
- SRVCTLを使用したサービスの再配置
- SRVCTLを使用したサービス・ステータスの取得
- SRVCTLを使用したサービスの構成の取得

SRVCTLを使用したサービスの作成

SRVCTLを使用してサービスを作成するには、コマンドラインで`srvctl add service`コマンドを使用します。

関連項目

- srvctl add service

アプリケーション・コンティニュイティおよびトランザクション・ガードのサービスの作成

アプリケーション・コンティニュイティのサービスを構成するには、SRVCTLを使用してサービスを作成する場合、`-failovertype`パラメータをTRANSACTIONに設定し、`-commit_outcome`をTRUEに設定します。

アプリケーションでアプリケーション・コンティニュイティおよびトランザクション・ガードを使用する場合は、サービスを構成する必要があります。この項では、実装予定の機能に応じてこれらのアプリケーション・サービスを構成する方法について説明します。

アプリケーション・コンティニュイティのサービスの作成

また、アプリケーション・コンティニュイティおよびロード・バランシングのその他のサービス・パラメータに値を設定することもできます。

- `-replay_init_time`: リプレイが開始できる時間を秒単位で指定します。リプレイが開始されるまでに許可する時間に基づいて値を選択することをお薦めします。デフォルト値は300秒です。
- `-retention`: コミット結果情報がデータベース内に保持される時間(秒)を指定します。デフォルト値は86400 (1日)です。
- `-session_state`: COMMITが実行され後にそのトランザクションの状態が変更された場合、セッションが失われている場合はトランザクションをリプレイしてその状態を再確立することはできません。アプリケーション・コンティニュイティを構成する場合、アプリケーションは、初期設定後のセッション状態が動的であるか静的であるか、および要求内の過去のCOMMIT操作を継続するのが適切であるかどうかに応じて分類されます。
    - 動的: (デフォルト)セッション状態の変更が初期化で完全にカプセル化されていない場合、およびフェイルオーバー時にコールバックで完全に取得できない場合、そのセッションは動的な状態です。要求内の最初のトランザクションがコミットされると、次の要求が開始されるまでフェイルオーバーは内部的に無効化されます。これは、ほとんどすべてのアプリケーションが要求に使用するデフォルト・モードです。

- 静的: (要求での特殊設定) NLS設定やPL/SQLパッケージの状態など、すべてのセッション状態の変更を初期化コールバックで繰り返すことができる場合、そのセッションは静的な状態です。この設定は、セッション状態を変更しないデータベース診断アプリケーションのみに使用されます。コールバックによって再確立できない非トランザクション状態変更がリクエスト内にある場合は、STATICを指定しないでください。どの状態を指定すればよいかがわからない場合は、DYNAMICを使用します。

- -failoverretry: 各接続試行に対する接続試行回数であり、推奨値は30です。
- -failoverdelay: 各接続試行間の遅延(秒単位)であり、推奨値は10です。
- -notification: FANは強く推奨されているため、この値をTRUEに設定して、OCIクライアントとODP.Netクライアントでを有効化します。
- -clbgoal: 接続ロード・バランシングの場合、実行時ロード・バランシングの使用時はSHORTを使用します。
- -rlbgoal: ランタイム・ロード・バランシングの場合は、SERVICE_TIMEに設定します。

ポリシー管理されたOracle RACデータベースに対してアプリケーション・コンティニュイティのサービスを作成するには、次のようなコマンドを使用しますが、racdbはOracle RACデータベースの名前を、app2は変更するサービスの名前を、そしてSvrpool1はサービスが提供されるサーバー・プールの名前を表します。

```
$ srvctl add service -db racdb -service app2 -serverpool Srvpool1
  -failovertype TRANSACTION -commit_outcome TRUE -replay_init_time 1800
  -retention 86400 -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
  -failoverretry 30 -failoverdelay 10
```

SRVCTLを使用してアプリケーション・コンティニュイティの既存のサービスを変更するには、次のようなコマンドを使用しますが、racdbは使用しているOracle RACデータベースの名前を、そしてapp1は変更するサービスの名前を表します。

```
$ srvctl modify service -db racdb -service app1 -clbgoal SHORT
  -rlbgoal SERVICE_TIME -failoverretry 30 -failoverdelay 10
  -failovertype TRANSACTION -commit_outcome TRUE -replay_init_time 1800
  -retention 86400 -notification TRUE
```

トランザクション・ガードのサービスの作成

トランザクション・ガードは有効にするが、アプリケーション・コンティニュイティは有効にしない場合は、SRVCTLを使用してサービスを作成し、-commit_outcome TRUEのみを設定します。

SRVCTLを使用して、トランザクション・ガードを有効にするように既存のサービスを変更するには、次のようなコマンドを使用しますが、racdbは使用しているOracle RACデータベースの名前を、そしてapp2は変更するサービスの名前を表します。

```
$ srvctl modify service -db racdb -service app2 -commit_outcome TRUE
  -retention 86400 -notification TRUE
```

前述の例では、-retentionパラメータは、履歴を保持する時間(秒単位)を指定しています。また、-notificationパラメータはTRUEに設定され、FANイベントを有効化しています。

トランザクション・ガードを使用するには、DBAは次のように権限を付与する必要があります。

```
GRANT EXECUTE ON DBMS_APP_CONT;
```

関連項目

- Oracle Database開発ガイド

SRVCTLを使用したサービスの起動および停止

アプリケーションがサーバーを使用して接続するために、サービスを起動する必要があります。停止したサービスは、一時的に使用できなくなりますが、引き続き自動再起動およびフェイルオーバーの対象となっています。

サービスを起動または停止するには、コマンドラインで次のSRVCTL構文を入力します。

```
$ srvctl start service -db db_unique_name [-service service_name_list]
    [-instance inst_name] [-startoption start_options]

$ srvctl stop service -db db_unique_name -service service_name_list
    [-instance inst_name] [-startoption start_options]
```

SRVCTLを使用したサービスの有効化および無効化

サービスを無効化にすると、Oracle Clusterwareは、そのサービスを自動起動、フェイルオーバーまたは再起動の対象とみなさなくなります。アプリケーション・メンテナンスを実行する場合は、そのメンテナンス操作が完了するまでサービスを無効化にすることで、Oracle Clusterwareが誤ってサービスを再起動しないようにできます。再び通常操作でサービスを使用できるようにするには、サービスを有効化します。

サービスを有効化および無効化するには、コマンドラインから次のSRVCTL構文を使用します。

```
$ srvctl enable service -db db_unique_name -service service_name_list
    [-instance inst_name]

$ srvctl disable service -db db_unique_name -service service_name_list
    [-instance inst_name]
```

SRVCTLを使用したサービスの再配置

サービスを再配置するには、コマンドラインからsrvctl relocate serviceを実行します。このコマンドを使用するのは、サービスが使用可能インスタンスにフェイルオーバーしたが、そのインスタンスの再起動後に、優先インスタンスにサービスを移動させたい場合です。

次のコマンドを実行すると、crmサービスがインスタンスapps1からインスタンスapps3に再配置されます。

```
$ srvctl relocate service -db apps -service crm -oldinst apps1 -newinst apps3
```

次のコマンドを実行すると、ノード構文を使用してcrmサービスがnode1からnode3に再配置されます。

```
$ srvctl relocate service -db apps -service crm -currentnode node1
    -targetnode node3
```

SRVCTLを使用したサービス・ステータスの取得

サービスのステータスを取得するには、コマンドラインからsrvctl status serviceコマンドを実行します。たとえば、次のコマンドを実行すると、appsデータベースで実行中のサービスのステータスが戻されます。

```
$ srvctl status service -db apps
Service erp is running on nodes: apps02,apps03
```

```
Service hr is running on nodes: apps02,apps03
Service sales is running on nodes: apps01,apps04
```

SRVCTLを使用したサービスの構成の取得

サービスの高可用性構成を取得するには、コマンドラインからsrvctl config serviceコマンドを実行します。たとえば、次のコマンドを実行すると、appsデータベースで実行中のerpサービスの構成が戻されます。

```
$ srvctl config service -db apps -service erp
Service name: erp
Service is enabled
Server pool: pool1
Cardinality: 1
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: true
Global: false
Commit Outcome: true
Failover type: TRANSACTION
Failover method: NONE
TAF failover retries: 30
TAF failover delay: 10
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: SERVICE_TIME
TAF policy specification: NONE
Edition:
Pluggable database name:
Maximum lag time: ANY
SQL Translation Profile:
Retention: 86400 seconds
Replay Initiation Time: 1800 seconds
Session State Consistency: STATIC
Preferred instances: apps
Available instances:
```

グローバル・サービス

Oracle RACはデータベース・サービスをサポートし、単一クラスタ内のインスタンス間でのサービスレベル・ワークロード管理を有効にします。

グローバル・サービスでは、共有サービスを提供する一連のレプリケート・データベースに動的ロード・バランシング、フェイルオーバーおよびサービスの集中管理を提供します。この一連のデータベースには、Oracle Data Guard、Oracle GoldenGateまたはその他のレプリケーション・テクノロジによって相互接続されたOracle RACおよび非クラスタOracle Databaseが含まれる場合があります。

グローバル・サービスを作成および使用する場合は、次のワークロード管理機能を使用できます。

- 優先データベースおよび使用可能なデータベースをグローバル・サービスに指定する機能
- レプリケーション・ラグの処理
- クライアントとサーバー間の地理的アフィニティ

- 接続ロード・バランシング
- ランタイム・ロード・バランシング
- 内部データベース・サービス・フェイルオーバー
- 高速接続フェイルオーバー
- 接続時フェイルオーバー
- アプリケーション・コンティニュイティ
- トランザクション・ガード
- 既存のクライアントとの下位互換性

ノート:

SRVCTL を使用して Oracle RAC データベース内のグローバル・サービスのインスタンス配置を管理できますが、GDSCTL を使用する場合は他のグローバル・サービス属性のみを管理できます。

関連項目

- Oracle Database Global Data Services概要および管理ガイド

サービス指向バッファ・キャッシュ・アクセス

サービス指向バッファ・キャッシュ・アクセスでは、データが属するサービスによってデータを管理することでパフォーマンスが向上します。

サービスを経由したオブジェクトのアクセスは、徐々にデータベースにマップされて永続化されます。この情報は、パフォーマンス向上のために使用できます。サービスを通じてアクセスされるブロックは、サービスを実行しているインスタンス内にキャッシュされますが、それよりも、サービスを実行していない場所には情報がキャッシュされないということが重要になります。

この情報は、サービスの開始前に、キャッシュを事前ウォーミングする際にも使用できます。サービスの起動は、インスタンスの起動またはサービスの再配置のどちらかでトリガーされます。サービス指向バッファ・キャッシュ・アクセスは、そのサービスのユーザーに安定したパフォーマンスを提供します。これは、サービスのユーザーがアクセスするブロックが、新しく再配置されたインスタンスにキャッシュされるためです。

サービスへの接続: 例

次の例では、サービスを作成する方法を示し、次に異なるクライアント・メソッドを使用してこのサービスに接続するいくつかの例を示します。

この例では、ランタイム・ロード・バランシングで、サービスが次のように有効になっています。

- サービス名: HR.example.com
  - CRMという名前のデータベース上で実行されています。
  - システムは4つのノードで構成されています。

- -rlbgoalパラメータの値としてSERVICE_TIMEを指定します。
- リスナーのSCANアドレスはrws3010104-scan.example.comです。
- リスナー・ポートは1585です。

サービスのカーディナリティは2ですが、必要な場合は、任意のCRMデータベース・インスタンスによって提供できます。サービス構成は、次のとおりです。

- 優先インスタンス: CRM1、CRM2
- 使用可能なインスタンス: CRM3、CRM4
- -clbgoalパラメータの値としてSHORTを指定します。

このサービスを使用するアプリケーションはアプリケーション・コンティニュイティを利用するため、-failovertypeおよび-commit_outcomeを設定する必要があります。デフォルトの保存パラメータを使用しますが、接続試行間に10秒の遅延を設定し、接続取得の失敗まで最大40回の再試行を設定します。

SRVCTLを使用したHRサービスの作成

次のように、SRVCTLを使用してHRサービスを作成します。

```
$ srvctl add service -db CRM -service HR.example.com -preferred CRM1,CRM2
  -available CRM3,CRM4 -clbgoal SHORT -failovertype TRANSACTION
  -commit_outcome TRUE -failoverdelay 10 -failoverretry 40
```

次のように、HR.example.comサービスを起動します。

```
$ srvctl start service -db CRM -service HR.example.com
```

このサービスは、最大2つのインスタンス上で使用できるようになり、CRM1およびCRM2が優先インスタンスです。

JDBCアプリケーションからHRサービスへの接続

この例では、HRサービスに接続するアプリケーションは、JDBC thinドライバでJDBC Universal Connection Poolを使用するJDBCアプリケーションです。

この例では、URLは、データベース指定子にthinスタイルのサービス名形式を指定して構築されます。高速接続フェイルオーバーが有効化され、リモートOracle Notification Serviceが構成され、クラスタ上のOracle Notification Serviceデーモンはポート6200でリスニングします。

```
//import packages and register the driver
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import oracle.ucp.jdbc.PoolDataSourceFactory;
import oracle.ucp.jdbc.PoolDataSource;
PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();
//set the connection properties on the data source.
pds.setConnectionPoolName("FCFPool");
pds.setFastConnectionFailoverEnabled(true);
pds.setONSConfiguration("nodes=rws3010104-scan.example.com:6200");
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin:@//rws3010104-scan.example.com:1585/HR.example.com");
pds.setUser("HR");
```

```
pds.setPassword("hr");


//Override any pool properties.
pds.setInitialPoolSize(5);

//Get a database connection from the datasource.

Connection conn = pds.getConnection();

// do some work

//return connection to pool
conn.close();
conn=null
```

関連項目

- Oracle Universal Connection Pool開発者ガイド

脚注の凡例

脚注1:

実行時接続ロード・バランシングは、基本的には作業リクエストをセッション・プール内で作業の処理に最も適切なセッションにルーティングする操作です。既存のセッション・プールからセッションを選択すると、有効になります。このため、ランタイム接続ロード・バランシングは、きわめて頻度の高いアクティビティです。

6 アプリケーション・コンティニュイティの確保

アプリケーション・コンティニュイティは、データベース・セッションを使用不可にするリカバリ可能なエラーの後に、データベースに対するリクエストを中断することなく、迅速な方法でリプレイできるようにする機能です。そのため、ユーザーにとって停止は、単なるリクエストの実行の遅延にしか見えなくなります。

リクエストには、トランザクション処理や非トランザクション処理が含まれる場合があります。リプレイが成功した後、アプリケーションはデータベース・セッションが中断された時点から処理を続行できるため、資金の振替や航空券の予約などに何が起こったかわからない不安な状態にユーザーを陥れることもなく、アプリケーションがオンライン状態に戻ったときの、ログインのオーバーロードからリカバリするために中間層サーバーを再起動する必要もなくなります。アプリケーション・コンティニュイティを使用すると、計画的な停止または計画外の停止の多くをマスクすることでエンド・ユーザーの使用感が向上します。アプリケーション開発者がリクエストをリカバリする必要はありません。

アプリケーション・コンティニュイティを使用しない場合、次のような理由により、アプリケーションが停止を安全な方法でマスクすることはほとんど不可能です。

- 入力したデータ、戻されたデータ、および変数がキャッシュされたまま、クライアントの状態が現時点のままになります。
- COMMITが発行されていた場合、クライアントまたはアプリケーションで受信されていないCOMMIT失敗のメッセージは取得できなくなります。
- ある時点でインダウト・トランザクションのステータスをチェックしても、その後でCOMMITしないという保証はありません。
- アプリケーションが処理する必要がある非トランザクションのデータベース・セッションの状態が失われます。
- リクエストが続行可能である場合、データベースおよびデータベース・セッションが正しい状態である必要があります。

その一方で、アプリケーション・コンティニュイティを使用すると、Oracle Database、Oracleドライバ、およびOracle接続プールのすべてが連携して、安全で信頼性の高い方法で多くの停止をマスクします。

アプリケーション・コンティニュイティは、マスク可能な停止のマスクを試行することにより、開発者の生産性を向上させます。ただし、次の場合は、従来どおりアプリケーションでエラー処理が行われる必要があります。

- 無効な入力データなどのリカバリ不能なエラー。(アプリケーション・コンティニュイティはリカバリ可能なエラーにのみ適用されます。)
- アプリケーションでの具象クラスの使用などの制限がリプレイ時に発生した場合や、リプレイによってクライアントの表示可能な状態を、クライアントが現時点までに決定した可能性のある状態にリストアできない場合にリカバリ可能なエラー。

Oracle Database 12cリリース1 (12.1.0.1)で導入されたアプリケーション・コンティニュイティにより、Oracleデータベースを使用するシステムおよびアプリケーションのフォルト・トレランスが強化されます。

この章では、Oracle WebLogic Server、Oracle RACまたはOracle Active Data Guard (Oracle ADG)など、アプリケーション・コンティニュイティを使用するテクノロジまたは製品環境に関連する主な概念や技術をよく理解していることを前提としています。

この章の内容は次のとおりです。

- 計画外停止の管理
- 計画メンテナンスの管理
- アプリケーション・コンティニュイティについて
- アプリケーション・コンティニュイティの操作および使用
- アプリケーション・コンティニュイティの潜在的な副作用
- アプリケーション・コンティニュイティに関する制限および他の考慮事項
- クライアント・フェイルオーバーを向上させるためのトランザクション・ガード
- 透過的アプリケーション・フェイルオーバーによるOCIクライアントのフェイルオーバー

高速アプリケーション通知

Oracle RAC高可用性フレームワークは、データベースとそのサービスを監視し、高速アプリケーション通知(FAN)を使用してイベント通知を送信します。

Oracle Databaseでは、最も高いサービス可用性の保持に焦点を当てています。Oracle RACのサービスは、1つ以上のインスタンス間で負荷を共有しながら、継続的に使用できるように設計されています。Oracle RACの高可用性フレームワークでは、Oracle Clusterwareとリソース・プロファイルを使用して、サービスの可用性を維持します。Oracle Clusterwareは、ビジネス・ルールとサービス属性に従って、サービスをリカバリするか、サービスを均等に分散します。

この項には次のトピックが含まれます:

- 高速アプリケーション通知の概要
- 高速アプリケーション通知の高可用性イベント
- 高可用性イベントのサブスクリプション
- 高速アプリケーション通知のコールアウトの使用方法

関連項目

- Oracle RACのクライアントの有効化

高速アプリケーション通知の概要

FANは、データベース、ノードおよびネットワークに関連する停止後に、クライアントの即時割込みを実現します。

FANは、障害直後のTCP/IPタイムアウトからクライアントを解放するために欠かせないものです。FANは、リソースが使用可能になると即座にクライアントに通知して、クライアントが計画メンテナンス中の停止を認識しないようにデータベース・セッションの排出を開始します。また、FANには構成レベルの情報とサービス・レベルの情報の通知も含まれています。この情報には、サービス・ステータスの変化が含まれます。

Oracleクライアント・ドライバおよびOracle接続プールは、FANイベントに応答し、ただちに処理します。FANのUPイベントとDOWNイベントは、サービス、データベース、インスタンス、ネットワーク、およびノードに適用されます。

ノート:

FAN は、Oracle Database 10g リリース 2 (10.2)以降でサポートされます。

たとえば、Oracle接続プールは、FANを使用して、障害についての非常に高速な通知を受信し、障害後の接続を均等に分散します。障害が発生したコンポーネントが修復されると、接続を再度均等に分散します。そのために、インスタンスのサービスが起動されると、接続プールはFANイベントを使用して、そのリソースに作業を即座にルーティングします。インスタンスまたはノードのサービスが失敗すると、接続プールはFANイベントを使用して、リカバリのためにアプリケーションを即座に中断します。FANは、TCP/IPタイムアウトでアプリケーションがハングしないようにするために不可欠です。

FANの重要性

アプリケーションは、次のような重要な局面で時間を浪費する可能性があります。

- ソケットをクローズしないでノードが失敗した場合のTCP/IPタイムアウトの待機、およびそのIPアドレスが停止している間の後続のすべての接続の待機。
- サービスが停止した場合の接続の試行。
- サービス再開時に接続しない。
- サーバーが停止したときのクライアントでの最後の結果の処理。
- 最適でないノードで作業を実行しようとする。

ソケットをクローズしないでノードが失敗した場合、I/O待機(読取りまたは書込み)でブロックされたすべてのセッションは tcp_keepaliveを待機します。この待機ステータスは、ソケットで接続されているアプリケーションの場合の典型的な状況です。最後の結果を処理するセッションの状況はさらに悪く、次のデータが要求されるまで割り込みを受信しません。FANイベントを使用すると、TCPタイムアウトを待機しているアプリケーション、障害の発生後にクライアントで最後の結果を処理するという時間の浪費、および低速なノード、停止したノードまたは使用不能なノードで作業を実行するという無駄な時間が排除されます。

クラスタの構成が変更されて、クラスタ内で状態の変更が発生した場合、Oracle RACの高可用性フレームワークは、ただちにFANイベントを発行します。アプリケーションは、データベースに対するタイムアウトおよび問題の検出を待たずに、FANイベントを受信して即時に対応できます。FANを使用すると、インフライト・トランザクションがただちに終了し、インスタンスの失敗時にクライアントに通知されます。

また、FANは、 ロード・バランシング・アドバイザ・イベントも発行します。アプリケーションでFANのロード・バランシング・アドバイザ・イベントを使用して、現在、クラスタ内で最適なサービス品質を提供しているインスタンスに作業要求を割り当てることができます。

Oracle Database 12c リリース2 (12.2)クライアント・ドライバは、FAN対応であり、デフォルトでFANが有効化されています。これには、JDBC Thinドライバ(12.2.0.1)とOracle Data Provider for Net (ODP.NET)ドライバが含まれます。クライアント・ドライバは、計画および計画外のFANイベントを検出して、アプリケーションの下でアクションを実行できます。

計画メンテナンスでアプリケーションがOCIまたはPro*使用している場合(およびOCIセッション・プールやTuxedoを使用していない場合)、アプリケーションはOCI_ATTR_SERVER_STATUSをチェックする必要があります。このチェックは、独自の接続プールにセッションが返されたときに追加します。また、アイドル接続に対しては定期的に、このチェックを追加します。

計画メンテナンスによるFANのDOWNイベント後、この属性はOCI_SERVER_NOT_CONNECTEDに設定されます。アプリケーション

は、この切断ステータスを読み取ると接続を閉じます。セッションは、アクティブな作業の排出のために、アプリケーションが閉じるまで開かれたままになります。これにより、エラーの発生しないフェールオーバーを実現します。

FANイベントは、次の方法で利用できます。

- Oracle統合クライアントを使用すると、プログラムを変更することなくアプリケーションでFANを使用できます。FANイベント用の統合クライアントには、Oracle JDBC Universal Connection Pool、ODP.NET接続プール、OCIセッション・プール、Oracle WebLogic Server Active Gridlink for Oracle RAC、OCIおよびODP.NETクライアントがあります。FANの高可用性イベントを活用するには、統合Oracleクライアントが Oracle Database 10gリリース2 (10.2)以上である必要があります。プール・クライアントは、ロード・バランシング・アドバイザFANイベントを活用することもできます。
- サード・パーティ製アプリケーション・コンテナ(Apache TomcatやWebSphereなどのコンテナ)は、デフォルトのプールのかわりにユニバーサル接続プール使用することで提供される組込みのFANサポートを使用するように構成できます。ユニバーサル接続プールは、Apache TomcatやWebSphereなどのサード・パーティ製Javaアプリケーション・サーバーの接続プールとして動作保証されています。
- サード・パーティ製アプリケーション・サーバーまたはカスタム・アプリケーションで使用中のサード・パーティ製接続プールからのgetまたはreleaseの接続を標準インタフェースを使用してテストするために、OracleドライバのFAN対応機能を使用します。
    - このソリューションは、標準のTNS接続文字列の使用と、アプリケーションのCLASSPATHでons.jarファイルとsimpleFAN.jarファイルが使用できるようにすることで、標準のJavaアプリケーションに適用されます。
    - OCI/OCCIドライバの場合、OCI_ATTR_SERVER_STATUSサーバー・コンテキスト・ハンドル属性はFANイベントに依存し、接続がFANイベントの影響を受けている場合には、OCI_SERVER_NOT_CONNECTEDを返します。
- サーバー側のコールアウトを使用して、データベース層にFANを実装できます。
- JDBCおよびOracle RAC FANアプリケーション・プログラミング・インタフェース(API)を使用するか、OCIおよびODP.NETとともにコールバックを使用してFANイベントをサブスクライブし、イベントの受信時にイベント処理アクションを実行することで、アプリケーションはFANをプログラムで使用できます。

前述のリストの最初の項目に示されている統合クライアントのいずれかを使用すると、DOWNイベントの場合、FAN対応クライアントが失敗したインスタンスまたはノードへの接続をそれらが再使用される前に終了するため、アプリケーションの停止が最小限になります。アクティブな作業は、完了できるようになり、稼働を続けるインスタンスがある場合は、進行中の作業のためにサービスの継続が維持されます。インスタンスまたはサービスの停止時にアクティブなセッションは停止され、アプリケーション・ユーザーに即座に通知されます。未完了のトランザクションは、アプリケーション・コンティニュイティによって保護されます(有効化されている場合)。接続を要求しているアプリケーション・ユーザーは、使用可能インスタンスにのみ割り当てられます。

UPイベントでは、サービスおよびインスタンスが起動されている場合、アプリケーションが追加のハードウェア・リソースまたは追加容量を即時に利用できるように、新しい接続が作成されます。

前述のリストで説明したように、ドライバでFAN対応機能を利用するには、次の事項が必要になります。

- Java Thinドライバの場合、リリース12.2以降のFANは、CLASSPATHにons.jarファイルとsimpleFAN.jarファイルを配置して、推奨されるTNSフォーマット(例6-1を参照)を使用すると自動的に有効化されます。推奨されるTNSフォーマットを使用して、自動的にONSを構成します。また、Java Thinドライバでは、計画イベントと計画外イベントの両方でFANがサポートされます。計画外停止の場合、FAN割込みが即時に発生します。計画メンテナンスの場合、Javaアプリケーション・サーバーまたはカスタム・プールを構成して、サード・パーティ製接続プールからのgetまたは

releaseの接続をテストするために標準インタフェースを使用します。たとえば、アプリケーション・サーバーに応じてTestConnectionsOnReserve、TestOnBorrow、またはPreTestの接続になります。

このアプローチでは、計画メンテナンス中にFANイベントが受信されると、この時点ではアプリケーションにデータベースへの接続がないため、高速接続フェイルオーバーがセッションを閉じて(セッションがテストされている場合)、新しい接続の再試行が可能になります。接続テストには、isValid、isClosed、isUsable、PingDatabase、またはヒント/*+ CLIENT_CONNECTION_VALIDATION */が先行するSQL文を使用できます。

- SQLテストの場合、SQL構文はヒント/*+ CLIENT_CONNECTION_VALIDATION */で始まっている必要があります。SQLコマンドの実行時、ドライバは接続を排出します(次回の計画メンテナンスで影響を受ける場合)。接続プール、データソース、およびカスタマ・アプリケーション(プログラムの場合)のすべては、SQLコマンドの実行時に発生する回復可能なエラー(多くの場合、物理接続を閉じるエラー)を管理できるようにしておく必要があります。

ノート:

SQL ヒントは、SQL 文字列内でコメント以外の最初のトークンとして配置して、現在のドライバ・ベースの SQL 解析が変更されないようにする必要があります。たとえば:

```
/*+ CLIENT_CONNECTION_VALIDATION */ SELECT 1 FROM DUAL;
```

- サード・パーティのJavaアプリケーション・サーバーとJavaアプリケーションは、接続プールの開発時にPooledConnection標準インタフェースを使用できます。
- OCI/OCCIドライバの11.2.0.3リリースから、OCI_ATTR_SERVER_STATUSサーバー・コンテキスト・ハンドル属性がOCI_SERVER_NOT_CONNECTEDを返すときには、アプリケーションは接続を終了する必要があります。計画メンテナンスの場合は作業が排出されます。ドライバの12.2リリースは、計画DOWNイベントを受信したときに、OCISessionReleaseとOCIRequestEndを検出することもできます。

FANコールアウトは、サーバー側スクリプトであるか、またはFANイベントが生成されると必ず実行される実行可能ファイルです。様々なことを行うためのコールアウトを設計および構築できます。たとえば:

- ステータス情報のログへの記録
- リソースの起動に失敗した場合に、DBAへの通知またはサポート・チケットの発行を行います。
- サービスと同じ場所に配置する必要がある外部依存アプリケーションの自動的な起動
- ノードの障害などによって、ポリシー管理データベースに使用できるインスタンスの数が減少した場合、リソース・プランを変更するか、またはサービスを停止します。
- 必要に応じて、サービスを管理者管理データベースの優先インスタンスに自動的にフェイルバックします。

FANイベントは、Oracle Notification Serviceとアドバンスト・キューイングを使用して発行され、後者はOracle Databaseの前のリリースとの下位互換性のために続行されます。発行メカニズムは、Oracle RACのインストール時に自動的に構成されます。Java JDBC Thin接続を使用している場合、データベース接続からデータベース・サーバーのOracle Notification Service構成を取得することで、クライアントは自動的にOracle Notification Serviceに対応するように構成されます。クライアント側でOracle Notification Serviceを構成する必要はありません。

Oracle Net Servicesリスナーおよびグローバル・データ・サービス(GDS)は、FANイベントと統合されていることによって、リスナーおよびGDSは、障害が発生したインスタンスによって提供されているサービスを即座に登録解除でき、また、障害が発生したインスタンスに対する誤った接続リクエストの送信を回避できます。

サービスの接続時ロード・バランシングの目標にCLB_GOAL_SHORTを指定している場合、リスナーは、接続時ロードを均等に分散する際に、ロード・バランシング・アドバイザを使用します。ロード・バランシング・アドバイザが使用可能であった場合、リスナーで使用されるメトリックはさらに詳細に制御できます。

関連項目

- 計画メンテナンスの前のサーバーの排出

高速アプリケーション通知の高可用性イベント

この項では、FANイベントでコールアウト・プログラムに配信される情報について説明します。

FANイベントのタイプは、次の例でリスト表示されます。表6-1では、各イベント・パラメータの名前と値のペアについて説明します。次の例に示すとおり、コールアウトを介してFAN情報を受信した場合、イベント・タイプは常に最初のエントリです。

```
SERVICEMEMBER VERSION=1.0
   service=test.company.com database=ractest
   instance=ractest11 host=ractest1_host0343_1 status=up reason=FAILURE
   timestamp=2018-05-08 22:06:02 timezone=-07:00 db_domain=company.com
```

前述の例は1つの行として表示される点に注意してください。

FANイベントのタイプは、次のとおりです。

- DATABASE
- INSTANCE
- NODE
- SERVICE
- SERVICEMEMBER
- SERVICEMETRICS

DATABASEとINSTANCEタイプは、デフォルト・データベース・サービスをDB_UNIQUE_NAMEとしてリストします。

NODEイベントを除くすべてのイベントにdb_domainフィールドが含まれています。

SERVICEMETRICSタイプのイベントは、ロード・バランシング・アドバイザのイベントです。

関連項目: ロード・バランシング・イベントの詳細は、表5-1を参照してください。

表6-1 イベント・パラメータの名前/値のペアと説明

パラメータ	説明
VERSION	イベント・レコードのバージョン。リリースの変更を識別するために使用されます。
database	サービスをサポートする一意のデータベース名。DB_UNIQUE_NAME の初期化パラメータ値に該当し、これにより、DB_NAME 初期化パラメータの値がデフォルトに設定されます。
instance	サービスをサポートしているインスタンスの名前であり、ORACLE_SID の値と一致します。
host	サービスをサポートしているノードまたは停止したノードの名前であり、Cluster

パラメータ	説明
	Synchronization Services(CSS)で認識されるノード名と一致します。
service	サービス名。DBA_SERVICES でリストされているサービスの名前に一致し、適宜修飾されたドメインです。次の例を参照してください。 SERVICEMEMBER VERSION=1.0 service=swingbench database=orcl instance=orcl_2 host=rwsbj13 status=up reason=USER card=1 timestamp=2018-05-29 17:26:37 timezone=-07:00 db_domain= SERVICEMEMBER VERSION=1.0 service=swingbench.example.com database=orcl instance=orcl1 host=rwsbj09 status=up reason=USER card=2 timestamp=2018-05-03 17:29:28 timezone=-07:00 db_domain=example.com SERVICEMEMBER VERSION=1.0 service=swingbench.example.com database=orcl instance=orcl2 host=rwsbj10 status=up reason=USER card=1 timestamp=2018-07-03 17:29:18 timezone=-07:00 db_domain=example.com
status	値は、UP、DOWN、NODEDOWN(これは NODE イベント・タイプにのみ適用される)、NOT_RESTARTING および UNKNOWN です。 ノート: ● ノードが停止している場合、ステータスは NODEDOWN です。他のイベント・タイプの場合は DOWN です。 ● STATUS=NODEDOWN および REASON=MEMBER_LEAVE の場合、ノードに障害が発生しており、ノードはクラスタの一部ではないか、ユーザーがノードを停止しています。 ● STATUS=NODEDOWN および REASON=PUBLIC_NW_DOWN の場合、ノードは起動していますが、障害またはユーザー・アクションによりパブリック・ネットワークが停止しているため、ノードは使用不可です。 ● 複数のパブリック・ネットワークが Oracle Clusterware でサポートされています。FAN イベントでは、この事実が反映されています。
reason	AUTOSTART、BOOT、DEPENDENCY、FAILURE、MEMBER_LEAVE、PUBLIC_NW_DOWN、USER。 ノート: ● DATABASE および SERVICE イベント・タイプの場合の REASON=AUTOSTART は、ノードの起動時に、AUTO_START リソース属性がリストアに設定されており、ノードの起動前にリソースがオフラインであったことを示します。 ● DATABASE および SERVICE イベント・タイプの場合の REASON=BOOT は、ノードの

パラメータ	説明
	起動時に、リソースがノードの起動前からオンラインであったためリソースが起動したことを示します。 ● SRVCTL および Oracle Enterprise Manager 操作の場合、REASON=USER はそのような操作の計画済アクションを作業の排出として記述します。
cardinality	現在アクティブなサービス・メンバーの数。すべての SERVICEMEMBER UP イベントに含まれます。 次は、SERVICEMEMBER UP イベントの例です。 SERVICEMEMBER VERSION=1.0 service=swingbench.example.com database=orcl instance=orcl_2 host=mjkbj09 status=up reason=USER card=1 timestamp=2018-07-12 14:46:46 timezone=-07:00 db_domain=example.com
incarnation	NODEDOWN イベント用の新しいクラスタ・インカネーション。この値は、メンバーがクラスタに参加したり、クラスタから離脱するたびに変更されます。 次に、NODEDOWN イベントの例を示します。 NODE VERSION=1.0 host=stru09 incarn=175615351 status=down reason=member_leave timestamp=27-Jul-2018 14:49:32 timezone=-07:00
timestamp	Oracle Clusterware に基づく、イベントが発生した時間。
timezone	GMT +/-hh:mm として指定され、イベントが発生した、Oracle Clusterware のタイム・ゾーン。

表6-2に示すように、FANイベント・レコード・パラメータのいくつかには、デフォルトのネームスペースUSERENVを使用しているSYS_CONTEXTファンクションによって戻される値に対応する値があります。

表6-2 FANパラメータおよび該当するセッション情報

FANパラメータ	該当するセッション情報
SERVICE	sys_context('userenv', 'service_name')
DATABASE_UNIQUE_NAME	sys_context('userenv', 'db_unique_name')
INSTANCE	sys_context('userenv', 'instance_name')
CLUSTER_NODE_NAME	sys_context('userenv', 'server_host')

高可用性イベントのサブスクリプション

Oracle RACは、FANを使用して、構成の変更や、サービスで使用可能な各インスタンスが提供している現在のサービス・レベ

ルをアプリケーションに通知します。OCIクライアントまたはODP.NETクライアントを使用してFANイベントを受信している場合は、SRVCTLに-notificationパラメータを使用して、そのクライアントが使用するサービスを有効にして、アラート通知キューにアクセスする必要があります。

高速アプリケーション通知のコールアウトの使用

高速アプリケーション通知(FAN)コールアウトは、高可用性イベントが発生したときにOracle RACで即座に実行されるサーバー側のプログラム・ファイルです。

FANコールアウトを使用すると、クラスタ構成でイベントが発生した場合に、次のようなアクティビティを自動的に実行できます。

- 障害追跡チケットのオープン
- ページャへのメッセージの送信
- 電子メールの送信
- サーバー側のアプリケーションの起動および停止
- 発生時の各イベントのロギングによるアップタイム・ログのメンテナンス
- 優先度の高いサービスがオンラインになった場合の優先度の低いサービスの再配置

FANコールアウトを使用するには、Oracle Clusterwareを実行しているすべてのノードのGrid_home/racg/usrcoディレクトリにプログラム・ファイルを配置します。プログラム・ファイルは、別のプログラムからオプションの引数でコールされた場合、スタンドアロンで実行可能である必要があります。次に、Grid_home/racg/usrcoディレクトリに配置されている、callout.shという名前のシェル・スクリプトの例を示します。

```
#! /bin/bash
FAN_LOGFILE= [your_path_name]/admin/log/'hostname'_uptime'.log
echo $* "reported="'date' >> $FAN_LOGFILE &
```

前述の例では、FANイベントが生成されるたびに、シェル・スクリプトの$FAN_LOGFILEによって示されるログ・ファイルに次のようなエントリを追加します。

```
NODE VERSION=2.0 host=my-exa status=nodedown reason=public_nw_down
incarn=0 timestamp=2019-10-24  09:02:35 timezone=+00:00 vip_ips=10.1.1.94
```

FANイベント・レコードの内容は、データベースにログオンしているユーザーの現行のセッションに一致します。また、Oracle Call Interface (OCI)接続ハンドルと記述子属性(OCIAttrGet()を使用)を使用すると、ユーザー環境(USERENV)情報も使用できます。この情報を使用すると、FANイベントのデータに該当するセッションでアクションを実行できます。

通常、イベントはそれが発生したノード上のユーザー・コールアウトにポストされるだけです。たとえば、node1のデータベースが停止した場合、コールアウトはnode1のみにポストされます。唯一の例外は、ノード停止とVIP停止イベントで、これらのイベントは、発生した場所にかかわらず、すべてのノードにポストされます。

関連項目

- 高速アプリケーション通知の高可用性イベント
- Oracle Call Interfaceプログラマーズ・ガイド

計画外停止の管理

サービスは、管理者管理のOracle RACデータベースの1つ以上のインスタンスに割り当てるか、またはポリシー管理データベースのサーバー・プールに割り当てることができます。

Oracle RACで障害が検出されると、Oracle Clusterwareは、障害が発生したコンポーネントを隔離して、依存するコンポーネントをリカバリします。サービスの場合、障害が発生したコンポーネントがインスタンスであった場合、Oracle Clusterwareはサービスのカーディナリティを維持しようとします。サービス定義によりフェイルオーバーが許可され、カーディナリティを維持するためにフェイルオーバーが必要な場合は、フェイルオーバーが発生します。

FANイベントはOracle Databaseアーキテクチャ内の様々なレベルで発生し、前のOCIクライアントとの下位互換性のためにOracle Notification Serviceおよびアドバンスト・キューイングを使用して発行されます。FANコールアウトは、FANイベントに応答してデータベース・サーバーで実行されるように記述することもできます。

ノート:

Oracle RAC のコールアウトの実行では、順序は保証されません。コールアウトは非同期で実行され、スケジュールは変動します。

障害ノードのサービスが停止すると、稼働を続けるノードからFANが発行されます。Oracle RAC環境内でサービスを提供するインスタンスの位置および数は、アプリケーションに対して透過的です。再起動およびリカバリは自動的に実行され、データベースのみでなく、リスナーやOracle Automatic Storage Management(Oracle ASM)プロセスなどのサブシステムも再起動されます。FANコールアウトを使用して、障害管理システムに障害を報告して、修復ジョブを起動できます。

アプリケーション開発者にとってデータベース・セッション(インスタンス、ノード、ストレージやネットワークなど関連するコンポーネント)の停止をマスクすることは複雑な作業です。その結果として、エラーとタイムアウトはユーザーにさらされます。これはユーザーの不満、生産性や機会の喪失につながります。FANとアプリケーション・コンティニュイティは連携して、停止後に影響を受けるデータベース・セッションの進行中の作業をリカバリすることで、ユーザーとアプリケーションから停止をマスクします。アプリケーション・コンティニュイティは、このリカバリをアプリケーションの下で実行します。これにより、アプリケーションは停止をリクエストの実行のわずかな遅延として認識するようになります。

関連項目

- アプリケーション・コンティニュイティについて
- Oracle Database Net Services管理者ガイド

計画メンテナンスの管理

アプリケーション・ユーザーに対するサービス中断を最小限に抑えるために、Oracle Real Application Clusters (Oracle RAC)には、サービスを再配置、無効および有効にするインタフェースが用意されています。

ユーザーを妨害しない計画メンテナンスの管理

FAN対応のOracleまたはOracle以外の接続プールにより制御された期間中のインスタンスから、またはOracle

Database18c以降はデータベース自体で、データベース・セッションを排出することをお薦めします。

データベース・セッションの排出は、アプリケーションを中断せずに作業を移行する最も安全な方法です。排出が接続テスト時およびリクエスト境界の外で発生した場合、これは100%適切です。既存の作業が完了すると、アプリケーションは中断せずに続行し、新しい作業が別のインスタンスで同じサービス機能のセッションを取得します。その結果、アプリケーションにはエラーが返されず、データベース・セッションの状態が不正になるリスクがありません。接続テストの場合、コール元は受信するリターン・コードが正しくても正しくなくても、結果を処理する準備ができているので、接続テストの検査が広範囲に適用可能で非常に強力なソリューションになります。

サービス属性の-drain_timeoutと-stopoptionでは排出期間を制御し、この期間の経過後に完了していないセッションをサービスで管理する方法を制御します。完了後にプールにチェックインするリクエストまたは完了後に閉じるリクエストは、計画メンテナンスの影響を受けない新しい場所に転送できます。

アプリケーション・コンティニュイティでは、割り当てられた排出時間内に完了しないリクエストにサービスを継続することで追加の対策を提供します。FAN対応のプールを使用すると、FANが計画したDOWNイベントの受信後に、セッションはリクエスト境界で排出できるようになります。

Oracle Database 18c以降、一部のアプリケーションがOracle接続プールを使用し、一部のアプリケーションがFAN対応であるため、データベースでは計画メンテナンス中にセッションを検査し、アプリケーションが中断されないようにセッションを停止する安全な場所を探します。サービスを停止した後、データベースは接続を閉じることができる安全な場所を検索します。接続が閉じると、データベースではセッションをクリーンアップします。

安全な場所でセッションを停止すると、アプリケーションは必要な状態で新規接続を開くことができます。セッションの排出では、各セッションに関係する作業が発生する場合があります。セッションを即座に閉じる必要はありませんが、可能であれば、排出のタイムアウト期間が経過する前にアプリケーションにエラーが表示されない安全な場所で閉じる必要があります。

発行済の作業はリクエストによって完了できるため、リクエストはトランザクションよりもはるかに重要です。Oracle Universal Connection Poolは、リクエストの排出の際に排出タイムアウトを使用して段階的な排出を実行します。元のセッションを一度に解放するのではなく、期間全体で徐々に解放することで、排出されるインスタンスのログインのオーバーロードを回避します。段階的な排出には、ターゲット・インスタンスで実行中の別の作業を妨害しないという利点があります。

DRAIN_TIMEOUTとSTOP_OPTIONは、サービスを追加するとき、または作成後のサービスを変更するときに定義できるサービスの属性です。これらの属性は、SRVCTLを使用して指定することもできます。このようにすると、サーバーで定義されているものよりも優先されます。次に示すSRVCTLコマンドを使用すると、-drain_timeoutパラメータと-stopoptionパラメータを指定できます。

- srvctl add service
- srvctl modify service
- srvctl relocate service
- srvctl stop service
- srvctl stop database
- srvctl stop instance

ユーザーを妨害することなく計画メンテナンスを管理するには:

1. SRVCTLを使用してシングルトン・サービス、またはすべてのノードで実行中でないサービスを再配置します。前述の各SRVCTLコマンド(addとmodifyを除く)に、-forceフラグを使用します。-forceフラグは、srvctl relocate serviceまたはsrvctl stop serviceのどちらかを実行するときに、コマンドラインで-stopoptionパラメータを指定した場合に使用する必要があります。たとえば:

```
$ srvctl relocate service -db mycdb01 -service myservice -drain_timeout 120
  -stopoption IMMEDIATE -oldinst mycdb01_01 -force
```

前述のコマンドは、mycdb01_01というインスタンスのmyservice01というサービスを、そのサービスを実行するように構成されたインスタンスに再配置します。Oracle Clusterwareは、このインスタンスを選択して(コマンドラインでターゲットを指定していない場合)、アクティブなセッションを排出するために2分間待機し(この例の場合)、その後でmycdb01_01に残っているセッションが強制的に切断されます。接続プールにより、要求境界で接続が自動的に解放されます。

ノート:

再配置するサービスが現在すべてのノードで実行中の均一サービスである場合、前述のコマンドはエラーを返し、サービスがすべてのインスタンスで起動していなければ、均一サービスの場合に前述のコマンド例は成功します。

2. FAN計画済DOWNイベントにより、アイドル・セッションが接続プールからただちにクリアされ、次のチェックインで解放されるアクティブ・セッションがマークされます。これらのFANアクションにより、ユーザーの作業を妨げずにセッションがインスタンスから排出されます。

   他のインスタンスの既存の接続は使用可能なままで、必要な場合はこれらのインスタンスに新しい接続をオープンできます。排出するセッションには、データベースによってマークも付けられます。データベースは、接続テストと安全なフェイルオーバー先(Oracle Database 19c以降の場合)を探します。透過的アプリケーション・コンティニュイティでの暗黙的な接続境界は、このような場所です。

3. どの場合でも、すべてのセッションが接続をプールにチェックインするわけではありません。ベスト・プラクティスとして、タイムアウト期間を設定して(-drain_timeoutパラメータで設定)、その後で、残っているクライアント接続を削除するためにインスタンスがシャットダウンされるようにするか、サービスが停止されるようにすることをお薦めします。
   排出間隔の経過後、-stopoptionパラメータが実施されます。このパラメータは、サービスまたはデータベースに対して次のように定義できます。
   - サービスを停止する場合(srvctl stop service)、-stopoptionパラメータを使用すると停止オプションのTRANSACTIONALまたはIMMEDIATEのいずれかを指定できます
   - データベースを停止する場合(srvctl stop database)、-stopoptionパラメータを使用すると停止オプションのNORMAL、TRANSACTIONAL、IMMEDIATE、ABORTのいずれかを指定できます

   データベースの停止オプションとサーバーの停止オプションの相関は次のとおりです。
   - NORMAL=NONE
   - TRANSACTIONAL/TRANSACTIONAL LOCAL=TRANSACTIONAL
   - IMMEDIATE/ABORT=IMMEDIATE

   アプリケーション・コンティニュイティを使用するように構成されたサービスの場合、ユーザーとアプリケーションから停止をマスクするために、終了された後で残っているセッションのリカバリが試行されます。

4. メンテナンスが完了後に、元のノードでインスタンスとサービスを再起動します。
5. サービスのFAN UPイベントにより、新しいインスタンスが使用可能で、次の要求境界でこのインスタンス上にセッションを作成できることが接続プールに通知されます。

関連項目

- アプリケーション・コンティニュイティについて
- 計画メンテナンスの前のサーバーの排出

メンテナンスのためのサービスのグループの管理

多くの企業が多数のサービスを実行しています。多数のサービスが単一のデータベースまたはインスタンスで提供されることも、多数のデータベースが同じノード上で実行する少数のサービスを提供することもあります。

個別のサービスごとにSRVCTLコマンドを実行する必要はなくなり、影響を受けるすべてのサービスのノード名、データベース名、プラガブル・データベース名、またはインスタンス名を指定することのみが必要になります。

- たとえば、特定のノードで実行しているすべてのサービスを停止する場合は、次に示すコマンドを使用できます。

  ```
  $ srvctl stop service -node racnode01 -drain_timeout 60 -stopoption IMMEDIATE
  ```

  このコマンドでは、60秒の排出間隔を割り当てて、racnode01で実行しているすべてのサービスを停止します。60秒後に、残っているセッションは即座に停止されます。この60秒の排出タイムアウト間隔は、あらゆるサービスの属性設定をオーバーライドします。

  このコマンドは、次の例に示すように、ノード上のデータベースを停止する場合にも適しています。

  ```
  $ srvctl stop instance  -node racnode01 -drain_timeout 60 -stopoption TRANSACTIONAL
    LOCAL -failover -force
  ```

  -failoverパラメータを指定すると、次のようになります。
  - すべてのサービスは、指定した排出タイムアウト間隔と停止オプションを考慮して再配置されます(可能な場合)。
  - フェイルオーバーできないサービスは、指定された停止オプションを使用して停止されます。
  - 排出タイムアウト間隔の経過まで待機するか、ターゲットのサービスのセッションがすべて削除されるまで待機します(どちらか早いほう)。
  - すべてのインスタンスは、停止オプションの指定に従って停止します。

  -stopoption TRANSACTIONAL LOCALパラメータを指定すると、次のようになります。
  - 残っているサービスは、指定された排出タイムアウト間隔と停止オプションに従って停止します。
  - 排出タイムアウト間隔の経過まで待機するか、ターゲットのサービスのセッションがすべて削除されるまで待機します(どちらか早いほう)。
  - インスタンスは、TRANSACTIONAL LOCAL停止オプションを使用して停止します。

この項には次のトピックが含まれます:

- サービスの開始
- プラガブル・データベース・レベルの操作
- サービスの再配置
- サービスの停止

サービスの開始

srvctl start serviceコマンドを使用すると、ノード上のすべてのサービス、データベースが提供するすべてのサービス、プラガブル・データベースが提供するすべてのサービス、またはインスタンス上や特定のサーバー・プール内で提供されるすべてのサービスを開始できます。

また、srvctl start serviceコマンドには、開始するサービスのリスト(すべてのサービスのサブセット)を指定することもできます。さらに、特定のノードで開始できるすべてのサービスに対して、データベース・オプションとともにノード制限を指定することもできます。srvctl start serviceコマンドは、-pqパラメータを指定することで、パラレル問合せサービスのみを開始するように制限できます。
次の各例では、サービスの開始方法を説明します。

- 単一のプラガブル・データベースが提供するサービスをすべて開始するには:

  ```
  $ srvctl start service -db myRACCDB01 -pdb myPDB01 -startoption OPEN
  ```

  特定のデータベースと、そのプラガブル・データベースのサービスをすべて開始するには:

  ```
  $ srvctl start service -db myRACDB
  ```

  関連付けられたプラガブル・データベースの有無にかかわらず、特定のデータベースのサービスのリストを開始するには:

  ```
  $ srvctl start service -db myRACDB -service "myFirstService,mySecondService,myThirdService"
  ```

  特定のノードで実行可能なデータベースのサービスをすべて開始するには:

  ```
  $ srvctl start service -d myRACDB -node racnode01
  ```

プラガブル・データベース・レベルの操作

SRVCTLを使用すると、プラガブル・データベースのサービスを管理できます。

- すべてのインスタンスまたは単一のインスタンスに対して、プラガブル・データベースのサービスをすべて開始するには:

  ```
  $ srvctl start service -db db_name -pdb pdb_name [-instance instance_name]
  ```

- すべてのインスタンスまたは単一のインスタンスに対して、プラガブル・データベースのサービスをすべて停止するには:

  ```
  $ srvctl stop service -db db_name -pdb pdb_name [-node node_name | -instance
    inst_name | -serverpool pool_name] [-stopoption stop_option] [-drain_timeout timeout]
    [-force [-noreplay]]
  ```

ノート:

-pdb pdb_name パラメータはオプションです。プラガブル・データベース名を省略すると、コンテナ・データベース全体(このコンテナ内のすべてのプラガブル・データベース)が操作の対象になります。

サービスの再配置

srvctl relocate serviceコマンドを使用すると、ターゲット宛先にサービスを再配置できます。インスタンス、ノードまたはデータベースが再配置の宛先になります。

次のコマンド例では、すべてのサービスが、名前付きデータベース、プラガブル・データベース、インスタンス、またはノードから再配

置されます。サービスは、サービス構成で定義されているとおりに、そのサービスをターゲットがサポートできる場合にのみ再配置されます。再配置できないサービスは、元の場所に残されます。再配置されていないサービスに対する配置エラーが記録されます。それ以外は、すでに新しいターゲットで実行されています。再配置に失敗したサービスは、そのサービスの元の場所で実行を続け、セッションはアクティブのままになります。

```
$ srvctl relocate service -db myRACCDB -oldinst RACCDB_01 -newinst RACCDB_03
  -drain_timeout 30 -stopoption immediate
```

または

```
$ srvctl relocate service -db myRACCDB -pdb myPDB01 -currentnode racnode01
  -targetnode racnode02 -drain_timeout 30 -stopoption immediate
```

再配置操作は、新しい場所でサービスを開始してから、既存の場所でサービスを停止します。

ターゲット宛先を指定していない場合、Oracle Clusterwareは、指定されたデータベース、プラガブル・データベース、インスタンス、またはノードから、すべてのサービスまたは特定のサービスを再配置します。次に、例を示します。

```
$ srvctl relocate service -db myRACCDB -service "myService01,myService02"
  -drain_timeout 30 -stopoption immediate
```

または

```
$ srvctl relocate service -db myRACCDB -pdb myPDB01 -drain_timeout 30
  -stopoption transactional
```

有効なターゲットが存在しない場合、サービスは元の場所に残され、セッションはアクティブのままになります。サービスを調べて、必要な場合はサービスを停止してください。

サービスを再配置すると、サービスは新しい場所で開始されてから、元の場所で停止されます。Oracle Clusterwareは、新しいインスタンスまたはプラガブル・データベースを依存性として開始できます。-drain_timeoutパラメータと-stopoptionパラメータが指定されていると、サービスの属性がオーバーライドされます。

サービスの停止

srvctl stop serviceコマンドを使用すると、ノード上のすべてのサービス、データベースが提供するすべてのサービス、プラガブル・データベースが提供するすべてのサービス、またはインスタンス上や特定のサーバー・プール内で提供されるすべてのサービスを停止できます。

srvctl stop serviceコマンドには、停止するサービスのリスト(すべてのサービスのサブリスト)を指定することもできます。さらに、-pqパラメータを指定することで、パラレル問合せサービスのみを停止するようにsrvctl stop serviceコマンドを制限することもできます。
次の各例では、サービスの停止方法を説明します。

- 単一のプラガブル・データベースが提供するサービスをすべて停止するには:

  ```
  $ srvctl stop service -db myRACCDB01 -pdb myPDB01 -drain_timeout 15 -stopoption TRANSACTIONAL
  ```

  特定のデータベースと、そのプラガブル・データベースのサービスをすべて停止するには:

  ```
  $ srvctl stop service -db myRACDB -drain_timeout 15 -stopoption IMMEDIATE
  ```

データベースが提供する一部のサービスのみを停止するには:

```
$ srvctl stop service -db myRACDB -service "myFirstService,mySecondService,
  myThirdService" -drain_timeout 60 -stopoption IMMEDIATE
```

ノート:

SRVCTL コマンドライン・パラメータの-wait YES を使用すると、-stopoption パラメータは排出タイムアウト間隔を経過するまで実施されなくなります(この間隔の完了前に、すべてのセッションが終了していても実施されません)。

計画メンテナンスの前のサーバーの排出

計画メンテナンスの前に、アプリケーションの動作が中断されないように、データベース・インスタンスでデータベースのセッションを排出またはフェイルオーバーします。Oracle Database 18c以降、データベース自体がセッションを排出します。

計画メンテナンスの準備を行う場合は、サーバー・インフラストラクチャを使用しているサービスを停止または再配置する必要があります。サービスの再配置は計画済停止の前に一定期間にわたって行われ、各サービスに関連する作業の性質に基づいています。

計画メンテナンスのローリングの手順では、メンテナンスの前にサービスを別のデータベース・インスタンスに移動し、クライアント側ドライバ、接続プール、データベース・インスタンス自体および他のサブスクライバにメンテナンスが保留中であることと、排出する必要のあるもの(このサービスを使用する接続またはセッション)を通知します。排出が通知されると、高速アプリケーション通知(FAN)イベントが送信され、クライアント・プールは他の場所で説明されているように動作し、さらにデータベースでは接続を解放する安全な場所を検索し、必要に応じて接続を移行します。

サービスを移動または停止すると、FAN通知がトリガーされ、サブスクライブしているOracleドライバおよびOracle接続プールで受信されます。Oracle Database 18c以降では、FAN通知によってもサーバーでのセッションの排出がトリガーされます。そのサービスに対する新規の作業が、ただちにサービスの機能している別のインスタンスに送信されます。既存のセッションは、その作業の完了後に解放用にマークされます。作業が完了して接続が接続プールに戻されると、Oracleドライバまたは接続プールのいずれかがこれらのセッションを終了します。

データベースでのセッションの排出

OLTPアプリケーション、アプリケーション・サーバーおよびカスタム・アプリケーションがデータベース・セッションを流用および返却する専用の接続プールを持っている場合、データベース・セッションが流用されなければ、そのセッションの排出は安全です。Oracleサーバー・インフラストラクチャがセッションを閉じるのに最適なのは、アプリケーション・サーバーがその接続の妥当性をテストした時点です。流用および解放時に接続プール・マネージャが接続の妥当性をテストして、接続が有効はでないことが検出した場合、エラーはアプリケーションに返されません。

安全な場所とは、アプリケーションが中断されない場所です。接続プールの場合、これは流用(チェックイン)されていない接続を意味し、アプリケーションの場合、接続を流用または返却する時点において同様のことが当てはまります。この時点では、すべての作業が完了しているか、起動していないかのいずれかです。データベースでは、すべての状態がアプリケーションに対して透過的にリストアできる場合、接続をフェイルオーバーすることもできます。

Oracle Database 18c以降では、データベースはルールとヒューリスティックの拡張可能なセットを使用して、データベース・セッションを取り除くタイミングを検出します。排出が開始すると、データベース・セッションはルールが満たされるまでデータベースで継続します。ルールには次の内容が含まれています。

- 標準アプリケーション・サーバーが妥当性をテストします
- カスタムSQLが妥当性をテストします
- リクエスト境界は有効になっており、アクティブなリクエストはありません
- リクエスト境界は有効になっており、現在のリクエストは終了しています
- セッションにはリカバリ可能なセッション状態が1つ以上あり、フェイルオーバー時にセッションを再作成できます

ノート:

Oracle Database 18c 以降、接続を排出するには、「サーバーで排出するための接続テストの追加、無効化、有効化および削除」を参照してください。

たとえば、接続テストの場合、標準的な手順として、接続プールからの流用時、プールへの返却時およびバッチ・コミット時に、アプリケーション・サーバー、プールされたアプリケーション、ジョブ・スケジューラなどが接続をテストします。排出時に、データベースは接続テストを中断して接続を閉じ、テストの失敗ステータスを返します。接続テストを発行しているアプリケーション・レイヤーは、失敗の戻りステータスを処理する準備ができています。通常、さらにリクエストを発行して、別の接続を取得します。アプリケーションは中断されません。

すべてのセッションを排出できるわけではありません。接続がプールに戻っていないときや、FANが使用されていないときには排出できません。透過的アプリケーション・コンティニュイティまたはアプリケーション・コンティニュイティが有効化されている場合、サーバーは、アプリケーション・コンティニュイティがセッションをすばやくリカバリできるリクエスト境界を検出します。サーバーはセッションを中断する可能性がありますが、アプリケーション・コンティニュイティはこれを中断せずにリカバリします(Oracle RACクラスタの別のサーバーに対してなど)。

排出しないデータベース・セッションの場合、データベースはセッションを置換できるブレーク・ポイントを見つける必要があります。ブレーク・ポイントでは、状態が既知およびリカバリ可能である場合に、接続は透過的にフェイルオーバーできます。ブレーク・ポイントは、トランザクション境界、コールがリクエスト内で処理される前のリクエストの先頭(beginRequest)、およびリクエストが開始または終了していることを通知する監査コールなど、パターンである場合があります。ブレーク・ポイントは、状態がリストア可能であると認識されている場合にのみ適用されます。

接続をフェイルオーバーすると、アプリケーションによっては、アプリケーション・コンティニュイティ、透過的アプリケーション・コンティニュイティまたは透過的アプリケーション・フェイルオーバー(TAF)を有効にする必要があります。

ノート:

UCP または OCI セッション・プールなど、Oracle 接続プールは継続的な可用性を実現し、ロード・バランシングなどを提供することで大きな利点を提供するため、これらの接続プールを使用することをお薦めします。

サーバーで排出するための接続テストの追加、無効化、有効化および削除

サービス、プラガブル・データベースまたは非コンテナ・データベースにSQL接続テストを追加できます。

デフォルトでは、すべてのデータベース・サービスおよびプラガブル・データベース・サービスに4つのSQL接続テストが追加されています。したがって、アプリケーションが接続で次のSQL接続テストを使用している場合、これらを追加する必要はありません。

```
SELECT 1 FROM DUAL;
SELECT COUNT(*) FROM DUAL;
SELECT 1;
BEGIN NULL;END;
```

- サービスにサーバー側SQL接続テストを追加するには、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.add_sql_connection_test('select dummy from dual','sw_orcl');
  ```

  プラガブル・データベースまたは非コンテナ・データベースにサーバー側SQL接続テストを追加するには、非コンテナ・データベースにログオンし、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.add_sql_connection_test('begin null;end;');
  ```

  SQL接続テストを追加すると、デフォルトでこれが有効になります。

- SQL接続テストが不要な場合またはこれを使用していない場合、プラガブル・データベースまたは非コンテナ・データベースにログオンして、次のようなSQL文を使用することにより、SQL接続テストを無効にできます。

  ```
  SQL> execute dbms_app_cont_admin.disable_connection_test(dbms_app_cont_admin.sql_test,'select
  dummy from dual');
  ```

  デフォルトではpingテストおよび終了リクエスト・テストは無効になっていますが、これらを有効にした後に無効にする場合、次のSQL文のいずれかを使用できます。

  pingテストを無効にする場合、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.disable_connection_test(dbms_app_cont_admin.ping_test);
  ```

  終了リクエスト・テストを無効にする場合、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.disable_connection_test(dbms_app_cont_admin.endrequest_test);
  ```

- プラガブル・データベースまたは非コンテナ・データベースにログオンして次のようなSQL文を使用することにより、SQL接続テストを無効にした後これを有効にできます。

  ```
  SQL> execute dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.sql_test,'select
  dummy from dual');
  ```

  無効になっている場合、次のSQL文のいずれかを使用してpingテストおよび終了リクエスト・テストを有効にすることもできます。

  isValid、isUsable、OCIpingまたはconnection.statusなどのpingを使用するすべてのテストを実行する場合、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.ping_test);
  ```

  リクエストの終了時に排出を有効にする場合、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.endrequest_test);
  ```

  リクエストの終了時に排出を無効にする場合、次のようなSQL文を使用します。

  ```
  SQL> execute dbms_app_cont_admin.disable_connection_test(dbms_app_cont_admin.endrequest_test);
  ```

- SQL接続テストが不要な場合、プラガブル・データベースまたは非コンテナ・データベースにログオンして、次のようなSQL文を実行すると、SQL接続テストを削除できます。

```
SQL> execute dbms_app_cont_admin.delete_sql_connection_test('select dummy from
dual','sw_orcl');
SQL> execute dbms_app_cont_admin.delete_sql_connection_test('begin null;end;');
```

すべてのアプリケーション・サーバーには各接続プールで接続の妥当性をテストする機能があり、これは構成プロパティまたは管理コンソールで設定されています。テストの目的は、使用できない接続をアプリケーションに渡さないようにすることと、使用できない接続を検出した場合に、プールへの解放時にこれを削除することです。

様々なアプリケーション・サーバーでテストの名前が類似しています。提供されているテストでは様々なアプローチを使用しており、最も一般的なものはSQL文です。Javaアプリケーション・サーバーで標準のJavaコールconnection.isValidを使用することをお薦めします。Oracle Database 18c以降、これらのテストを使用してデータベースを排出します。また、Oracle Database 18c以降、データベースは安全な排出ポイントのためにセッションを調査することにより、FANを使用しないでセッションを排出します。

次の表は、より一般的ないくつかのアプリケーション・サーバーに利用可能な標準接続テストを示しています。

表6-3 一般的なアプリケーション・サーバーの標準接続テスト

アプリケーション・サーバー	**データベースへの接続テスト**
Oracle WebLogic サーバー	用意されているテストは次のとおりです。 ● dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.sql_test,'select 1 from dual'); ● TestConnectionsonReserve: isUsable、isValid、または PingDatabase ● サーバーの排出場合、TestConnectionsOnCreate (SQL 構文): Select 1 from dual;
Oracle WebLogic Server Active Gridlink	次のテストが埋め込まれています。isUsable
IBM WebSphere	dbms_app_cont_admin.enable_connection_test(dbms_app_cont.sql_test,'select 1 from dual'); サーバーの排出のために接続を事前テストします(SQL 構文)。 Select 1 from dual;
RedHat JBoss	check-valid-connection-sql (SQL 構文):dbms_app_cont_admin.enable_connection_test(dbms_app

アプリケーション・サーバー	データベースへの接続テスト
	_cont_admin.sql_test,'select 1 from dual');
Apache Tomcat	2 つのテスト testOnBorrow および testOnReturn を利用できます。どちらも、データベースへの接続テストに SQL 構文を使用します。dbms_app_cont.enable_connection_test(dbms_app_cont.sql_test,'select 1 from dual'); アプリケーション・サーバーでは次を使用します。Select 1 from dual;

Oracle Notification Services (ONS)の自動構成をサポートするために次に示すフォーマットの使用をお薦めします。これにより、FANイベントを受信できます(ONS経由)。

例6-1 FANの自動構成

```
alias =(DESCRIPTION =
 (CONNECT_TIMEOUT=90)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
   (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   ( ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
   (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   ( ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

関連項目

- ユーザーを妨害しない計画メンテナンスの管理
- 透過的アプリケーション・コンティニュイティ

アプリケーション・コンティニュイティについて

Oracle Databaseに付属するアプリケーション・コンティニュイティ機能により、データベースを使用するシステムおよびアプリケーションのフォルト・トレランスが向上します。

クライアント要求には、トランザクション処理や非トランザクション処理が含まれる場合があります。Oracle Databaseでリプレイが成功すると、アプリケーションはデータベース・セッションが中断された時点から処理を続行できるため、ユーザーは資金の振替や航空券の予約などの進行状況がわからない不安な状態に放置されることがなくなります。こうしたクライアント要求をリカバリすることで、アプリケーションがオンラインに戻ったときに、ログインのオーバーロードからのリカバリのために中間層サーバーを再起動する必要がなくなります。アプリケーション・コンティニュイティを使用すると、計画的な停止または計画外の停止の多くをマスクすることでエンド・ユーザーの使用感が向上します。アプリケーション開発者がリクエストをリカバリする必要はありません。

アプリケーション・コンティニュイティは、データベース・セッション(すべての状態、カーソル、変数、および存在する場合は最後のトランザクションを含むフル・セッション)をリストアすることで、アプリケーションとユーザーからリカバリ可能なOracle Databaseの停止の多くをマスクします(リプレイが成功した場合)。アプリケーション・コンティニュイティは、計画外停止または計画済メンテナンス(タ

イムアウト、ネットワーク停止、インスタンス障害、修復、構成変更、パッチ適用など)が原因で使用不可になったデータベースおよびデータベース・インスタンスにアプリケーションがアクセスしようとする際に発生する問題に対処します。アプリケーション・コンティニュイティが使用されていない場合、データベースのリカバリによってアプリケーションおよびエンドユーザーに対して停止がマスクされません。このような場合、開発者およびユーザーは例外条件に対処する必要が生じ、ユーザーは資金の振替、タイムシート、注文、請求書の支払などに何が起こったかわからないままになる可能性があります。コミットされていないデータの画面が消失し、ログインしなおしてデータの再入力が必要な場合もあります。最悪の場合、管理者は、大量のログインからリカバリするために中間層の再起動を迫られることになります。

アプリケーション・コンティニュイティを使用すると、データベース・インスタンスが使用不可になった場合、アプリケーション・コンティニュイティは正しい状態を使用してセッションおよび任意のオープン・トランザクションを再構築しようとします。トランザクションがコミットされていて、再送信する必要がない場合は、正常な戻りステータスがアプリケーションに戻されます。リプレイが成功した場合、重複のリスクなしにリクエストを安全に続行できます。アプリケーションですでに処理しており、決定したと思われるデータをリプレイによってリストアできない場合、データベースはリプレイを拒否し、アプリケーションは元のエラーを受け取ります。

アプリケーション・コンティニュイティは、進行中のトランザクションおよびデータベース・セッション状態のリカバリを実行しながら、トランザクション・ガードによって実現されるトランザクションの冪等性を確保します。各データベース・セッションには論理トランザクションID (LTXID)がタグ付けられているため、データベースは、リプレイごとにトランザクションがコミットされたかどうかのみでなく、トランザクションがコミットされた場合は処理が完了まで実行されたかどうかも識別します。アプリケーション・コンティニュイティがリプレイしようとしている間、アプリケーションはリプレイを遅延処理として認識するか、元のトランザクションに対するコミット・レスポンスを受け取ります(最後のトランザクションが停止前に完了していた場合)。

アプリケーション・コンティニュイティは、Oracle RACとOracle Active Data Guardでサポートされています。これは、マルチテナント・アーキテクチャを使用しているOracle Databaseで(プラガブル・データベース・レベルでのフェイルオーバーによって)サポートされます。現在、Oracle GoldenGate、ロジカル・スタンバイ、サード・パーティのレプリケーション・ソリューション、またはDMLリダイレクト(Oracle Active Data Guardを使用する場合)ではサポートされません。

関連項目

- アプリケーション・コンティニュイティの操作および使用

アプリケーション・コンティニュイティの主な概念

この項では、アプリケーション・コンティニュイティを使用するために理解する必要があるいくつかの用語や概念について説明します。

次の用語は、この章全体を通じて使用されています。

データベース・リクエスト

データベース・リクエストとは、アプリケーションからデータベースに送信された作業のユニット(トランザクションなど)です。一般に、リクエストは、単一のデータベース接続上の単一のWebリクエストのSQLとPL/SQL、および他のデータベース・コールに相当し、通常、接続プールのデータベース接続をチェックアウトおよびチェックインするためのコールによって区別されます。

リカバリ可能なエラー

リカバリ可能なエラーとは、実行中のアプリケーション・セッション・ロジックとは関係なく、外部システムの障害が原因で発生するエ

ラー(切断された接続や無効な接続など)です。リカバリ可能なエラーは、フォアグラウンド、ネットワーク、ノード、記憶域、データベースの計画済停止および計画外停止に続いて発生するエラーです。アプリケーションは、最後に発行された操作のステータスを把握しないままの状態で残される可能性があるエラー・コードを受信します。アプリケーション・コンティニュイティは、データベース・セッションを再確立し、リカバリ可能なエラーのクラスに対して保留されている作業を再発行します。

アプリケーション・コンティニュイティは、リカバリ不能なエラーが原因であるコール障害に続く作業は再発行しません。リプレイされないリカバリ不能なエラーの例には、無効なデータ値の発行があります。

コミット結果

トランザクションは、トランザクション表内のエントリを更新することによってコミットされます。Oracle Databaseは、この更新に対応するREDOログ・レコードを生成し、このREDOログ・レコードを書き出します。このREDOログ・レコードがディスク上のREDOログに書き出されると、トランザクションはデータベースでコミットされたとみなされます。クライアントの観点からは、REDOが書き込まれた後に生成されたOracleメッセージ(コミット結果と呼ばれます)をクライアントが受信した時点でトランザクションはコミットされたとみなされます。ただし、COMMITが発行されていた場合、クライアントまたはアプリケーションで受信されていないCOMMIT失敗のメッセージは取得できなくなります。

可変関数

可変関数とは、コールされるたびに新しい値を取得できる非DETERMINISTIC関数です。このため結果は頻繁に変化します。可変関数を使用すると、結果がリプレイ時に変化することがあるため、リプレイの問題が発生します。キー値でしばしば使用されるsequence. NEXTVALおよびSYSDATEについて検討してください。主キーがこれらのファンクション・コールの値を使用して構築され、後で外部キーまたは他のバインドで使用される場合、リプレイ時に同じファンクション結果が戻される必要があります。

アプリケーション・コンティニュイティは、付与されているOracleファンクション・コールに対してリプレイ時に可変オブジェクト値の置換を提供することにより、不透明バインド変数の一貫性を実現します。不変のデータベース・ファンクション(sequence. NEXTVAL、SYSDATE、SYSTIMESTAMPおよびSYSGUIDを含む)がコールに使用される場合、ファンクションの実行から戻される元の値が保存され、リプレイ時に再適用されます。

セッション状態の一貫性

COMMIT文が実行された後、このトランザクションで状態が変更された場合、セッションが失われたときにトランザクションをリプレイしてこの状態を再確立することはできません。アプリケーション・コンティニュイティの構成時には、初期設定後のセッション状態が静的と動的のどちらであるか(または自動的に決定されるようにAUTOを使用)、さらにリクエスト内のCOMMIT操作後の処理続行が正しいかどうかに応じて、アプリケーションは分類されます。

- セッション状態の変更が初期化によって不完全にカプセル化されていて、フェイルオーバー時にFAILOVER_RESTOREまたはコールバックで完全に取り込むことができない場合、セッションの状態は動的です。最初のトランザクションが完了した後、フェイルオーバーは次のリクエストが開始されるまでは内部的に無効化されます。セッション状態は、リクエストの過程で変化することがあります。
- セッション状態の変更(NLS設定やPL/SQLパッケージ状態など)がすべて初期化の一環として行われ、フェイルオーバー時にFAILOVER_RESTOREまたはコールバックでカプセル化できる場合、セッションの状態は静的です。静的アプリケーションとは、アプリケーション・コンティニュイティの前に透過アプリケーション・フェイルオーバー(TAF)を使用できるアプリケーションのことです。セッション状態は、リクエストの過程で変化しません。(可能な場合、自動モードでは事前アプリ

ケーション・コンティニュイティTAFモードよりも効率的にパージしてクリーンアップするため、STATICモードでセッション状態の一貫性をAUTOに設定することを選択します。)

- 透過的アプリケーション・コンティニュイティを使用すると、セッション状態の一貫性をAUTO設定することにより、状態が管理されます(これは透過的アプリケーション・コンティニュイティの必須設定です)。これらのセッション状態は、フェイルオーバー時に追跡および検証されます。事前設定された状態の外側にはさらに状態を追加できます。

透過的アプリケーション・コンティニュイティ

データベースの計画メンテナンスおよび計画外停止が透過的な場合、アプリケーションでは継続的な可用性を実現します。

- 透過的アプリケーション・コンティニュイティについて
- 様々なアプリケーションの場合の透過的アプリケーション・コンティニュイティ

透過的アプリケーション・コンティニュイティについて

透過的アプリケーション・コンティニュイティは、Oracle Databaseリリース18cのOracle Real Application Clusters (Oracle RAC)に導入されたアプリケーション・コンティニュイティの機能モードであり、セッションおよびトランザクションの状態を透過的に追跡および記録して、リカバリ可能な停止後にデータベース・セッションをリカバリできるようにします。

ユーザー・データベース・セッションのリカバリは安全に実行され、DBAはアプリケーションについて理解したり、アプリケーション・コードを変更したりする必要がありません。アプリケーションがユーザー・コールを発行したときに、セッション状態の使用を分類する状態追跡インフラストラクチャを使用することにより、透過性を実現します。

FAILOVER_TYPE=AUTOの場合、透過的アプリケーション・コンティニュイティは有効になります。

透過的アプリケーション・コンティニュイティを有効にすると、計画メンテナンス時および計画外停止が発生したときにアプリケーションを保護できます。計画メンテナンスの場合、安全な場所(接続テストまたは既知のリカバリ可能ポイントなど)に到達するデータベース・セッションは、データベースで排出されます。排出されないデータベース・セッションの場合、データベースではデータベース・セッションをフェイルオーバーする場所を決定し、アプリケーション・コンティニュイティを起動してこれを実行します。アプリケーション・コンティニュイティは、ユニバーサル接続プールを使用して、Javaベースのアプリケーション、OCIおよびODP.NETアプリケーション(SQL*Plus、すべてのOracle接続プール、Tuxedo、WebLogic Serverおよびサード・パーティのアプリケーション・サーバーを含む)の計画外停止が認識されないようにします。

計画外停止の場合、透過的アプリケーション・コンティニュイティはリカバリ可能なエラー(通常、基盤となるソフトウェア、フォアグラウンド、ハードウェア、通信、ネットワークまたはストレージ・レイヤーに関連)が発生する停止に対して起動され、アプリケーションおよびユーザーにはほとんどの障害が認識されません。

透過的アプリケーション・コンティニュイティを使用すると、DBAはアプリケーションの知識がなくても次のことを実現できます。

- 事前設定された状態のリストアー実行時に、透過的アプリケーション・コンティニュイティは、初期の事前設定されたセッション状態を記録し、さらに状態を監視し、監視対象の状態がフェイルオーバー時にセッション状態を逸脱したことを検出できるセッションの痕跡を記録します。フェイルオーバー時に、透過的アプリケーション・コンティニュイティは事前設定されたセッション状態をリプレイの開始前にリストアし、これらのセッション状態がリプレイの開始前の元の状態と完全に一致することを検証します。これは、アプリケーション・コンティニュイティおよびその他のメカニズム(ログオン・トリガー、ラベル、接続コールバックなど)の両方を使用してリストアされたセッション状態も対象となります。状態が事前設定された状態の外側にある場合は、ログオン・トリガー、コールバックまたはラベルを引き続き追加します。

- セッションのリカバリ時にアプリケーション・レベルの副作用を認識して無効化する—通常の実行時に、透過的アプリケーション・コンティニュイティは副作用を検出します。副作用のタイプは、アプリケーションのロジックに関するものとデータベース・ハウスキーピングに関する内部的なもので区別されています。副作用を含む文を使用するアプリケーションの場合、文を実行しているときの取得は無効になっています。新しいリクエストが開始されると、取得は自動的に再度有効になります。
- 所有関数の可変値を保持する—可変関数は実行のたびに新しい値を返す関数です。可変関数SYSDATE、SYSTIMESTAMP、SYS_GUID、sequence.NEXTVALの元の結果を保持するためのサポートが提供されています。元の値が保持されていない場合、および異なる値がリプレイ時にアプリケーションに返された場合、透過的アプリケーション・コンティニュイティはリプレイを拒否します。権限を使用して順序、日付および時間を保持します。アプリケーションが独自のスキーマを使用している場合、保持するための権限をロールに割り当てると、このロールをユーザーに付与できます。
- リクエスト境界について理解する—リクエスト境界は、アプリケーションとアプリケーション・サーバーが接続プールから接続を流用して返却する場所を決定します。JDBC Thinドライバ(Oracle Database 18c以降)、OCIおよびODP.NET Unmanaged Provider (Oracle Database 19cリリース19.3以降)でアプリケーション・コンティニュイティを使用するアプリケーションの場合、DBAはリクエスト境界について理解している必要はありませんが、リクエスト境界の使用時には透過的アプリケーション・コンティニュイティがリクエスト境界を活用するようになります。リクエスト境界を挿入できるチェックポイントを必ずしも識別できるわけではないため、リクエスト境界の使用をお薦めします。

  Oracle RACリリース18cより前にはリクエスト限界がなく、下位のレイヤー(データベースやドライバなど)でアプリケーションおよびアプリケーション・サーバーがその接続を管理する方法を示す情報がありませんでした。ほぼすべてのアプリケーション・サーバーとエンタープライズ・アプリケーション、および適切なプラクティスを使用するカスタム開発は、最適なパフォーマンスを得るために、そのレイヤーに接続をキャッシュします。下位のレイヤーでは、接続を処理してバランスを取る方法がありません。下位のレイヤーではデータベースへのユーザー・コールしか表示できませんでした。

  透過的アプリケーション・コンティニュイティにより、サーバーおよびドライバはトランザクションとセッションの状態の使用状況を追跡しています。これにより、ドライバは可能なリクエスト境界(暗黙的なリクエスト境界と呼ばれる)を検出して挿入できるようになります。使用可能なリクエスト境界では、開いているオブジェクトはなく、カーソルはドライバ文キャッシュに戻され、開いているトランザクションはありません。セッション状態はリストア可能であると認識されています。無効化イベントが存在していた場合、ドライバは現在の取得を閉じて新しい取得を開始するか、取得を有効にします。サーバーへの次回コール時にサーバーが検証され、必要に応じて、以前に明示的な境界が存在しなかったリクエスト境界が作成されます。

Java (Oracle Database 18c以降)、OCIおよびODP.NET Unmanaged Provider (Oracle Database 19cリリース19.3以降)で透過的アプリケーション・コンティニュイティを使用すると、アプリケーションのリソース使用率が少なくなりリカバリ時間が短縮されます。これは、状態に影響しない文が記録されず、それらが不要になったときにパージされ、リクエスト境界が自動的に拡張されるためです。

様々なアプリケーションの場合の透過的アプリケーション・コンティニュイティ

透過的アプリケーション・コンティニュイティは、自動的に状態追跡システムによって追跡される3つの異なるグループに属するアプリケーションに対応しています。

次のタイプの様々なアプリケーションがあります。

- リクエスト境界: リクエスト境界とともにコンテナを使用するアプリケーションでは、アプリケーション・コンティニュイティで明示的な境界間のリプレイを管理できます。
- データベースに依存しない: アプリケーションでは接続の確立時に状態を設定します。非トランザクション・セッション状

態を再度変更しません。変更することはごくまれです。これらのアプリケーションの場合、アプリケーション・コンティニュイティは暗黙的な境界を指定します。

- ブラック・ボックス: 実行時にOracle専用の状態を使用しているか、状態を変更している(あるいはその両方の)アプリケーション。このカテゴリはさらに次のように分かれます。
  - 表示可能な境界のないOLTPなどの短いユーザー・コールが含まれたアプリケーション
  - DSS、レポートやウェアハウスなどの長いユーザー・コールが含まれたアプリケーション

リクエスト境界

リクエスト境界はデータベース・リクエストの開始と終了をマークするタグです。Oracle Database 12cリリース2 (12.2.0.1)以降、リクエスト境界を埋め込んだ接続プールには、Oracle Universal Connection Pool、すべてのWebLogic Serverデータ・ソース、Tuxedo、Oracle Call Interface、ODP.NET Unmanaged Providerおよび標準のサード・パーティのアプリケーション・サーバーとスタンドアロンのJavaプールがあります。これらはOracle Database 12c JDBCドライバの`PooledConnection`インタフェースおよびSQL*PLUSを使用します。

Oracle Databaseがリクエスト境界を認識した場合、次のようになります。

- データベースは、接続をアタッチおよび解放する際に発生するパフォーマンス・オーバーヘッドがなく、効率的にWebリクエストを処理できるため、リクエスト内の複雑な状態を多重化、排出、削除および許可したり、再度分散させることができます。リクエスト境界を使用しない場合、データベースの下位のレイヤーは、Webリクエストを認識しません。その結果、データベースはOracleクライアント・アクション、高速接続フェイルオーバーなどのアドバイザ・メソッドとヒューリスティック、接続検証および状態のアドバイスに依存します。
- リプレイの長さは、アプリケーション・コンティニュイティによってパージされるものより小さいリクエスト内にあるユーザー・コール後の初期状態に制限されます。リクエスト境界は、リプレイの長さ、および計画メンテナンスでの排出場所(リクエストの終了時)および計画メンテナンスでのフェイルオーバーの場所(リクエストの開始時)の制御の重要なヒントになります。
- Java用の透過的アプリケーション・コンティニュイティを使用する場合、最初のリクエスト境界のみが必要になります(Oracle Database 18cの場合のみ)。
- Java用のアプリケーション・コンティニュイティを使用する場合、リプレイ・ドライバが安全な場所を検出して自動的にリクエスト境界を移動します。この機能はAUTOでのみ使用できます。
- リクエスト境界を設定する中間層コンテナを使用してデプロイされたアプリケーションは、データベース・サーバーが提供する透過性機能の完全なセットにアクセスできます。データベースはクライアントがリクエスト境界を設定するタイミングを検出し、境界を使用して排出、フェイルオーバー、集中およびスループットの測定のための安全な場所をマークします。

リクエスト境界により、アプリケーションはすべての複雑な非トランザクション・セッション状態をリクエスト内で使用できます。リクエスト境界の仕様では、これらの状態が境界を超えて依存しないことが必要です。

データベースに依存しないアプリケーション

データベースに依存しないアプリケーション(リクエスト境界のないアプリケーション)は単純な非トランザクション状態を設定し、Oracle固有の機能または順序のいずれも使用しません。これらのアプリケーションは、通常、接続が作成されたときに一度状態を設定します。その後、状態を再度変更することはありません。変更することはごくまれです。このカテゴリのプリケーションには、サーバー側のセッション状態を作成しない匿名のPL/SQLを使用するアプリケーションが含まれています。

JDBCアプリケーションに透過的アプリケーション・コンティニュイティを使用する場合、状態の分類を使用して、認証後にアプリ

ケーション・コンティニュイティの記録を有効化および開始するポイント、および取得が無効化イベントによって無効化された後に記録を再度有効化するポイントを検出します。最初のリクエスト境界のみが必要ですが、存在するリクエスト境界が使用されます。リクエスト境界は、SQL*Plusの場合は必須ではありません。これらはODP.NET、OCIセッション・プール、Tuxedo、Oracle Universal Connection Pool用に埋め込まれています。

アプリケーション・コンティニュイティ保護チェック

アプリケーション・コンティニュイティ保護チェック(ACCHK)機能は、アプリケーション・コンティニュイティによるアプリケーションの保護を説明するアプリケーション・コンティニュイティのカバレッジ・レポートおよびビューを生成します。

- アプリケーション・コンティニュイティ保護チェックについて
- Oracle Database 19c用のACCHKビューおよびロールの作成
- アプリケーション・コンティニュイティ保護チェックの有効化および無効化
- アプリケーション・コンティニュイティ保護チェックの実行

アプリケーション・コンティニュイティ保護チェックについて

アプリケーション・コンティニュイティ保護チェック(ACCHK)ユーティリティは、アプリケーション・コンティニュイティを使用するアプリケーションの保護ガイダンスを提供します。

ACCHKは、アプリケーション・コンティニュイティを使用する各アプリケーションの保護レベルに関するガイダンスを提供し、必要に応じて保護を向上させるために役立ちます。ACCHKでは、アプリケーション・コンティニュイティのトレースを使用してワークロードのカバレッジを収集し、リクエストに従って詳細情報を提供します。データベース・ワークロードを実行する前に、アプリケーション・コンティニュイティのトレースを有効にしてカバレッジを収集する必要があります。ACCHKは、失敗したフェイルオーバーの診断も提供します。

データベース・ビューおよびPL/SQLベースのレポートには、フェイルオーバーに対するアプリケーションの保護レベルが表示されます。アプリケーションが完全には保護されていない場合、ACCHKはそのアプリケーションを識別し、アプリケーションが完全に保護されていない理由を検出して、保護を強化する方法を示します。

保護されたアプリケーションの場合、ACCHKは、アプリケーションのどの操作が保護されていて、アプリケーションのどの操作が保護されていないかも報告します。アプリケーション・コンティニュイティによって保護されていないアプリケーションの操作または構成がある場合は、構成を変更して保護のカバレッジを増やすことができます。ACCHKは、ワークロードのカバレッジ文およびパーセンテージ値を含むレポートを生成します。ACCHKレポートには、実行された操作の数、完全に保護された操作の数および完全には保護されなかった操作の数も表示されます。

関連項目

- アプリケーション・コンティニュイティの理解
- 透過的アプリケーション・コンティニュイティ

Oracle Database 19c用のACCHKビューおよびロールの作成

Oracle Database 19cでアプリケーション・コンティニュイティ保護チェック(ACCHK)を初めて使用する前に、PDBでACCHKビューおよびロールを手動で作成する必要があります。

1. SQL*Plusを使用して、Oracleプラガブル・データベース(PDB)に接続します。
2. dbms_app_cont_admin.acchk_viewsプロシージャを使用して、PDB用のアプリケーション・コンティニュイティ保護チェック・ビューおよびロールを作成します。

```
SQL> execute dbms_app_cont_admin.acchk_views;
```

前述のプロシージャにより、ACCHKで使用されるビューおよびロールが作成されます。ビューおよびロールがすでに存在する場合でも、このプロシージャを安全に繰り返すことができます。

ノート:

COMPATIBLE パラメータを 12.2.0 以上に設定します。以前に COMPATIBLE パラメータが低い値に設定されていた場合、COMPATIBLE パラメータの更新後に初めてプロシージャを実行すると、acchk_views プロシージャによって ACCHK ビューおよびロールが作成されます。

関連項目

- アプリケーション・コンティニュイティ保護チェックの実行

アプリケーション・コンティニュイティ保護チェックの有効化および無効化

アプリケーション・コンティニュイティを使用するアプリケーションのアプリケーション・コンティニュイティ保護チェック(ACCHK)機能を手動で有効または無効にできます。

アプリケーション・コンティニュイティ保護チェックはデフォルトでは有効化されていません。ACCHKを有効または無効にし、アプリケーションの保護レベルを確認するレポートを生成するには、次の手順に従います。

1. ACCHK_READロールを使用して、アプリケーション・コンティニュイティ保護チェック・レポートおよびビューを実行するユーザーに読取りアクセス権を付与します。

```
GRANT ACCHK_READ TO USER;
```

2. dbms_app_cont_admin.acchk_set(true)プロシージャを使用して、アプリケーションのアプリケーション・コンティニュイティのトレースを有効にします。

```
SQL> execute dbms_app_cont_admin.acchk_set(true);
```

デフォルトでは、ACCHKは600秒後に自動的に無効になります。より小さい数値を指定すると、自動無効化時間を短縮できます。たとえば、300秒後にACCHKを無効にするには、dbms_app_cont_admin.acchk_set(true, 300)プロシージャを使用します。

dbms_app_cont_admin.acchk_set(true)プロシージャは、接続しているデータベース・レベルでアプリケーション・コンティニュイティのトレースを有効にします。CDBレベルで接続している場合、CDBに対してトレースが有効になり、PDBレベルで接続している場合、PDBに対してトレースが有効になります。

ノート:

COMPATIBLE パラメータを 12.2.0 以上に設定します。以前に COMPATIBLE パラメータが低い値に設定されていた場合、COMPATIBLE パラメータの更新後に初めてプロシージャを実行したときに、acchk_set プロシージャによって ACCHK ビューおよびロールが作成されます。

3. dbms_app_cont_admin.acchk_set(false)プロシージャを使用して、アプリケーションの新しいセッションのアプリケーション・コンティニュイティのトレースを無効にします。

```
SQL> execute dbms_app_cont_admin.acchk_set(false);
```

ノート:

- 時間が経過すると、現在のセッションのトレースが無効になります。
- トレースは、Oracle Real Application (Oracle RAC)クラスタ全体に対してデフォルトで有効になっています。

関連項目

- ACCHK_SETプロシージャ
- アプリケーション・コンティニュイティ保護チェックの実行

アプリケーション・コンティニュイティ保護チェックの実行

アプリケーション・コンティニュイティ保護チェック(ACCHK)レポートを生成して、保護レベルのガイダンス、不完全な保護の理由、および保護レベルを上げる方法を取得します。

ACCHKユーティリティは、事前に生成されたデータベース・トレースを使用してアプリケーション・コンティニュイティ・カバレッジをレポートする後処理ツールです。ワークロードを実行してレポートを生成する前に、アプリケーション・コンティニュイティのトレースおよびアプリケーション・コンティニュイティ保護チェックを有効にします。

1. アプリケーションのACCHKおよびトレースを有効にした後、一連のデータベース・オプションを実行します。
   ACCHKは、アプリケーション・コンティニュイティ・セッションのレポートのみを生成します。
2. dbms_app_cont_report.acchk_reportプロシージャを使用して、アプリケーション・コンティニュイティ保護チェック・レポートを生成します。

   ```
   SQL> SET SERVEROUTPUT ON FORMAT WRAPPED;
   SQL> execute dbms_app_cont_report.acchk_report;
   ```

   レポートのタイプは、FULL、WARNINGまたはSUMMARYから指定できます。たとえば:

   ```
   SQL> SET SERVEROUTPUT ON FORMAT WRAPPED;
   SQL> execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.FULL);
   SQL> execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.WARNING);
   SQL> execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.SUMMARY);
   ```

   デフォルトのレポート・タイプはSUMMARYです。

3. レポートを分析し、完全には保護されていないアプリケーションの保護レベルを上げます。たとえば、サマリー・レポートは次のようになります。

   ```
   ------------------------------------
   ------------ ACCHK Report ------------
   ------------------------------------
   CON_ID SERVICE       FAILOVER PROTECTED_ PROTECTED_ REQUESTS AVG_CALLS/ PROTECTED_
   AVG_TIME/  PROTECTED_TIME/ EVENT_  ERROR_ PROGRAM   MODULE           ACTION    SQL_ CALL
   TOTAL
   ```

```
                                CALLS %    TIME %              REQUEST    CALLS/REQUEST REQUEST
MS REQUEST MS      TYPE    CODE                                           ID
------ ------------ -------- ---------- ---------- -------- ---------- ------------ --------
-- --------------- ------- ------ --------- ----------------- --------- ---- --------- ------
3      srv_tacr_pdb1 AUTO     98.734     98.432     117      9.453      9.333         2279.751
2244.014        DISABLE 41409  JDBC Thin AddCustNewOrder   Action-20      COMMIT    1

Client
3      srv_tacr_pdb1 AUTO     98.734     98.432     117      9.453      9.333         2279.751
2244.014        REPLAY_ 41412  JDBC Thin InsertNewChecksum Action-1       SQL/PLSQL 1

FAILED         Client                                    Execu
End of report.
```

次の例は、ACCHKビューを使用してACCHKレポートから詳細情報を問い合せる方法を示します。

例6-2 DBA_ACCHK_EVENTSビューの使用

この例の最後の行は、srv_tacr_pdb1サービスを使用しているアプリケーションにアプリケーション・コンティニュイティの失敗の原因となったイベントがあることを示しています。

```
SQL> SELECT * FROM DBA_ACCHK_EVENTS ORDER BY TIMESTAMP;
INST_ID CON_ID TIMESTAMP        SESSION_ID SERIAL# SERVICE_NAME  PROGRAM  MODULE            ACTION
SQL_ID CALL_NAME EVENT_TYPE ERROR_CODE
------- ------ ---------- ----------- ---------- ------- -------------- -------  ----------------- ---------
------ --------- ---------- ----------
2       3    21-SEP-20          9598       1644    srv_tacr_pdb1 JDBC     AddCustNewOrder   Action-36
COMMIT    DISABLE    41409
             06.54.18.191 PM                                    Thin
             -07:00                                             Client
2       3    21-SEP-20          1703       61265   srv_tacr_pdb1 JDBC     InsertNewChecksum  Action-1
SQL/PLSQL REPLAY_    41412
             06.51.07.624 PM                                    Thin
Execution FAILED
             -07:00                                              Client
```

例6-3 DBA_ACCHK_EVENTS_SUMMARYビューの使用

この例の最後の行は、srv_tacr_pdb1サービスを使用しているアプリケーションにアプリケーション・コンティニュイティの失敗の原因となったイベントがあることを示しています。

```
SQL> SELECT * FROM DBA_ACCHK_EVENTS_SUMMARY ORDER BY SERVICE_NAME;
INST_ID CON_ID SERVICE_NAME  FAILOVER_TYPE FAILOVER_RESTORE RESET_STATE PROGRAM MODULE
ACTION    SQL_ID CALL_NAME EVENT_TYPE ERROR_CODE FREQUENCY
------- ------ ------------ ------------- ---------------- ----------- ------- ----------------- ----
----- ------ --------- ---------- ---------- ----------
2       3      srv_tacr_pdb1 AUTO          AUTO             LEVEL1      JDBC    AddCustNewOrder
Action-20        COMMIT    DISABLE    41409      1
                                                                        Thin
Execution
                                                                        Client
2       3      srv_tacr_pdb1 AUTO          AUTO             LEVEL1      JDBC    InsertNewChecksum
Action-1         SQL/PLSQL REPLAY_    41412      1
                                                                        Thin
Execution FAILED
                                                                        Client
```

例6-4 DBA_ACCHK_STATISTICSビューの使用

この例では、最初の行は、srv_tacr_pdb1サービスを使用しているアプリケーションに、JDBCからの11個の暗黙的なリクエストとアプリケーション内の31個のコールがあることを示しています。これらのリクエストの30個のコールは保護されています。

```
SQL> SELECT * FROM DBA_ACCHK_STATISTICS ORDER BY TIMESTAMP;
INST_ID CON_ID TIMESTAMP        SESSION_ID SERIAL# STAT_TYPE  SERVICE_NAME  FAILOVER_ FAILOVER_ RESET_
PROGRAM  BEGIN_   END_     USER_CALLS_ PROTECTED_CALLS_ TIME_IN_ TIME_PROTECTED_
                                                                        TYPE      RESTORE   STATE
REQUESTS REQUESTS IN_REQUESTS IN_REQUESTS      REQUESTS IN_REQUEST
------- ------ ---------------- ---------- ------- ---------- ------------- --------- --------- ------
-------  -------- -------- ------------ ----------------- -------- ----------------
2       3      21-SEP-20        5653       54237   SESSION_   srv_tacr_pdb1 AUTO      AUTO      LEVEL1
JDBC     11       11       31          30              13316750 12415247
               06.54.25.321 PM                     STATISTICS
Thin
               -07:00
Client
2       3      21-SEP-20        11291      26560   SESSION_   srv_tacr_pdb1 AUTO      AUTO      LEVEL1
JDBC     3        3        50          49              13094072 13068259
               06.54.24.915 PM                     STATISTICS
Thin
               -07:00
Client
```

例6-5 DBA_ACCHK_STATISTICS_SUMMARYビューの使用

この例では、srv_tacr_pdb1サービスを使用しているアプリケーションに144個の暗黙的なリクエストがあり、これらのリクエストで99.5688328パーセントのコールがアプリケーション・コンティニュイティまたは透過的アプリケーション・コンティニュイティによって保護されています。

```
SQL> SELECT * FROM DBA_ACCHK_STATISTICS_SUMMARY ORDER BY SERVICE_NAME;
INST_ID CON_ID SERVICE_NAME  FAILOVER_ FAILOVER_ RESET_ TOTAL_   PROTECTED_CALLS_ PROTECTED_TIME_
AVG_USER_CALLS_ AVG_PROTECTED_    AVG_TIME_   AVG_TIME_
                              TYPE      RESTORE   STATE  REQUESTS PERCENT          PERCENT
IN_REQUESTS     CALLS_IN_REQUESTS IN_REQUESTS PROTECTED_IN_REQUESTS
------- ------ ------------- --------- --------- ------ -------- ---------------- --------------- ----
----------- ----------------- ----------- ---------------------
2       3      srv_tacr_pdb1 AUTO      AUTO      LEVEL1 144       99.5688328       99.0130288
22.5486111      22.4513889         3078654.35  3048268.92
```

次の統計を使用して、アプリケーションの保護を監視することもできます。

- 累積開始リクエスト
- 累積終了リクエスト
- リクエストの累積時間
- リクエスト内の累積ユーザー・コール
- アプリケーション・コンティニュイティで保護される累積ユーザー・コール
- リクエストの累積DB時間
- リクエスト内で保護される累積DB時間

関連項目

- ACCHK_REPORTプロシージャ
- アプリケーション・コンティニュイティ保護チェックの有効化

アプリケーション・コンティニュイティの操作および使用

この項では、アプリケーション・コンティニュイティの動作、およびアプリケーション・コンティニュイティをアプリケーションで使用する方法について説明します。

この項には次のトピックが含まれます:

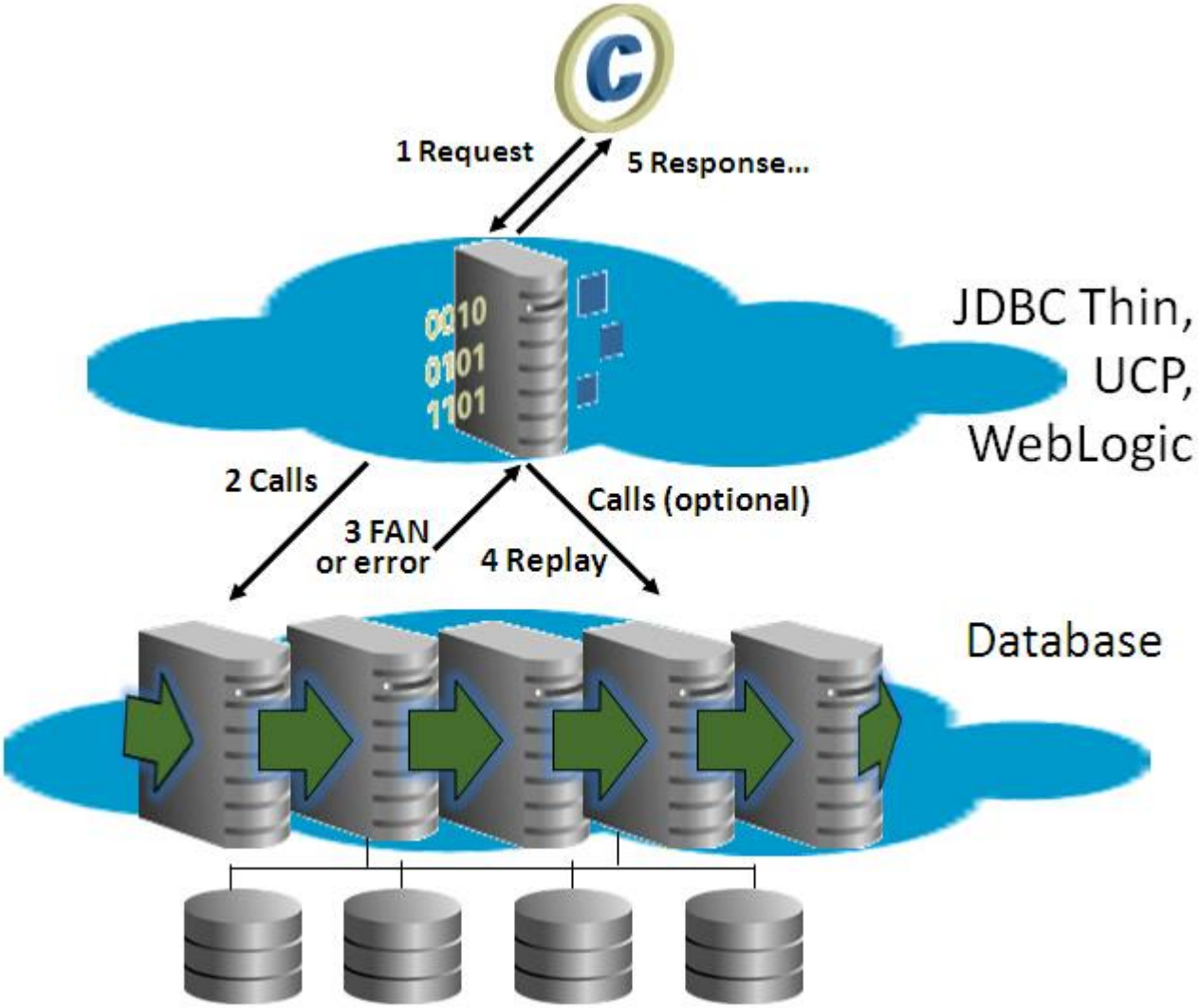
- アプリケーション・コンティニュイティがアプリケーションで機能する仕組み
- アプリケーション・コンティニュイティの使用のアクション
- 可変関数とアプリケーション・コンティニュイティ
- 可変値の管理
- 保護レベルの統計
- セッション状態一貫性

アプリケーション・コンティニュイティがアプリケーションで機能する仕組み

リカバリ可能なエラーが発生したときに、リプレイが有効化されていると、アプリケーション・コンティニュイティによってデータベース・セッションのリカバリが試行されます。

次に、アプリケーション・コンティニュイティの動作のしくみを図で示します。

図6-1 アプリケーション・コンティニュイティ



回復可能なエラーの後にデータベース・セッションのリカバリを試行するために、アプリケーション・コンティニュイティは次のステップを実行します。

ノート:

データベース・セッションのリカバリ・ステップは、計画外の停止と計画的な停止の両方に適用されますが、特定のステップは停止のタイプに応じて変化します。

1. クライアント・アプリケーションが要求を発行します。この要求は、中間層(ユニバーサル接続プール(UCP)、ODP.NET、WebLogic Server、OCIセッション・プール、Tuxedo、またはUCPを使用するサード・パーティ・プールなど)に渡されてデータベースに転送されます。アプリケーションは、JDBCリプレイ・ドライバまたはOCIドライバを使用して直接データベースに要求を発行することもあります。
2. 中間層またはJDBCリプレイ・ドライバやOCIドライバが要求内の各コールを発行します。
3. 計画済または計画外のDOWN高速アプリケーション通知(FAN)イベントまたはリカバリ可能なエラーが発生します。FANまたは高速接続フェイルオーバー(FCF)により、デッド状態の物理セッションが中断されます。
4. アプリケーション・コンティニュイティがリプレイを開始し、次を実行します。
    a. デッド状態の物理セッションを新しいクリーンなセッションに置き換えます。
    b. 進行中のトランザクションがオープンであった場合、その結果を確認するためにトランザクション・ガードを使用してリプレイを準備します。
    c. FAILOVER_RESTORE=LEVEL1またはFAILOVER_TYPE=AUTOの場合、アプリケーション・コンティニュイティでは共

通の初期セッション状態をリストアします。アプリケーション・コンティニュイティは、アプリケーションがコールバック時にFAILOVER_RESTOREで指定されていない初期セッション状態も設定している場合、ラベル・コールバックまたは初期コールバックを使用します

d. トランザクション状態および非トランザクション状態をリカバリし、クライアント・ドライバによって確認されたデータおよびメッセージが、クライアントが確認して決定した可能性があったものと同じであることをステップごとに検証し、データベース・セッションを再構築します。

e. リプレイが終了し、ランタイム・モードに戻ります。

f. 最後にキューに入れられたコールを発行します。

これは、停止が検出されたときに実行された最後のコールです。リプレイ時には、このコールのみがCOMMITを実行できます。セッションの再構築の途中でCOMMITが行われると、リプレイは中断されます(自律型トランザクションは除く)。

5. レスポンスがアプリケーションに戻されます。

リプレイが成功した場合、アプリケーションは問題をマスクした状態で続行できます。失敗した場合、アプリケーションは元のエラーを処理する必要があります。

通信障害後のアプリケーション・コンティニュイティの動作は、関連するOracle製品およびテクノロジによって異なります。たとえば:

- Oracle RACまたはOracle Active Data Guardファームを使用している場合、実行中の別のインスタンスで接続が再確立された後、アプリケーション・コンティニュイティはセッションを再構築し、最後のトランザクション(進行中のものがある場合)をリプレイしようとします。
- Oracle Active Data Guardを使用し、スタンバイ・サイトにフェイルオーバーする場合、アプリケーション・コンティニュイティはフェイルオーバー・インスタンスに接続し、セッションを再構築し、最後のトランザクション(トランザクションが進行中であった場合)をリプレイしようとします。(Oracle Active Data Guardスイッチオーバーおよびフェイルオーバーによってデータが失われ、これがラグが承認されたOracle Active Data Guardリーダー・ファームでない場合、アプリケーション・コンティニュイティはリプレイしません)。
- Oracle RACまたはOracle RAC One Nodeを使用していて、Oracle Active Data Guardは使用していない場合、停止が原因ですべてのパブリック・ネットワークが中断されるか、データベースまたはデータベース・セッションがしばらくの間停止すると、アプリケーション・コンティニュイティは、セッションを再構築し、接続がリストアされた後にデータベースに対して最後のトランザクション(トランザクションが進行中であった場合)をリプレイしようとします。
- 個別のサービスごとにSRVCTLコマンドを実行する必要はなくなり、影響を受けるすべてのサービスのノード名、データベース名、プラガブル・データベース名、またはインスタンス名を指定することのみが必要になります。

関連項目

- 計画メンテナンスに対するアプリケーション・コンティニュイティの使用

アプリケーション・コンティニュイティの使用のアクション

アプリケーション・コンティニュイティは、Oracle統合スタックを使用しているときに、アプリケーションの変更なしに(またはわずかな変更で)停止をマスクします。

Oracle Application Continuityおよび透過的アプリケーション・コンティニュイティのサポート

アプリケーション・コンティニュイティのサポートは、多くのOracleアプリケーションに統合されています。

アプリケーション・コンティニュイティは、次のOracleテクノロジを使用した一般的な用途で使用できます。

- ODP.NET、管理対象外ドライバ12.2以降
- OCIセッション・プール12.2以降
- ユニバーサル接続プール12.1以降
- Oracle WebLogic Server 12c
- JDBC Thin Oracleリプレイ・ドライバ12.1以降
- Java接続プールまたはスタンドアロンJavaアプリケーション(Oracle JDBC - Replay Driver 12c以降をリクエスト境界で使用)
- SQL*Plus 19.3以降
- ユニバーサル接続プールを使用したサード・パーティのJDBCアプリケーション・サーバー

透過的アプリケーション・コンティニュイティは、次のOracleテクノロジとともに一般的な用途で使用できます。

- Oracle Call Interface (OCI)およびOracle C++ Call Interface (OCCI)
- ODP.NET、管理対象外ドライバ12.2以降
- Oracle Tuxedo 19.3以降
- OCIセッション・プール12.2以降
- SQL*Plus 19.3以降
- Oracle JDBC OCIドライバ(Thickドライバは一般に非推奨)

Javaのアプリケーション・コンティニュイティは、ユニバーサル接続プール、WebLogicデータ・ソース(非XAおよびXAデータ・ソースを含む)に埋め込まれています。また、JDBC Thinリプレイ・ドライバ単独(Apache TomcatやカスタムのJava接続プールなど、Oracle接続プールなしのJDBCリプレイ・ドライバ)で使用できます。OCIのアプリケーション・コンティニュイティは、SQL*Plus、OCIセッション・プール12.2以降、およびODP.NET Unmanaged Providerに埋め込まれています。透過的アプリケーション・コンティニュイティを使用すると、Oracle Database 18c以降ではJDBCアプリケーション、およびOracle Database 19c (19.3)以降ではOCIアプリケーションが自動的に有効になります。

接続プールまたはコンテナがOracle接続プールを使用しない場合、多くのサード・パーティのJavaアプリケーションは、ユニバーサル接続プールによる接続プールの置換を完全にサポートしています。これには、IBM WebSphereとApache Tomcatが含まれます。また、アプリケーションは独自のリクエスト境界を追加できます(Javaの場合のみ)。

リクエスト境界

Oracle Databaseリリース12.1.以降、リクエスト境界はOracle接続プールに埋め込まれています。リクエスト境界は、JDK9以降の標準であるサード・パーティ製Javaアプリケーション・サーバーにも埋め込まれています。Oracle接続プールを使用すると、各リプレイのサイズを定めるリクエスト境界がチェックアウト時およびチェックイン時に暗黙的にマークされます。サード・パーティの接続プールを使用するときは、UCPを使用するか(Javaの場合)、透過的アプリケーション・コンティニュイティを使用するか、リクエスト境界を追加するか、JDK9以降の標準であるサード・パーティ製Javaアプリケーション・サーバーを使用します。リクエスト境界は、透過的アプリケーション・コンティニュイティの使用時に状態追跡を使用して検出されます。この機能は、Oracle Database 18c Javaリプレイ・ドライバおよびOracle Database 19c OCIドライバ(オープン・ソースとODBCを含む)以降で使用できます。

ノート:

Oracle Database 18c のみ: Java には初期 beginRequest が必要になります。これは、最新バージョンの Java リプレイ・ドライバを使用しているときには不要です。

関連項目

- Oracle Data Provider for .NETの概要
- JDBCの概要

アプリケーション・コンティニュイティ構成タスクの概要

各種Oracleアプリケーションのアプリケーション・コンティニュイティ機能が自動的に使用されます(必要なサービス属性を設定した場合)。

アプリケーション・コンティニュイティのサポートは多くのOracleアプリケーションに統合されているため、アプリケーション・コンティニュイティに関連するサービス属性を設定する場合、それらのアプリケーションの機能は自動的に使用されます。

アプリケーションの透過的リプレイを確実にするための主なアクションは次のとおりです。

1. Javaを使用している場合のみ、アプリケーションがOracle JDBCの具象クラスを使用するかどうか判別します。アプリケーション・コンティニュイティを使用するには、非推奨の具象クラスを置換する必要があります。

   ORAchkユーティリティに-acchkパラメータを使用して、アプリケーションに具象クラスがあるかどうかを確認します。リプレイされてはならないものがある場合は、アプリケーション・コンティニュイティのない接続を使用します。(ほとんどのアプリケーションがリプレイ可能です)。

   関連項目:

   ORAchkの詳細は、*Oracle Autonomous Health Framework*ユーザーズ・ガイドを参照してください
2. 必要なCPUおよびメモリー・リソースがあることを確認します。
   - CPU: アプリケーション・コンティニュイティは、クライアント側とサーバー側で管理され、動作のための最小限のCPUオーバーヘッドが必要になります。

     クライアントでは、プロキシ・オブジェクトの構築とガベージ・コレクション(GC)のためにCPUが使用されます。

     サーバーでは、CPUは検証のために使用されます。CPUオーバーヘッドは、検証がハードウェアによって補佐されている、現在のIntelおよびSPARCチップを使用したプラットフォームでは減少します。
   - メモリー: アプリケーション・コンティニュイティを使用している場合、呼出しがリクエストの最後まで保持されるため、リプレイ・ドライバにはベース・ドライバよりも多くのメモリーが必要です。リクエストの最後で、呼出しはガベージ・コレクタに解放されます。この処理は、クローズされた呼出しを解放するベース・ドライバによって異なります。

     リプレイ・ドライバのメモリー消費量は、リクエストごとのコール数によって異なります。この数が小さい場合、リプレイ・ドライバのメモリー消費量は少なくなり、ベース・ドライバと同程度になります。

     最良のパフォーマンスを得るには、クライアント側でパラメータ-Xmxと-Xmsの両方に同じ値を設定する必要があります。たとえば、十分なメモリーがある場合、仮想マシン(VM)に4から8GB (以上)を割り当てます。たとえば、4GBであれば-Xms4gと設定します。-Xmsパラメータがこれより低い値に設定されている場合、VMもオペレー

ティング・システムから低い値を使用するため、パフォーマンスが悪化する可能性があり、ガベージ・コレクション操作が増加します。

3. 各リクエストに対してアプリケーションが接続プール(たとえば、WebLogic Server Pool、ユニバーサル接続プール、OCIセッション・プール、Oracle Tuxedoリクエストまたは各リクエストのODP.NET接続プール)から接続を流用して返却しているかどうか、またはリクエスト境界を識別するためにbeginRequestおよびendRequest APIをアプリケーション固有の接続プールに追加するかどうかを判別します(Javaの場合のみ)。

   重要:

   リクエスト境界以外の場所では、Java API コールの beginRequest および endRequest は使用しないでください(接続プールの接続の流用と返却)。endRequest は、リクエストが完了していることと、現在はステートレスであることを示します。次の beginRequest からリプレイが開始されます。前の状態がある場合は、FAILOVER_RESTORE またはコールバックを使用して、その状態を再確立する必要があります。

4. アプリケーション・コンティニュイティは、リクエスト内のすべての状態をリプレイします。アプリケーションが接続を公表する前に状態を設定する場合は、FAILOVER_RESTOREまたはコールバックが必要です。Oracle WebLogic Serverまたはユニバーサル接続プールの使用時には、FAILOVER_RESTORE、接続ラベリングまたはトリガーを使用します。OCIセッション・プール、Oracle TuxedoまたはODP.NETをOracle Database 18c以降のクライアントともに使用する場合、FAILOVER_RESTOREを使用して、必要な場合はTAFコールバックのみを追加します。ラベリングはランタイムとリプレイの両方で使用されます。

5. アプリケーションがフェイルオーバー時にSYSDATE、SYSTIMESTAMPおよびSYS_GUIDとその順序を必要とするか、さらにこれらの元の値の維持構成を行う必要があるかどうかを判別します。

6. session_state_consistency値のアプリケーション・スタイルを評価し、サービスに適した値を設定します。

   - session_state_consistencyがAUTOに設定されている場合、透過的アプリケーション・コンティニュイティはセッション状態を監視し、処理を決定します。状態の使用状況が不明か、状態が将来変更されることがわかっている場合は、透過的アプリケーション・コンティニュイティを使用します。追加の事前設定された状態をリストアする必要がある場合があるため、事前設定されたセッション状態のリストを参照してください。
   - session_state_consistencyがDYNAMICに設定されている場合、アプリケーションはリクエスト時に環境または設定を変更します。リプレイは、最初のCOMMITの後、次のリクエストの開始まで無効化されます。デフォルトのモードはDYNAMICであり、ほとんどのアプリケーションに適しています。
   - session_state_consistencyがSTATICに設定されている場合、アプリケーションは、初期設定後にセッション状態を変更しません。このモードは、PL/SQL状態を使用せずにトランザクションの途中でALTERを使用しない、データベースに依存しないアプリケーションの基本的なモードです。透過的アプリケーション・コンティニュイティは、session_state_consistencyをSTATICではなくAUTOに設定して使用します。AUTO設定により、セッション・ステートが静的であることが検証されます。

7. アプリケーションにリプレイが不要なリクエストがあるかどうかを調べます。

   たとえば、外部PL/SQLアクションを使用するリクエストに対して、リプレイを無効化する必要がある場合があります。

8. 次の構成のガイドラインに従ってください。

   - Javaの場合は、Oracle Database 12c リリース1 (12.1.0.1)以降を使用します。OCIベースのアプリケーションの場合は、Oracle Database 12c リリース2 (12.2)以降を使用します。
   - .NETアプリケーションの場合は、Oracle Database 12c リリース2 (12.2)以降に接続しているODP.NET管理対象外ドライバ12.2以降を使用します。デフォルトでは、この構成でODP.NETアプリケー

ションのアプリケーション・コンティニュイティが有効化されます。OCIセッション・プールを使用しないOCIベース・アプリケーション(SQL*Plusを含む)を使用する場合は、自動で境界が追加される透過的アプリケーション・コンティニュイティを使用します。

- Javaベースのアプリケーションでは、JDBC Replayデータソースに対して構成されたユニバーサル接続プール12.1 (以上)またはWebLogic Server 12.1.2 (以上)を使用します。または、サード・パーティ・アプリケーション(サード・パーティJDBCプールなど)の場合はJDBCリプレイ・ドライバを使用します。IBM WebSphere、Apache TomcatおよびRedHat Springの場合は、プールされたデータ・ソースとしてUCPを使用することが最も効果的なソリューションになります。

  カスタムJavaプールおよびスタンドアロンJavaアプリケーションは、JDBC Replayデータソースを直接使用することもできます。カスタムJavaプールとスタンドアロン・アプリケーションを使用する場合は、自動で境界が追加される透過的アプリケーション・コンティニュイティの使用をお薦めします。アプリケーションに、Java APIのbeginRequestとendRequestを追加することもできます。

- アプリケーションがOracle接続プールからの接続の流用および戻しを行わない場合は、明示的にリクエスト境界をマークしてください。たとえば、カスタムJDBCプールなどのプールを使用する場合は、自動で境界が追加される透過的アプリケーション・コンティニュイティの使用をお薦めします。アプリケーションに、Java APIのbeginRequestとendRequestを追加することもできます。これらのAPIは、接続プールのないスタンドアロンJDBCアプリケーションに対しても使用できます。

- エラーに対する高速な中断としてFANを有効化します。これは、フェイルオーバーが開始する前にTCPハングが発生することを回避するにはを不可欠です。12.2 FANはJDBCドライバおよびOCIドライバに組み込まれ、Javaではデフォルトでオンになっています。

- 接続にはデータベース・サービスを使用します。SIDやインスタンス名、または管理サービス(DB_NAMEまたはDB_UNIQUE_NAME)は使用しないでください。

- 新規着信接続に対する再試行およびこれらの再試行間の遅延を設定する接続文字列を使用します。

- サービスに対して、アプリケーション・コンティニュイティの手動モードの場合はFAILOVER_TYPEをTRANSACTIONに設定するか、透過的アプリケーション・コンティニュイティの場合はFAILOVER_TYPEをAUTOに設定します。COMMIT_OUTCOMEをTRUEに設定し、OCI FANに対してNOTIFICATIONをTRUE-に設定します。必要に応じて、使用に適した接続を探す場合は、GOALをSERVICE_TIMEに、CLB_GOALをLONGに設定します。

- リクエスト境界および保護レベルの統計を使用してカバレッジのレベルを監視します。さらに詳細情報が必要な場合、アプリケーション・コンティニュイティ・チェック・カバレッジ(ORAchkユーティリティに付属)を使用すると、アプリケーション・コンティニュイティによって完全に保護されているリクエストの割合と、完全には保護されていないリクエストの場所が報告されます。このカバレッジ・チェックは、デプロイメントの前と、アプリケーションの変更後に使用します。開発者と管理者は、アプリケーション・リリースが基盤のインフラストラクチャの障害から、どの程度適切に保護されているかを認識できます。問題がある場合、アプリケーションのリリース前に、その問題を修正できます。または、カバレッジのレベルを考えて放棄します。

高可用性およびアプリケーション・コンティニュイティに対応する接続の構成

高可用性のアプリケーションに使用する接続を構成する際の一般的な推奨事項を示します。

Javaを使用している場合、oracle.jdbc.replay.OracleDataSourceImpl、
oracle.jdbc.replay.OracleConnectionPoolDataSourceImplまたは
oracle.jdbc.replay.driver.OracleXADataSourceImplデータ・ソースを使用して、JDBC接続を取得する必要がありま

す。これらのデータ・ソースは、すべてのOracle JDBCデータソース(oracle.jdbc.pool.OracleDataSourceなど)のプロパティおよび構成パラメータをすべてサポートしています。

OCIベースのアプリケーション(SQL*Plus、ODP.NET、OCIドライバ12.2以降など)は、アプリケーション・コンティニュイティをサポートしています。

接続URLの使用時には次の点に注意する必要があります。

- データベースのREMOTE_LISTENER設定がクライアントのADDRESS_LISTのアドレスと一致しない場合、接続は行われず、「サービスが見つかりません」と表示されます。このため、データベースのREMOTE_LISTENER設定はクライアントのADDRESS_LIST内のアドレスと一致する必要があります。
    - 接続文字列がSCAN名を使用する場合は、REMOTE_LISTENERにSCAN名を設定する必要があります。
    - 接続文字列がホストVIPのADDRESS_LISTを使用する場合は、REMOTE_LISTENERにすべてのSCAN VIPとすべてのホストVIPを含むアドレス・リストを設定する必要があります

ノート:

場所に依存しない SCAN を使用すると、ノードの追加時や削除時または別のノードで実行するためのデータベースの変更時に、クライアントを再構成する必要がなくなります。

- 接続文字列にRETRY_COUNT、RETRY_DELAY、CONNECT_TIMEOUTおよびTRANSPORT_CONNECT_TIMEOUTパラメータを設定します。これらの設定により、ランタイム時、リプレイ時、および計画済停止の作業排出時に新規接続の取得効率が向上します。

  CONNECT_TIMEOUTは、sqlnet.oraファイル内のSQLNET.OUTBOUND_CONNECT_TIMEOUTパラメータと等価であり、完全接続に適用されます。TRANSPORT_CONNECT_TIMEOUTパラメータは、アドレスごとに適用されます。

- CONNECT_TIMEOUTを上限値に設定して、過剰なログインを防止します。低い値にすると、ログイン・ストームが発生して、アプリケーションやサーバー・プールが取消しと再試行を繰り返す可能性があります。(RETRY_COUNT+1)*RETRY_DELAYまたはCONNECT_TIMEOUTは、レスポンス時間のSLAよりも大きな値に設定しないでください。アプリケーションは、レスポンス時間のSLAの範囲内で接続するか、エラーを受信する必要があります。

- Oracle Databaseリリース19c以降、高可用性機能があるため、簡易接続構文を使用できます。たとえば:

```
primary-vip,secondary-vip:1521/sales.example.com?connect_timeout=90&transport_connect_timeout=3&retry_count=30&retry_delay=3
```

例6-6 ONSのTNSエントリの例

次に、Transparent Network Substrate (TNSエントリ)の例を示します。これは、Oracle Notification Service (ONS)を自動構成する際に必須のTNS書式です。ONSは、高速アプリケーション通知(FAN)に使用されるトランスポート・システムです。アプリケーション・コンティニュイティでFANを使用して、高速の停止検出を提供することをお薦めします。

```
myAlias=(DESCRIPTION=
    (CONNECT_TIMEOUT=90)(RETRY_COUNT=30)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
    (ADDRESS_LIST=
       (LOAD_BALANCE=ON)
       (ADDRESS=(PROTOCOL=TCP)(HOST=RAC-scan)(PORT=1521)))
    (ADDRESS_LIST=
       (LOAD_BALANCE=ON)
       (ADDRESS=(PROTOCOL=TCP)(HOST=DG-Scan)(PORT=1521)))
```

```
(CONNECT_DATA=(SERVICE_NAME=service_name))
```

関連項目

- tnsnames.oraファイル内のローカル・ネーミング・パラメータ
- Oracle Data Provider for .NETのインストールおよび構成

アプリケーション・コンティニュイティのためのOracle Databaseの構成

アプリケーション・コンティニュイティを使用するには、Oracle Database構成に次が含まれている必要があります。

- Oracle Real Application Clusters (Oracle RAC)、Oracle RAC One Node、Oracle Data GuardまたはOracle Active Data Guardを使用している場合、Oracle Database 12cのプールおよびドライバと通信するためにOracle Notification Service (ONS)とともにFANが構成されていることを確認します。
- リプレイおよびロード・バランシングにサービスのサービス属性を設定します。たとえば、次を設定します。
  - FAILOVER_TYPE = AUTO | TRANSACTION: 透過的アプリケーション・コンティニュイティの場合、FAILOVER_TYPE=AUTOを使用するか、手動アプリケーション・コンティニュイティの場合、FAILOVER_TYPE=TRANSACTIONを使用します。この属性は、リプレイ・ドライバおよびアプリケーション・コンティニュイティのリプレイ機能を有効にします。Oracleドライバは、データベース・セッション中に発行されたすべてのリプレイ可能な文を追跡します。すべての文がリプレイ可能で、進行中のトランザクションがコミットしなかったか、セッションが対話中の場合、Oracleでは計画済または計画外のデータベース停止の後にコミットしていない作業をリプレイします。このモードでは、追加のアプリケーション・ステップを使用しないで自動的にトランザクション状態および非トランザクション状態を再確立します。
  - REPLAY_INITIATION_TIMEOUT = n: リプレイの開始を許可するまでの期間(秒数)を設定する場合(nには、必要に応じて60、300、900、1800などを指定できます)。
  - FAILOVER_RETRIES = 30: リプレイごとに接続の再試行回数を指定する場合
  - FAILOVER_DELAY = 10: 接続の再試行間の遅延時間を秒単位で設定する場合
  - GOAL = SERVICE_TIME: Oracle RACまたはOracle Global Data Servicesを使用する場合の推奨設定です
  - CLB_GOAL = SHORT: Oracle RACまたはOracle Global Data Servicesを使用する場合の推奨設定です
  - COMMIT_OUTCOME = TRUE: トランザクション・ガードを使用する場合
  - FAILOVER_RESTORE = AUTO | LEVEL1: 透過的アプリケーション・コンティニュイティにFAILOVER_RESTORE=AUTOを、手動アプリケーション・コンティニュイティにFAILOVER_RESTORE=LEVEL1を使用します。リプレイの開始前に、接続プールに事前設定されているクライアント状態を自動的にリストアするには—AUTOCOMMIT状態(JavaおよびSQL*Plus用)、NLS状態およびTAGS (MODULE、ACTION、ECID、CLIENT_ID、CLIENT_INFO)状態を含む。
- アプリケーション・コンティニュイティを使用してフェイルオーバーさせるデータベース・ユーザーに、アプリケーション・コンティニュイティ・パッケージの権限(DBMS_APP_CONT)を付与します。次に例を示します。

```
GRANT EXECUTE ON DBMS_APP_CONT TO user_name;
```

- DB_NAMEまたはDB_UNIQUE_NAMEに対応するデータベース・サービスは使用しないでください。また、高可用性のデフォルト・データベース・サービスも使用しないでください。このサービスは有効化および無効化できず、Oracle RACで再配置

することもOracle Data Guardに切り替えることもできないためです。このサービスは、Oracle Enterprise Manager Cloud Control (Cloud Control)およびDBA用として予約されています。

アプリケーション・コンティニュイティのリプレイ前の初期状態の確立

一部のアプリケーションは、接続を使用できるようにするために、まず、初期状態を設定します。

アプリケーション・コンティニュイティは、この初期状態をリプレイの開始前に確立する必要があります。このようなアプリケーションの場合は、FAILOVER_RESTOREによって、ここに示した一般的な状態をリストアします。アプリケーションで事前設定される状態がここにリストされていない場合で、アプリケーションが初期状態を必要とする場合は、別のコールバックを追加する必要があります。

関連項目:

各リリースでリストアされるパラメータが増えるため、ご使用のプラットフォーム用の『Oracle Databaseリリース・ノート』

次に、事前設定が可能な状態の例を示します。

- PL/SQLパッケージの状態
- NLS設定
- オプティマイザ設定

リクエスト時に、アプリケーション・コンティニュイティはリクエストの状態全体を再確立します。この前提条件は、アプリケーション・コンティニュイティがリプレイを開始する前の初期状態のためのものです。

必要とされるすべての状態がFAILOVER_RESTOREによってリストアされる場合、コールバックは不要です。これは、ほとんどすべてのアプリケーションに当てはまります。

この項のトピックは、リクエストの開始時にのみ状態を設定するアプリケーションや、事前設定された状態で接続を使用することでパフォーマンスが向上するステートフル・アプリケーションに適用されます。

- FAILOVER_RESTORE
- FAILOVER_RESTOREによってリストアされる状態
- FAILOVER_RESTORE拡張
- FAILOVER_RESTOREのためのキーストアの構成
- FAILOVER_RESTOREのためのウォレットとSQLNET.ORAの構成
- FAILOVER_RESTORE = NONEおよびコールバックなし
- 接続ラベリング
- 接続初期化コールバック

FAILOVER_RESTORE

FAILOVER_RESTOREをLEVEL1(手動アプリケーション・コンティニュイティの場合)またはAUTO(透過的アプリケーション・コンティニュイティの場合)に設定すると、リクエストのリプレイ前に一般的な状態の初期設定が自動的にリストアされます。

FAILOVER_RESTOREは、サービスの設定です。Oracle Database 12.2以降で使用可能なFAILOVER_RESTOREにより、クライアント側のアプリケーションに使用可能なすべてのセッション・ステートが自動的にリストアされます。

すべてのアプリケーションでFAILOVER_RESTOREをLEVEL1またはAUTOに設定することをお薦めします。

リストアされるクライアント側のセッション状態については、「FAILOVER_RESTOREによってリストアされる状態」を参照してください。

FAILOVER_RESTOREによってリストアされる状態

このトピックでは、FAILOVER_RESTOREがLEVEL1またはAUTOに設定されている場合に、リストアされるセッション状態とサポートされないセッション状態を示します。

リストアされるセッション状態

- NLS_CALENDAR
- NLS_CURRENCY
- NLS_DATE_FORMAT
- NLS_DATE_LANGUAGE
- NLS_DUAL_CURRENCY
- NLS_ISO_CURRENCY
- NLS_LANGUAGE
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP
- NLS_NUMERIC_CHARACTER
- NLS_SORT
- NLS_TERRITORY
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- TIME_ZONE
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT
- CURRENT_SCHEMA
- MODULE
- ACTION
- CLIENT_ID
- AUTOCOMMIT状態(JavaおよびSQL*Plusの場合)
- CONTAINER (PDB)およびSERVICE
- ROLES (引き続きコールバックを必要とするセキュア・ロールは除く)
- ROW_ARCHIVAL
- EDITION
- ERROR_ON_OVERLAP_TIME
- SQL_TRANSLATION_PROFILE
- CLIENT_INFO(JDBC)

FAILOVER_RESTORE=AUTOではリストアされないセッション状態

次のものはTHINドライバではサポートされていません。そのため、自動リストア・オプションから除外されます。

- NLS_COMP
- CALL_COLLECT_TIME
- CLIENT_INFO

FAILOVER_RESTORE拡張

Oracle Database 19.5およびOracleクライアント・ドライバ19.5以降では、アプリケーションが共通のクライアント側セッション状態以外のセッション状態を使用する場合、FAILOVER_RESTOREは、リクエストがリプレイされる前にALTER SESSIONで設定したすべてのセッション・パラメータをリストアします。

フェイルオーバー時に、拡張FAILOVER_RESTOREは、セッションで変更されたセッション・パラメータをリストアします。リストアされるセッション・パラメータには、セッションで設定されたoptimizer_capture_sql_plan_baselinesやcreate_stored_outlinesなどがあります。

セッション・パラメータのリストアにログオン・トリガー、接続ラベルまたはコールバックをすでに使用している場合は、それらを引き続き使用できます。ラベルとコールバックは、拡張FAILOVER_RESTOREの有無にかかわらず完全にサポートされています。拡張FAILOVER_RESTOREの使用には、アプリケーションの変更に合わせた更新の必要がないという利点があります。

この機能を使用するには、FAILOVER_RESTOREをLEVEL1またはAUTOに設定して、ディクショナリ資格証明がシステムで暗号化されるようにする必要があります。

ディクショナリ資格証明の暗号化用にウォレットまたはキーストアを追加する方法は2つあります。

- 推奨: WALLET_ROOTデータベース・インスタンス初期化パラメータを使用して、ウォレットの場所を指定します。ウォレットの場所に初期化パラメータを使用すると、Oracle Real Application Clusters (Oracle RAC)とOracle Data Guardの間の一貫性が確保されます。この方法には、データベースのローリング再起動が必要です。
- ウォレットの場所を指すように、データベース・サーバーのTNS_ADMINディレクトリ内にあるsqlnet.oraファイルを変更します。この方法ではデータベースの再起動は必要ありません。ただし、Microsoft Windowsオペレーティング・システムでデータベースを実行している場合は再起動が必要です。すべてのORACLE_HOMEディレクトリでsqlnet.oraファイルの一貫性が保たれていることを確認する必要があります。また、データベースのアップグレードを実行するときには、sqlnet.oraに追加のメンテナンスが必要になる場合もあります。

関連項目

- Oracle Databaseのアップグレード時のOracle Net Servicesに関する推奨事項
- アプリケーション・コンテキストを使用したユーザー情報の取得
- 接続初期化コールバック

FAILOVER_RESTOREのためのキーストアの構成

次のステップを使用して、FAILOVER_RESTOREで使用するためのソフトウェア・キーストア(ウォレット)と透過的データ暗号化(TDE)を使用してディクショナリ資格証明の暗号化を構成します。

1. Oracle Autonomous Databaseを使用している場合は、ここに示すステップを実行する必要はありません。Oracle Autonomous Databaseの場合、すでにソフトウェア・キーストアが存在していて、ディクショナリ資格証明が暗号化されています。
2. Oracle Autonomous Databaseを使用していない場合は、すでにシステムがディクショナリ資格証明の暗号化を強制するように構成されているかどうかを確認します。
    a. 次のSQL問合せを使用して、ウォレット(キーストア)が存在していることを確認します。

```
SELECT con_id, wrl_type, status , wallet_type FROM V$ENCRYPTION_WALLET
ORDER BY con_id;
    CON_ID WRL_TYPE     STATUS    WALLET_TYPE
```

```
---------- ------------ -------- -----------
         0 FILE         OPEN     PASSWORD
```

このSQL問合せで行が戻されない場合は、ウォレット(キーストア)が存在していません。

b. 次のSQL問合せを使用して、ディクショナリ資格証明が暗号化されていることを確認します。

```
SQL> SELECT enforcement FROM DICTIONARY_CREDENTIALS_ENCRYPT;
ENFORCEMENT
---------------
ENABLED
```

このSQL問合せでDISABLEDが返される場合、ディクショナリは暗号化されません。

ウォレットと暗号化したディクショナリ資格証明の準備が完了していると、サービスの属性を設定することで拡張FAILOVER_RESTOREを使用できます。この手順の以降のステップを完了する必要はありません。

既存のウォレットがない場合やディクショナリ資格証明の暗号化を有効にする必要がある場合は、次のステップに進みます。

3. ソフトウェア・キーストアを使用するようにデータベースを構成します。

次のステップは、SYSKM権限を持つオペレータが実行する必要があります。オペレータ・ユーザーにロールSYSKMを付与します。

a. 必要に応じて、ウォレットを格納するディレクトリを作成します。

選択した場所はOracle RACノード間で共有して、Oracle Data Guardサイトにレプリケートする必要があります。Oracle RACの場合、ディレクトリは共有記憶域上にあることが必要です。

b. 静的な初期化パラメータWALLET_ROOTを変更します。
パラメータの値は、ウォレットが格納されているディレクトリにする必要があります。

```
ALTER SYSTEM SET WALLET_ROOT='/myOracleBase/admin/wallet/' SCOPE=spfile;
```

c. 初期化パラメータTDE_CONFIGURATIONを変更して、ソフトウェア・キーストアを指定します。

```
ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=FILE" SCOPE=BOTH SID='*'
```

d. データベース・インスタンスのローリング再起動を実行し、新しい初期化パラメータをアクティブ化します。

たとえば、orclという名前の2ノード・クラスタ・データベースがあり、インスタンス名がorcl1とorcl2の場合は、次のコマンドを使用して、データベースが完全に停止しないように各インスタンスを個別に停止および再起動します。

```
$ srvctl stop instance -db orcl -instance orcl1 -drain_timeout 600 -stopoption
IMMEDIATE
$ srvctl start instance -db orcl -instance orcl1

srvctl stop instance -db orcl -instance orcl2 -drain_timeout 600 -stopoption IMMEDIATE
srvctl start instance -db orcl -instance orcl2
```

ノート:

フリート・パッチ適用およびプロビジョニングを使用すると、このプロセスが自動化されます。また、

パッチ・アップグレード時にパラメータを変更する場合は、かわりに使用できます。

e. インスタンスの再起動後に、パラメータが正しい値に設定されていることを確認します。

```
SQL> SHOW PARAMETER WALLET_ROOT;
SQL> SHOW PARAMETER TDE_CONFIGURATION;
```

4. パスワードを使用してキーストアを作成します(まだキーストアが存在しない場合)。

次の例では、passwordはキーストアのパスワードです。パスワードの大文字と小文字は区別されます。キーストアのパスワードはデータベース・ユーザーのパスワードと同じ規則に従います。

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "password";
```

5. キーストアを開いて、暗号化キーを設定します。

データベースがOracle Multitenantデータベースとして構成されている場合は、CONTAINER=all句を使用して、各PDBにキーストアと暗号化キーを設定する必要があります。次の例では、passwordはキーストアのパスワードです。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password" CONTAINER=all;
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password"
WITH BACKUP CONTAINER=all;
```

データベースがOracle Multitenantデータベースとして構成されていない場合は、次のSQLコマンドを使用します。passwordは、キーストアのパスワードです。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password"
WITH BACKUP;
```

6. データベース・ディクショナリ資格証明を暗号化します。

SYSKMロールを持つ演算子を使用して、コンテナ・データベース(CDB)ルートと各PDBから次のSQLコマンドを実行します。

```
ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;
```

情報の暗号化と復号化は、フェイルオーバーのリストア中にサーバーで自動的に実行されます。

警告:

ソフトウェア・キーストアとウォレットの場所をバックアップするようお薦めします。TDE ソフトウェア・キーストアまたは WALLET_ROOT の場所を失わないようにしてください。そうしたときに、アプリケーション・コンティニュイティまたは透過的アプリケーション・コンティニュイティの場合、新しいキーストアは作成できますが、暗号化したディクショナリ資格証明は再インスタンス化が必要になります。ウォレット・キーに不一致があると、フェイルオーバーは成功しません。

関連項目

- ソフトウェア・キーストアの構成
- キーストアおよびマスター暗号化キーの管理
- Oracle Database製品で許可される機能、オプションおよび管理パック

FAILOVER_RESTOREのためのウォレットとSQLNET.ORAの構成

次のステップを使用して、FAILOVER_RESTOREで使用するためのウォレットの場所を指すためにSQLNET.ORAを使用してディクショナリ資格証明の暗号化を構成します。

この方法ではデータベースの再起動は必要ありません。ただし、Microsoft Windowsオペレーティング・システムでデータベースを実行している場合は再起動が必要です。すべてのORACLE_HOMEディレクトリでsqlnet.oraファイルの一貫性が保たれていることを確認する必要があります。

1. Oracle Autonomous Databaseを使用している場合は、ここに示すステップを実行する必要はありません。Oracle Autonomous Databaseの場合、すでにソフトウェア・キーストアが存在していて、ディクショナリ資格証明が暗号化されています。
2. Oracle Autonomous Databaseを使用していない場合は、すでにシステムがディクショナリ資格証明の暗号化を強制するように構成されているかどうかを確認します。
    a. 次のSQL問合せを使用して、ウォレットが存在していることを確認します。

```
SELECT con_id, wrl_type, status , wallet_type FROM V$ENCRYPTION_WALLET
ORDER BY con_id;
    CON_ID WRL_TYPE     STATUS   WALLET_TYPE
---------- ------------ -------- ------------
         0 FILE         OPEN     PASSWORD
```

    このSQL問合せで行が戻されない場合は、ウォレット(キーストア)が存在していません。

    b. 次のSQL問合せを使用して、ディクショナリ資格証明が暗号化されていることを確認します。

```
SQL> SELECT enforcement FROM DICTIONARY_CREDENTIALS_ENCRYPT;
ENFORCEMENT
---------------
ENABLED
```

    このSQL問合せでDISABLEDが返される場合、ディクショナリは暗号化されません。

    ウォレットと暗号化したディクショナリ資格証明の準備が完了していると、サービスの属性を設定することで拡張FAILOVER_RESTOREを使用できます。この手順の以降のステップを完了する必要はありません。

    既存のウォレットがない場合やディクショナリ資格証明の暗号化を有効にする必要がある場合は、次のステップに進みます。

3. ウォレットを使用するようにデータベースを構成します。
    a. TNS_ADMIN環境変数を表示して、データベースで使用されるネットワーク構成ファイルの場所を探します。
        - LinuxおよびUNIXシステムでは、Oracleホーム・ソフトウェア所有者として、TNS_ADMIN環境変数の現在の設定を表示します。

```
$ env | grep TNS_ADMIN
```

        - Microsoft Windowsシステムでは、環境変数とレジストリ(パスComputer¥HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥KEY_HOME_NAME)の両方でTNS_ADMINの値セットを確認します。

    TNS_ADMIN変数が設定されない場合は、Oracle Net構成ファイルにデフォルトの場所の$ORACLE_HOME¥network¥adminが使用されます。

    b. 必要に応じて、ウォレットを格納するディレクトリを作成します。

選択した場所はOracle RACノード間で共有して、Oracle Data Guardサイトにレプリケートする必要があります。Oracle RACの場合、ディレクトリは共有記憶域上にあることが必要です。

c. SQLNET.ORAファイルを探して編集します。
前述のサブステップで取得した場所を使用して、sqlnet.oraファイルを編集し、次のエントリを追加します。/myOracleWalletLocは、ウォレットを格納するために作成したディレクトリの完全パス名です。

```
ENCRYPTION_WALLET_LOCATION =
  (SOURCE=
   (METHOD=FILE)
    (METHOD_DATA=
     (DIRECTORY=/myOracleWalletLoc)))
```

d. 初期化パラメータTDE_CONFIGURATIONを変更して、ソフトウェア・キーストアを指定します。

```
ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=FILE" SCOPE=BOTH SID='*'
```

4. パスワードを使用してキーストアを作成します(まだキーストアが存在しない場合)。

次の例のmyOracleWalletLocは、ウォレット(またはキーストア)を格納するために作成したディレクトリのフルパス名です。passwordは、キーストアのパスワードです。パスワードの大文字と小文字は区別されます。キーストアのパスワードはデータベース・ユーザーのパスワードと同じ規則に従います。

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/myOracleWalletLoc' IDENTIFIED BY "password";
```

5. キーストアを開いて、暗号化キーを設定します。

データベースがOracle Multitenantデータベースとして構成されている場合は、CONTAINER=all句を使用して、各PDBにキーストアと暗号化キーを設定する必要があります。次の例では、passwordはキーストアのパスワードです。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password" CONTAINER=all;
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password"
WITH BACKUP CONTAINER=all;
```

データベースがOracle Multitenantデータベースとして構成されていない場合は、次のSQLコマンドを使用します。passwordは、キーストアのパスワードです。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password"
WITH BACKUP;
```

6. データベース・ディクショナリ資格証明を暗号化します。

SYSKMロールを持つ演算子を使用して、コンテナ・データベース(CDB)ルートと各PDBから次のSQLコマンドを実行します。

```
ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;
```

情報の暗号化と復号化は、フェイルオーバーのリストア中にサーバーで自動的に実行されます。

警告:

ウォレットの場所は、バックアップすることをお薦めします。ウォレットまたは場所は失われないようにしてください。そうし

たときに、アプリケーション・コンティニュイティまたは透過的アプリケーション・コンティニュイティの場合、新しいウォレットは作成できますが、暗号化したディクショナリ資格証明は再インスタンス化が必要になります。ウォレット・キーに不一致があると、フェイルオーバーは成功しません。

関連項目

- Oracle Net Services構成ファイルの保存場所
- sqlnet.oraを使用した透過的データ暗号化キーストアの構成
- Oracle Database製品で許可される機能、オプションおよび管理パック

FAILOVER_RESTORE = NONEおよびコールバックなし

このシナリオは、Oracle Database 18cより前のデータベースおよびクライアントに適用できますが、アプリケーションはプールからの接続を流用するときに、どのような状態も想定しません。また、初期状態を再確立するためにUCPやWebLogicラベルを使用します。

Oracle Database 18c以降のデータベースおよびクライアントでは、すべてのアプリケーションに対してFAILOVER_RESTOREをLEVEL1またはAUTOに設定することをお薦めします。

接続ラベリング

汎用プール機能である接続ラベリングを使用することをベスト・プラクティスとしてお薦めします。接続ラベリングが存在する場合、アプリケーション・コンティニュイティはこれを使用します。接続ラベリングが状態を再作成するため、FAILOVER_RESTOREをNONEに設定できます。

このシナリオは、Universal Connection Pool (UCP)およびOracle WebLogic Serverに適用できます。接続に事前設定された状態を活用するようアプリケーションを変更できます。接続ラベリングAPIにより、接続の一致状況が判別され、接続が流用される際にコールバックを使用してギャップが移入されます。

関連項目

- Oracle Universal Connection Pool開発者ガイド

接続初期化コールバック

このシナリオでは、リプレイ・ドライバ(JDBCまたはOCI)がアプリケーション・コールバックを使用して、実行時およびリプレイ中にセッションの初期状態を設定します。JDBCリプレイ・ドライバには、oracle. jdbc. replay. OracleDataSourceインタフェースで接続初期化コールバックを登録および登録解除するために、接続初期化コールバックのインタフェースおよびメソッドが用意されています。OCIとODP.NETの場合は、TAFコールバックを登録します。

登録されると、接続がプールから流用されるたびに、またリカバリ可能なエラー後に再接続が成功するたびに、初期化コールバックは実行されます。(これは、JDBC/UCP接続初期化コールバックの場合trueであり、TAFに対して同じにする必要があります。)ランタイムとリプレイの両方で同じコールバックを使用すると、セッションが最初に確立されたときと同じ初期化がリプレイ時に確立されます。アプリケーションは、初期化アクションがフェイルオーバー前の元の接続時のものと同じであることを確認する必要があります。コールバックの起動が失敗した場合、リプレイはその接続で無効になります。接続初期化コールバックは、アプリケーションにUCPとWebLogic接続ラベリングが実装されていないため、FAILOVER_RESTORE=AUTO(透過的アプリケーション・コンティニュイティの場合)またはFAILOVER_RESTORE=LEVEL1(手動アプリケーション・コンティニュイティの場合)の設定では自動的に

状態がリストアできない場合にのみ使用します。

アプリケーション・コンティニュイティでの再接続の遅延

デフォルトでは、アプリケーション・コンティニュイティがフェイルオーバーを開始した場合、ドライバが、サービスを利用できるインスタンスで実行中の作業のリカバリを試みます。

作業をリカバリするために、ドライバはインスタンスとの良好な接続を確立する必要があります。サービスが再配置および発行される前にデータベースまたはインスタンスを再起動する必要がある場合、再接続に時間がかかる場合があります。したがって、サービスが別のインスタンスまたはデータベースから利用可能になるまでに、フェイルオーバーを遅延させる必要があります。

接続と再接続を管理するには、FAILOVER_RETRIESおよびFAILOVER_DELAYパラメータを使用する必要があります。これらのパラメータは計画済停止と連動して使用すると効果があります。たとえば、サービスが数分間使用不可になる可能性がある停止の場合などです。FAILOVER_DELAYおよびFAILOVER_RETRIESパラメータを設定する場合は、REPLAY_INITIAITION_TIMEOUTパラメータの値を最初に確認します。このパラメータのデフォルト値は900秒です。FAILOVER_DELAYパラメータの値が高い場合、リプレイが取り消されることがあります。

パラメータ名	使用可能な値	デフォルト値
FAILOVER_RETRIES	正の整数ゼロ以上	30
FAILOVER_DELAY	秒数	10

次の例は、様々なフェイルオーバー・シナリオを示します。

- アプリケーション・コンティニュイティを使用するOracle RACのサービスの作成
- 単一インスタンスのデータベースのサービスがアプリケーション・コンティニュイティを使用するように変更する方法

アプリケーション・コンティニュイティを使用するOracle RACのサービスの作成

透過的アプリケーション・コンティニュイティまたは手動アプリケーション・コンティニュイティを利用するサービスをOracle RACで作成できます。

次のように、透過的アプリケーション・コンティニュイティを使用するサービスを作成できます。

ポリシー管理データベースの場合:

```
$ srvctl add service -db mydb -service TACSERVICE -serverpool ora.Srvpool -clbgoal SHORT
  -rlbgoal SERVICE_TIME -failovertype AUTO -failover_restore AUTO -commit_outcome TRUE --
replay_init_time 600
  -retention 86400 -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE
```

管理者管理データベースの場合:

```
$ srvctl add service -db mydb -service TACSERVICE -pdb mypdb -preferred inst1 -available inst2
  -failovertype AUTO -failover_restore AUTO -commit_outcome TRUE --replay_init_time 600 -retention
86400
  -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE -role PRIMARY
```

次のように、手動アプリケーション・コンティニュイティを使用するサービスを作成できます。

ポリシー管理データベースの場合:

```
$ srvctl add service -db mydb -service ACSERVICE -serverpool ora.Srvpool -failovertype TRANSACTION
  -failover_restore LEVEL1 -commit_outcome TRUE -session_state dynamic -replay_init_time 600 -
retention 86400
  -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE
```

管理者管理データベースの場合:

```
$ srvctl add service -db mydb -service ACSERVICE -pdb mypdb -preferred inst1 -available inst2
  -failovertype TRANSACTION -failover_restore LEVEL1 -commit_outcome TRUE -session_state dynamic -
replay_init_time 600
  -retention 86400 -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE -role PRIMARY
```

単一インスタンスのデータベースのサービスがアプリケーション・コンティニュイティを使用するように変更する方法

単一インスタンス・データベースを使用している場合は、次に示すようにDBMS_SERVICEパッケージを使用してサービスを変更します。

手動アプリケーション・コンティニュイティの場合:

```
DECLARE
params dbms_service.svc_parameter_array;
BEGIN
params('FAILOVER_TYPE'):='TRANSACTION';
params('REPLAY_INITIATION_TIMEOUT'):=1800;
params('RETENTION_TIMEOUT'):=86400;
params('FAILOVER_DELAY'):=10;
params('FAILOVER_RETRIES'):=30;
params('FAILOVER_RESTORE'):='LEVEL1';
params('commit_outcome'):='true';
params('aq_ha_notifications'):='true';
dbms_service.modify_service('[your service]',params);
END;
/
```

透過的アプリケーション・コンティニュイティ:

```
DECLARE
params dbms_service.svc_parameter_array;
BEGIN
params('FAILOVER_TYPE'):='AUTO';
params('REPLAY_INITIATION_TIMEOUT'):=1800;
params('RETENTION_TIMEOUT'):=86400;
params('FAILOVER_DELAY'):=10;
params('FAILOVER_RETRIES'):=30;
params('FAILOVER_RESTORE'):='AUTO';
params('commit_outcome'):='true';
params('aq_ha_notifications'):='true';
dbms_service.modify_service('[your service]',params);
END;
/
```

計画メンテナンスに対するアプリケーション・コンティニュイティの使用

計画メンテナンスの場合、推奨されるアプローチは、完了していないリクエストについてアプリケーション・コンティニュイティと連携し、Oracle接続プールからリクエストを排出する方法です。パッチが適用されたソフトウェアにスイッチオーバーするには、インスタンスを停止する必要があります。

完了する必要があるリカバリが最小限である場合、この方法が最も影響の少ない方法です。

計画メンテナンスにアプリケーション・コンティニュイティを使用するには:

1. FAN対応プール(OCI、UCP、WebLogic Server、ODP.NET管理ドライバまたは管理対象外ドライバ)を使用します。

   FAN計画イベントの場合、リクエスト境界で排出します。

   ノート:

   ODP.NET 管理ドライバは、アプリケーション・コンティニュイティをサポートしていません。
2. `srvctl relocate service`コマンドを使用して、セッションを中断することなくインスタンスのサービスを再配置します。または、均一サービスの場合、インスタンスで`srvctl stop service`コマンドを使用します(`-force`パラメータは使用しないでください)。

   FANの計画イベントはアイドル・セッションを即時にクリアして、アクティブ・セッションをチェックイン時(リクエストの最後)に解放するとマーク付けします。これによって作業を妨げずにセッションがインスタンスから排出されます。
3. 一部のセッションがチェックインしておらず、インスタンスを停止する時間になったら、インスタンスを停止(中断)します。

   アプリケーション・コンティニュイティが有効なプール(UCP、WebLogic、Tuxedo、ODP.NET、およびOCI)、および`beginRequest/endRequest`を追加するJavaプールの場合、アプリケーション・コンティニュイティは、そうした残っているセッションをリカバリしようとします。
4. インスタンスおよびサービスを再起動します。

   実行時ロード・バランシング(がもし有効な場合)では、次のリクエスト境界でリストアされたインスタンスにセッションを戻して均衡化します。

アプリケーション・コンティニュイティなしでの実行

無効化コールが発行されたために、アプリケーション・コンティニュイティが有効でない場合もあります。

アプリケーション・コンティニュイティは、起動されていない場合や無効化されている場合は有効ではありません。無効化されている場合、`endRequest`コールを介して無効のままにされます。

サービス・プロパティ`FAILOVER_TYPE`の値が`TRANSACTION`または`AUTO`に設定されていない場合、アプリケーション・コンティニュイティは起動されません。計画メンテナンスの場合、`FAILOVER_TYPE`の値を事前に`TRANSACTION`または`AUTO`に設定します。この設定が新しい接続に適用され、既存の接続は元のサービス値のままになります。

アプリケーション・コンティニュイティは、次のいずれかが発生したときには、現在のリクエストに対して無効化されます。

- アプリケーション・コンティニュイティが制限されている文をアプリケーションが実行する(たとえば、`ALTER SYSTEM`)。
- `disableReplay`を使用してアプリケーション・コンティニュイティが明示的に無効化されます。
- サービス・パラメータ`session_state_consistency`が`Dynamic` (透過的アプリケーション・コンティニュイティを使用しな

い場合はデフォルト)に設定されているときにCOMMIT文が発行されます。

- 次のbeginRequestが発行されるまでendRequest文が発行されます。
- セッションが終了または切断され、NOREPLAYキーワードが指定されます。

関連項目

- 透過的アプリケーション・コンティニュイティ
- アプリケーション・コンティニュイティにおけるリプレイの無効化
- リプレイなしのセッションの終了または切断

アプリケーション・コンティニュイティにおけるリプレイの無効化

リプレイはリカバリ可能なエラーの後に実行されますが、リプレイを無効にできます。

繰り返す必要のないリクエストがアプリケーションにある場合、アプリケーションはアプリケーション・コンティニュイティが有効化されていないサービスへの接続を取得するか、これらのリクエストのリプレイを無効にするAPIを明示的に呼び出すことができます。透過的アプリケーション・コンティニュイティを使用する場合、副作用は自動的に検出され、無効化されます。アプリケーションを理解したり、副作用を含むリクエストを無効にする必要はありません。

手動アプリケーション・コンティニュイティを使用する場合、すべてのコールがリプレイされます。たとえば、アプリケーションがUTL_SMTPを使用しているためにメッセージを繰り返す必要がない場合は、アプリケーションでは別のサービスへの接続を使用するか、disableReplay API (Javaの場合)またはOCIRequestDisableReplay API (OCIの場合)を使用できます。他のすべてのリクエストは引き続きリプレイされます。

外部アクション(自律型トランザクション、またはUTL_HTTPを使用したSOAコールを発行するトランザクションの使用など)の場合、障害後にこれらの外部アクションがリプレイされるときにアプリケーションの正確性が保持されている場合、アプリケーション・コンティニュイティは透過的のままです。

次に、汎用的なルールを示します。これらは、アプリケーション・コンティニュイティおよびTAF (リリース12.2以降)が含まれていて、作業をリプレイするアプリケーションのすべてに適用されます。

- 繰り返さない自律型トランザクション、外部PL/SQLまたはJavaアクションをアプリケーションがコールする場合
- アプリケーションが独立セッションを同期化する場合
- アプリケーションが実行ロジックで中間層の時刻を使用する場合
- ROWIDが変更されないことをアプリケーションが前提としている場合
- ロケーション値が変更されないことをアプリケーションが前提としている場合

関連項目

- 透過的アプリケーション・コンティニュイティ
- アプリケーション・コンティニュイティの潜在的な副作用
- アプリケーション・コンティニュイティに関する制限および他の考慮事項

繰り返さない自律型トランザクション、外部PL/SQLまたはJavaアクションをアプリケーションがコールする場合

自律型トランザクション、外部PL/SQLコールおよびJavaコールアウトには、メイン・トランザクションとは異なる副作用がある場合があり、これらの副作用は、リプレイされないよう指定しないかぎりリプレイされます。

メイン・トランザクションとは異なる副作用の例には、外部表への書込み、電子メールの送信、PL/SQL (UTL_HTTP、UTL_URL、UTL_FILE、UTL_FILE_TRANSFER、UTL_SMPT、UTL_TCP、UTL_MAIL、DBMS_PIPEまたはDBMS_ALERTへのコールを含む)またはJava (フォームProcess proc = rt.exec(command);でのシェル・スクリプトの実行を含む)からのセッションの分岐、ファイルの転送、および外部URLへのアクセスなどがあります。このようなアクションの結果、永続的な副作用が残ります。PL/SQLメッセージングおよびJavaコールアウトの結果、永続的な結果が残される可能性があります。たとえば、ユーザーがコミットせずに作業の途中で席を離れてセッションがタイムアウトするか、ユーザーが[Ctrl]を押しながら[C]を押すと、フォアグラウンドまたはコンポーネントが失敗します。メイン・トランザクションがロールバックし、その間に副作用が適用されることがあります。(副作用の詳細は、「アプリケーション・コンティニュイティの潜在的な副作用」を参照してください。)

アプリケーション開発者は、外部アクションに対してリプレイを許可するかどうかを決定します。この例には、UTL_HTTPを使用したSOAコールの発行、UTL_SMTPを使用したメッセージの送信、またはUTL_URLを使用したWebサイトへのアクセスなどがあります。そのような外部アクションがリプレイを必要としない場合、ACのない接続を使用するか、リプレイを無効化するAPIのいずれかを使用します。

アプリケーションが独立したセッションを同期する場合

COMMIT、ROLLBACKまたはセッションの喪失まで維持される揮発エンティティを使用して、アプリケーションが独立セッションを同期する場合、リプレイ用にアプリケーションを構成しないでください。たとえば、アプリケーションが、複数のデータソースに接続されている複数のセッションを同期化することがあります(同期化されない場合、これらのセッションはデータベース・ロックなどのリソースを使用して相互に依存しています)。アプリケーションでこれらのセッションのみがシリアライズされ、いずれかのセッションが失敗する可能性があることが認識されている場合、このような同期が許容されることがあります。ただし、1つのデータソースによって保持されているロックまたは他の高揮発リソースが、他の接続から同一または別個のデータソースのデータに対する排他的アクセスを実現することをアプリケーションが想定している場合、この想定はリプレイ時に否定される可能性があります。

リプレイ中、セッションがロックまたは他の揮発リソースを維持している別のセッションに依存していることは、クライアント・ドライバでは認識されていません。また、リソース(セマフォ、デバイスまたはソケットなどの)を使用するパイプ、バッファ・キュー、ストアド・プロシージャを使用して、失敗によって失われる同期を実行することもできます。

アプリケーションが実行ロジックで中間層の時刻を使用する場合

アプリケーションが実行ロジックの一部として中間層で実時間を使用する場合は、リプレイ用にアプリケーションを構成しないでください。クライアント・ドライバは中間層の時間ロジックを繰り返しませんが、このロジックの一部として実行されるデータベース・コールを使用します。たとえば、中間層の時間を使用するアプリケーションが明示的にこれを使用していなければ、時間T1で実行された文が時間T2で再実行されていないと想定することがあります。

ROWIDが変更されないことをアプリケーションが前提としている場合

アプリケーションがROWIDをキャッシュする場合、データベースが変更されているためにROWIDへのアクセスが無効になることがあります。ROWIDが表内の一意の行を特定しても、次のような状況ではROWIDの値が変更されることがあります。

- 基礎となる表が再編成されます。
- 表で索引が作成されます。
- 基礎となる表がパーティション化されます。

- 基礎となる表が移行されます。
- 基礎となる表がEXP/IMP/DULを使用してエクスポートおよびインポートされます。
- 基礎となる表がGolden Gate、ロジカル・スタンバイまたは他のレプリケーション・テクノロジを使用して再構築されます。
- 基礎となる表のデータベースがフラッシュバックまたはリストアされます。

一般に、将来の使用のためにアプリケーションがROWIDを保存することはお薦めできません。これは、対応する行が存在しないことや、まったく異なるデータを含んでいることがあるためです。ROWIDがアプリケーション・コンティニュイティの使用を妨げることはありません。リプレイは拒否できます。

ロケーション値が変更されないことをアプリケーションが前提としている場合

SYSCONTEXTオプションは、各国語サポート(NLS)設定、ISDBA、CLIENT_IDENTIFIER、MODULEおよびACTIONなどのロケーション非依存セットと、物理ロケータを使用するロケーション依存セットで構成されています。通常、アプリケーションはテスト環境を除いて物理的な識別子を使用しません。物理ロケータがメインライン・コードで使用されている場合、リプレイによって不一致が検出され、拒否されます。ただし、リクエスト間(beginRequestの前)またはコールバックでは物理ロケータを使用してもかまいません。QAの一般的な問題は、テスト・アプリケーションがV$INSTANCEを選択するように変更することです。V$INSTANCEは変更可能なため、このチェックはコールバックにのみ配置するか、インスタンスをデータベースからではなくクライアントでローカルに選択します。

例

```
select
    sys_context('USERENV','DB_NAME')
    ,sys_context('USERENV','HOST')
    ,sys_context('USERENV','INSTANCE')
    ,sys_context('USERENV','IP_ADDRESS')
    ,sys_context('USERENV','ISDBA')
    ,sys_context('USERENV','SESSIONID')
    ,sys_context('USERENV','TERMINAL')
    ,sys_context('USERENV','SID')
from dual;
```

リプレイなしのセッションの終了または切断

アプリケーション・コンティニュイティが構成されている場合、DBAがALTER SYSTEM KILL SESSIONまたはALTER SYSTEM DISCONNECT SESSION文を使用してセッションを終了または切断すると、デフォルトではアプリケーション・コンティニュイティがセッションをリカバリしようとします。ただし、セッションをリプレイしない場合は、次に示すようにNOREPLAYキーワードを使用します。

```
alter system kill session 'sid, serial#, @inst' noreplay;
alter system disconnect session 'sid, serial#, @inst' noreplay
$ srvctl stop service -db orcl -instance orcl2 -drain_timeout 60 -stopoption immediate -force -
noreplay
$ srvctl stop service -db orcl -node myode3 -noreplay -drain_timeout 60 -stopoption immediate -force
$ srvctl stop instance -node mynode3 -drain_timeout 60 -stopoption immediate -force -noreplay
```

ローカル・インスタンスで実行している(1つのセッションのみではなく)すべてのセッションを終了して、それらのセッションがリプレイされないようにする場合は、DBMS_SERVICE.DISCONNECT_SESSION PL/SQLプロシージャを使用して、disconnect_optionパラメータにNOREPLAYを指定することもできます。

関連項目

- ALTER SYSTEM
- DBMS_SERVICE.DISCONNECT_SESSION

可変関数とアプリケーション・コンティニュイティ

リクエストがリプレイされると、可変オブジェクトのデフォルトおよび目的の処理が変化する可能性があります。

デフォルトでは、SQLの場合、受信した元の値は順序に対してリプレイされます。これは、アプリケーションが所有する値です。PL/SQL、DATEとTIME、およびSYSGUID可変の場合、KEEP句をスキーマの一部として付与する必要があります。

現在、可変関数値を保持するためのサポートは、SYSDATE、SYSTIMESTAMP、LOCAL_TIMESTAMP、CURRENT_TIMESTAMP、SYS_GUIDおよびsequence.NEXTVAL.に対して提供されています。元の値が保持されていないため、これらの可変オブジェクトの別の値がクライアントに戻されると、クライアントが認識する値が異なるため、リプレイは拒否されます。アプリケーションが元の値を使用できる場合、所有されている順序にはKEEP句を使用し、他のユーザーにはGRANT KEEPを使用して、可変関数を構成してください。(ほとんどのアプリケーションでは、バインド変数の一貫性を維持するために、リプレイ時に順序値を保持する必要があります。)

ノート:

SYS_GUID 値の保持は、シリアル処理計画に対してのみサポートされています。パラレル問合せを使用する場合、アプリケーション・コンティニュイティは、SYS_GUID の元の値をリストアできません。

次の表に、リプレイ時の可変関数の処理の例を製品別に示します。(実際の実装は特定の製品およびリリースによって異なります。)

表6-4 リプレイ時の可変オブジェクトの処理の製品別の例

可変関数	製品1	製品2	製品3
SYSDATE、SYSTIMESTAMP	オリジナル	オリジナル	現行
順序 NEXTVAL および CURRVAL	オリジナル	オリジナル	(該当なし)
SYS_GUID	オリジナル	(該当なし)	(該当なし)

アプリケーション・コンティニュイティがリプレイ時に元のファンクション結果を保持および使用できるようにするには:

- アプリケーションを実行するデータベース・ユーザーに、KEEP DATE TIMEおよびKEEP SYSGUID権限を付与し、値を保持する順序ごとにKEEP SEQUENCEオブジェクト権限を付与できます。たとえば:

```
GRANT KEEP DATE TIME TO user2;
GRANT KEEP SYSGUID TO user2;
GRANT KEEP SEQUENCE ON sales.seq1 TO user2;
```

ノート:

- Oracle Database 19c 以降、順序の SQL に対して可変を保持するための付与は必要ありません。
- GRANT ALL ON object には、KEEP DATE TIME および KEEP SYSGUID 権限と KEEP SEQUENCE オブジェクト権限は含まれません(つまり、これらによって提供されるアクセスは承認しません)。

- 可変関数サポートに関連する権限はアプリケーション・ユーザーに対してのみ付与し、各アプリケーション・ユーザーに対して必要な権限のみを付与します。

- リプレイを有効化するアプリケーションを実行するデータベース・ユーザーには、DBA 権限を付与しないでください。

- アプリケーション内の順序には、KEEP属性を使用できます。この属性は、順序所有者に対してsequence.NEXTVALの元の値を保持することにより、リプレイ時にキーが一致するようにします。ほとんどのアプリケーションでは、リプレイ時に順序値を保持する必要があります。次の例では、順序に対してKEEP属性を設定しています(この場合は、文を実行するユーザーが所有する順序ですが、他の場合はGRANT KEEP SEQUENCEを使用してください)。

```
SQL> CREATE SEQUENCE my_seq KEEP;
SQL> -- Or, if the sequence already exists but without KEEP:
SQL> ALTER SEQUENCE my_seq KEEP;
```

ノート:

ALTER SEQUENCE ... KEEP/NOKEEP の指定は、順序の所有者に適用されます。これは、KEEP SEQUENCE オブジェクト権限を持つ他のユーザー(所有者ではありません)には影響しません。すべてのユーザーに NOKEEP を指定する場合、これらのユーザーに KEEP SEQUENCE オブジェクト権限を付与しない(または、すでに権限が付与されている場合はそれを取り消さない)ようにしてください。

- リプレイ時にファンクション結果(名前付きファンクションの場合)を保持するには、ファンクションを起動するユーザーにDBAがKEEP権限を付与する必要があります。このセキュリティ制限により、ユーザーによって所有されていないコードに対するファンクション結果をリプレイのために保存およびリストアできるようになります。

アイデンティティ順序の場合、可変の保持は所有順序でサポートされます。SQLレベルでの可変の保持は、アイデンティティ順序に対して自動的に行われます。アイデンティティ順序のPL/SQLで可変を保持するには、KEEP句を使用します。プロシージャと表の定義は次のとおりです。

```
create table tab_identity_mine( id NUMBER GENERATED ALWAYS AS IDENTITY keep, content varchar2(50));
```

プロシージャを作成または置換するには、次の文を使用します。

```
insert_identity(cnt in varchar2,newid out number as
begin
insert into tab_identity_mine(content) values(cnt) returning id into newid;
end insert_identity;
```

関連項目

- 可変に対する権限のルール
- ALTER SEQUENCE

- GRANT

可変値の管理

可変値を管理するには、特定の権限を付与する必要があります。

可変に対する権限の維持の付与および取消し

リプレイで関数の結果を保持するには、関数を呼び出すユーザーにKEEP権限を付与する必要があります。

- SYSDATEおよびSYSTIMESTAMPまたはSYSGUIDの可変を維持する権限を付与するには:

  ```
  GRANT [KEEP DATE TIME | KEEP SYSGUID]...[to USER]
  ```

  たとえば、次のように、元の日付でOracle E-Business Suiteを使用できます。

  ```
  GRANT KEEP DATE TIME, KEEP SYSGUID to [custom user];
  GRANT KEEP DATE TIME, KEEP SYSGUID to [apps user];
  ```

- SYSDATEおよびSYSTIMESTAMPまたはSYSGUIDの可変を維持する権限を取り消すには、次のようにします。

  ```
  REVOKE [KEEP DATE TIME | KEEP SYSGUID]...[from USER]
  ```

Oracle順序の可変を維持するための権限の付与

キーが一致するようにリプレイでsequence.nextvalの元の値を維持するには、順序に対する権限を付与する必要があります。

- 順序の所有者としての権限を付与するには:

  ```
  CREATE SEQUENCE [sequence object] [KEEP|NOKEEP];
  ALTER SEQUENCE [sequence object] [KEEP|NOKEEP];
  ```

- 順序を使用するその他のユーザーに対して権限の付与および取消しを行うには:

  ```
  GRANT KEEP SEQUENCE on sequence.object to [myUser|role];
  REVOKE KEEP SEQUENCE on sequence.object from [myUser|role];
  ```

  たとえば、次のように、元の順序値でOracle E-Business Suiteを使用できます。

  ```
  GRANT KEEP SEQUENCE on sequence.object to apps-user;
  REVOKE KEEP SEQUENCE on sequence.object from my-user ;
  ```

  たとえば、アイデンティティ順序の場合は、表のcreateまたはalter文でKEEP句を使用します。

  ```
  CREATE TABLE tab_identity_mine(id NUMBER GENERATED ALWAYS AS IDENTITY keep,
  content varchar2(50));
  ```

可変に対する権限のルール

これらの考慮事項は、可変関数に対する権限の付与および取消しに適用されます。

- ユーザーのオブジェクトに対してすべてを付与していても、可変は除外されています。可変には明示的な付与が必要です。SYS、AUDSYS、GSMUSER、SYSTEMなど、Oracle Databaseによって提供または作成されるユーザーへの可変の付与はサポートしていません。
- DBAロールには、可変権限が含まれます。

- ユーザーに可変が付与されている場合、(SYS_GUID、SYSDATEおよびSYSTIMESTAMPで)可変関数が呼び出されたときに、オブジェクトは可変アクセスを継承します。
- 順序オブジェクトに対する可変の維持が取り消されると、そのオブジェクトを使用するSQLまたはPL/SQLコマンドは、その順序の可変コレクションまたはアプリケーションを許可しません。
- ランタイムおよびフェイルオーバー間で権限が取り消されると、収集された可変は適用されません。
- ランタイムおよびフェイルオーバー間で権限が付与されると、可変は収集されず、したがって何も適用されません。

保護レベルの統計

リクエスト境界および保護レベルの統計を使用してカバレッジのレベルを監視します。

アプリケーション・コンティニュイティはシステム、セッション、およびサービスの統計を収集し、これにより、ユーザーは保護レベルを監視できるようになります。統計情報は、V$SESSTAT、V$SYSSTATで入手でき、サービス統計が有効になっている場合は、V$SERVICE_STATSでも入手できます。たとえば、V$SESSTATを問い合せてV$STATNAMEと結合した場合は、次のような出力が表示されます。

```
NAME                                                             VALUE
---------------------------------------------------------------- ----------
cumulative begin requests                                           731
cumulative end requests                                             739
cumulative user calls in requests                                  7285
cumulative user calls protected by Application Continuity          7228
cumulative time in requests                                  2665167909
```

これらの統計は自動ワークロード・リポジトリ(AWR)に保存され、AWRレポートで使用できます。統計には、次の情報が含まれます。

- 完了リクエスト/秒
- リクエストでのユーザー・コール
- 保護されたユーザー・コール

AWRレポートの出力は、次のようになります。

```
Statistic                                Total     per Second    per Trans
---------------------------------------- --------- ------------- ---------
cumulative requests                        177,406          49.2       5.0
cumulative user calls in request           493,329         136.8      13.8
cumulative user calls protected            493,329         136.8      13.8
```

保護レベルの統計を有効にするには、(_request_boundaries = 3)を使用します。

セッション状態一貫性

セッション状態一貫性は、リクエスト中に非トランザクション状態がどのように変化するかを示します。

Oracleでは、session_state_consistencyをAUTO(透過的アプリケーション・コンティニュイティで使用可能)に設定することをお薦めします。これにより、セッション状態が追跡および管理されます。透過的アプリケーション・コンティニュイティを使用することを選択した場合、セッション状態の一貫性を確保するために何かを行う必要はありません。

手動アプリケーション・コンティニュイティの場合、session_state_consistencyをDYNAMICまたはSTATICに設定できます。アプリケーションを十分確認し、アプリケーションが値設定を変更する必要がない場合、session_state_consistencyをDYNAMICまたはSTATICに設定します。

セッション状態の例としては、NLS設定、オプティマイザのプリファレンス、イベントの設定、PL/SQLグローバル変数、一時表、アドバンスト・キュー、LOBおよび結果キャッシュがあります。コミットされたトランザクションで非トランザクションの値を変更する場合は、デフォルト値のDYNAMICを使用します(session_state_consistencyはサービス・レベルの属性であり、DYNAMICのデフォルト値です)。

COMMITの実行後にDYNAMICモードを使用すると、そのトランザクションで状態が変更された場合、セッションが失われたときに、その状態を再確立するためのトランザクションのリプレイができなくなります。初期設定後のセッション状態が静的と動的のどちらであるか、さらにCOMMIT操作後の処理続行が正しいかどうかに応じて、アプリケーションを分類できます。

DYNAMICモードはほとんどすべてのアプリケーションに適しています。不明な場合は、DYNAMICモードを使用してください。ユーザーがアプリケーションを変更できる場合は、DYNAMICモードを使用する必要があります。

ノート:

長時間実行するステートレスのアプリケーションの場合、session_state_consistency を AUTO または STATIC に設定します。ステートレスでないアプリケーションの場合、session_state_consistency を STATIC に設定しないでください。手動アプリケーション・コンティニュイティを必要としない場合、session_state_consistency を AUTO に設定することをお薦めします。

この項には次のトピックが含まれます:

- 自動的なセッション状態の一貫性
- 動的なセッション状態の一貫性
- 静的なセッション状態の一貫性

自動的なセッション状態の一貫性

session_state_consistencyをAUTOに設定した場合、リカバリ可能な停止の後にデータベース・セッションがリカバリできるように、透過的アプリケーション・コンティニュイティはセッションおよびトランザクションの状態を追跡および記録します。session_state_consistencyのAUTOへの設定は、透過的アプリケーション・コンティニュイティでのみ許容されます。

AUTOに設定すると、アプリケーションがユーザー・コールを発行するときに、状態追跡インフラストラクチャによりセッション状態の使用状況が分類されます。追跡されるセッション状態は監視および検証されます。

動的なセッション状態の一貫性

セッション状態の値がFAILOVER_RESTOREまたは初期化コールバックの追加で完全にリストアできない場合、セッションの状態は動的になります。

最初のトランザクションが完了した後、フェイルオーバーは次のリクエストが開始されるまでは内部的に無効化されます。Dynamicセッション状態一貫性モードでは、リクエスト中に状態の変更が行われ、次のリクエストの開始時点でリプレイが有効化されます。

トランザクションの実行時に非トランザクション・セッション状態が変更される場合、セッション状態一貫性モードをDynamicに設定します。ランタイム時に変更される可能性がある非トランザクション・セッション状態の例には、ALTER SESSION、PL/SQLグローバル変数、SYS_CONTEXTおよび一時表のコンテンツがあります。アプリケーションがトランザクション以外の状態をトランザクション内で変更し、それをコミットする場合、この状態はリプレイできないため、状態の設定をDynamicに設定する必要があります。アプリケーション・コンティニュイティに対してDynamicモードを使用する場合、次のリクエストが開始されるまではCOMMIT時にリプレイは無効です。デフォルト値はDynamicです。

セッション状態一貫性モードがDynamicのときには、リクエスト中に非トランザクションのセッション状態(NTSS)が変更されます。

リプレイ(つまり、アプリケーション・コンティニュイティ)はbeginRequestコールで有効化され、COMMIT時(endRequestコール)または制限付きのコール時に無効化されます。次に、3つのアプリケーション・シナリオのステップ・ロジックを示します。

- トランザクションなし
- 最後の文としてCOMMITが使用されたトランザクション
- COMMIT文が埋め込まれたトランザクション

トランザクションなしのリクエストの場合、論理的なステップは次のようになります。

1. チェックアウトします。
2. リクエストを開始し、リプレイを有効化します。
3. 1つ以上のSELECT文、および場合によっては他のPL/SQL文を発行します。
4. その他のアクションを実行します。
5. チェックインします。
6. リクエストを終了し、リプレイを無効化します。

最後の文としてCOMMITが使用されたトランザクションのリクエストの場合、論理的なステップは次のようになります。

1. チェックアウトします。
2. リクエストを開始し、リプレイを有効化します。
3. 1つ以上のSELECT文、および場合によっては他のPL/SQL文を発行します。
4. トランザクションが開始されます。
5. その他のアクションを実行します。
6. コミット(リプレイを無効化)します。
7. チェックインします。
8. リクエストを終了します。

COMMIT文が埋め込まれたトランザクションのリクエストの場合、論理的なステップは次のようになります。

1. チェックアウトします。
2. リクエストを開始し、リプレイを有効化します。
3. 1つ以上のSELECT文、および場合によっては他のPL/SQL文を発行します。
4. トランザクションが開始されます。

5. その他のアクションを実行します。
6. コミット(リプレイを無効化)します。
7. その他のアクション。この間、アプリケーションはアプリケーション・コンティニュイティでカバーされません。
8. チェックインします。
9. リクエストを終了します。

静的なセッション状態の一貫性

Staticモードは、長時間実行するステートレスのアプリケーションに使用します。ステートレスではないアプリケーションには、Staticモードを使用しないでください。

NLS設定、SYS_CONTEXT、PL/SQL変数、およびオプティマイザ・プリファレンスなどのすべての非トランザクション状態の変更がリクエストごとに1回の初期化の一環として設定され、このセッション状態がトランザクション中に変更されない場合にのみ、セッション状態一貫性モードをStaticに設定します。これらの設定は、FAILOVER_RESTORE=LEVEL1、コールバック、またはラベルなどを使用した接続の確立時、またはプールからの各チェックアウト時に、接続ごとに1回確立できます。

アプリケーション・コンティニュイティに対してStaticモードを使用する場合、トランザクションのフェイルオーバーはリクエストの最初のトランザクションの後も続行されます。これは、beginRequestを1回設定して、バッチ・ジョブや長いレポートなど、長時間の処理操作を実行するアプリケーションに役立ちます。

静的モードは、トランザクションで非トランザクション状態を変更する呼出しを使用するアプリケーションではサポートされません。このようなコールの具体的な例を次に示します。

- PL/SQLサブプログラム
- SYS_CONTEXT
- ALTER SESSION

静的モードは慎重に指定してください。静的モードは、アプリケーションがトランザクション内で非トランザクションのセッション状態を変更しない場合にのみ使用します。セッション状態一貫性モードをStaticとして宣言することは、リクエスト内の最初のCOMMITの後に続行しても安全であることを示します。動的モードはほとんどのアプリケーションに適しています。ユーザーがアプリケーションを変更またはカスタマイズできる場合は、静的モードを使用しないでください。

セッション状態一貫性モードがStaticのときに、リクエスト中の非トランザクションのセッション状態は一定に保たれます(つまり、変更されません)。

リプレイ(つまり、アプリケーション・コンティニュイティ)はbeginRequestコールで有効化され、制限付きコール、disableReplayまたはOCIRequestDisableReplayコール、またはendRequestコールで無効化されます。

次に、3つのアプリケーション・シナリオのステップ・ロジックを示します。

- トランザクションなし
- 最後の文としてCOMMITが最後に使用される1つ以上のトランザクション
- アプリケーション・コンティニュイティを無効化する制限付きコールを使用したトランザクションが続くCOMMIT文を使用したトランザクション

トランザクションなしのリクエストの場合、論理的なステップは次のようになります。

1. チェックアウトします。
2. リクエストを開始し、リプレイを有効化します。
3. 1つ以上のSELECT文、および場合によっては他のPL/SQL文を発行します。
4. その他のアクションを実行します。
5. チェックインします。
6. リクエストを終了し、リプレイを無効化します。

リプレイは、endRequest、制限付きコール、および明示的なdisableReplayまたはOCIRequestDisableReplayコールで無効化されます。

1つ以上のトランザクション(それぞれの最後の文としてCOMMITが使用される)のリクエストの場合、論理的なステップは次のようになります。

1. チェックアウトします。
2. リクエストを開始し、リプレイを有効化します。
3. 1つ以上のSELECT文、および場合によっては他のPL/SQL文を発行します。
4. トランザクションが開始されます。
5. トランザクションがコミットされます。
6. トランザクションがパージされます。

   (追加トランザクションごとに、ステップ4から6が実行されます。)
7. その他のアクションを実行します。
8. チェックインします。
9. リクエストを終了します。

リプレイは、endRequest、制限付きコール、および明示的なdisableReplayまたはOCIRequestDisableReplayコールで無効化されます。

制限付きコールを使用したトランザクションが続くCOMMIT文を使用したトランザクションのリクエストの場合、論理的なステップは次のようになります。

1. チェックアウトします。
2. リクエストを開始し、リプレイを有効化します。
3. 1つ以上のSELECT文、および場合によっては他のPL/SQL文を発行します。
4. トランザクションが開始されます。
5. トランザクションがコミットされます。
6. トランザクションがパージされます。
7. 2番目のトランザクションが開始されます。
8. トランザクションが制限付きのコールを発行し、これによってアプリケーション・コンティニュイティが無効化されます。
9. トランザクションがパージされます。

10. その他のアクションを実行します

11. チェックインします。

12. リクエストを終了します。

リプレイは、endRequest、制限付きコール、および明示的なdisableReplayまたはOCIRequestDisableReplayコールで無効化されます。

関連項目

- FAILOVER_RESTORE

アプリケーション・コンティニュイティの潜在的な副作用

FAILOVER_TYPEをTRANSACTIONに設定してアプリケーション・コンティニュイティを使用する場合、副作用を残す文がリプレイされます。

ノート:

アプリケーション所有者として、繰り返す必要のない副作用が含まれるリクエストのリプレイを無効にすることを選択できます。副作用を無効にする最も簡単な方法は、透過的アプリケーション・コンティニュイティ(FAILOVER_TYPE を AUTO に設定)を使用することです。これにより、副作用が無効になります。

アプリケーション・コンティニュイティは、データベース状態をリストアするためにPL/SQLを時間の経過順にリプレイします。これは、ユーザーによる発行が遅延されたものとしてセッションを再構築する上で役立ちます。ほとんどのアプリケーションは、レポートの作成や監査の完了など、発行が繰り返されたものとして完全な状態を再構築することを必要とします。ただし、状態を構築するためにリプレイされるアクションには、リプレイの影響に対応したりこの影響を軽減するためのアクションが必要なものもあります。一部のアプリケーションでは、繰り返す必要のないコールが含まれるリクエストのリプレイを無効化することを選択します。

副作用があるアクションの例は、次のとおりです。

- DBMS_ALERTコール(電子メールまたは他の通知)
- DBMS_FILE_TRANSFERコール(ファイルのコピー)
- DBMS_PIPEおよびRPCコール(外部ソース向け)
- UTL_FILEコール(テキスト・ファイルの作成)
- UTL_HTTPコール(HTTPコールアウトの実行)
- UTL_MAILコール(電子メールの送信)
- UTL_SMTPコール(SMTPメッセージの送信)
- UTL_TCPコール(TCPメッセージの送信)
- UTL_URLコール(URLへのアクセス)

外部アクション(自律型トランザクションやUTL_HTTPによるサービス指向アプリケーション(SOA)のコールの発行など)を伴うアプリケーションの場合、アプリケーションが外部アクションのリプレイ(電子メールの再送信、監査、およびファイルの転送など)によって満たされるときに、アプリケーション・コンティニュイティは透過的になります。

関連項目

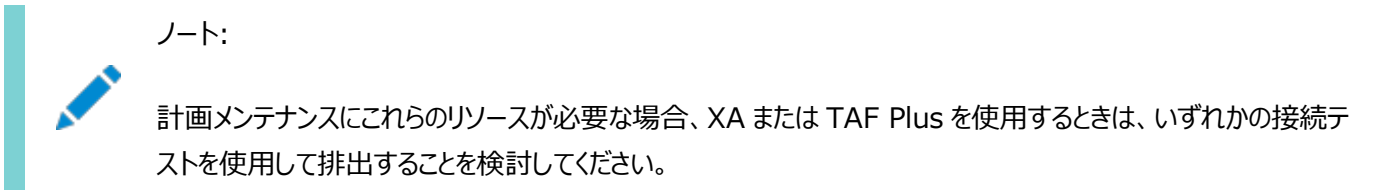
- アプリケーション・コンティニュイティにおけるリプレイの無効化

アプリケーション・コンティニュイティに関する制限および他の考慮事項

アプリケーション・コンティニュイティを使用するときには、次の制限事項と考慮事項に注意してください。

アプリケーション・コンティニュイティは次のものを除外します。

- JDBC OCIドライバ(タイプ2)
- ODP.NET管理対象ドライバ
- OLE DB
- ODBC
- OCCI
- Pro*プリコンパイラ(Proc*C、Pro*COBOL、Pro*FORTRANなど)



ノート:

計画メンテナンスにこれらのリソースが必要な場合、XA または TAF Plus を使用するときは、いずれかの接続テストを使用して排出することを検討してください。

Oracle Database 12cリリース2 (12.2.0.1)のOCIおよびODP.NETの場合、OCIドライバのアプリケーション・コンティニュイティは、ADT、拡張キューおよび一部のLOB APIを除外します。このような除外は、Javaには適用されません。

JDBCを使用するアプリケーションの場合、oracle.sqlの非推奨具象クラスOPAQUE、ANYDATAまたはSTRUCTはサポートされません。

アプリケーション・サーバー・レベルの文キャッシュが有効な場合(WebLogicやサードパーティのアプリケーション・サーバーの文キャッシュなど)、このキャッシュはリプレイの使用時に無効にする必要があります。かわりに、JDBC文キャッシュを構成します。このキャッシュはアプリケーション・コンティニュイティをサポートしていて、JDBCおよびOracle Databaseに向けて最適化されています(oracle.jdbc.implicitstatementcachesize=nnn)。

トランザクションのリプレイが発生する可能性があるときに、次の制限事項が関連する点に注意してください。

- Oracle Database 12リリース2 (12.2)以降、JavaおよびODP.NET管理対象外ドライバのXAデータ・ソースでリプレイがサポートされています。リプレイは、ローカル・トランザクションをサポートします。2フェーズを使用すると、リプレイはサイレントに無効化されます。これにより、アプリケーション・コンティニュイティは、昇格可能なXAおよびXAデータ・ソースを使用するアプリケーションと、XAを使用しないほとんどのアプリケーションをサポートできるようになります。

  リクエストで2フェーズ・コミットXAを使用する場合、Oracle Database 12cリリース2 (12.2)以降では、アプリケーション・コンティニュイティは昇格可能なXAでサポートされ、XAが使用されていないときにXAデータ・ソースを使用します。
- リクエストがALTER SYSTEM文またはALTER DATABASE文を発行すると、リプレイは無効になります。
- リプレイは、セッションを再構築するのに安全でないとみなされているALTER SESSION文のリクエスト・レベルでは無効化されています。これには、サポート・レベルのイベントを設定する文が含まれ、COMMIT IN PROCEDUREおよびGUARDを有効化および無効化します。

ただし、アプリケーション・レベルでのALTER SESSION文がリプレイでサポートされています。これらには、グローバリゼーション・サポート(NLS)設定の文、格納されているプライベートの概要、コンテナ(CDB/PDB)の設定、SQLトレースおよびPL/SQL警告が含まれます。

- リプレイのターゲット・データベースは、ソース・データベースと同じデータベース・クラスタ(Oracle RAC、Oracle Data Guard、Oracle Active Data Guard、またはOracle Multitenant)に存在している必要があります。ビジネス・トランザクションの整合性を保護するため、ターゲットが別のデータベースになる場合、アプリケーション・コンティニュイティはリプレイを実行しません。ターゲット・データベースがソース・データベース(またはプラガブル・データベース)と同じであっても、データベースがフラッシュ・バックされていたり、メディア・リカバリによって不完全にリカバリされていたり、Oracle Data Guardによって以前の時点でオープンされていたりするなどでデータが失われている場合、アプリケーション・コンティニュイティはリプレイを実行しません。
- ストリーム引数の場合、リプレイはベスト・エフォート・ベースです。たとえば、アプリケーションが物理アドレスを使用している場合、アドレスは停止によって失われると、再配置できなくなります。たとえば、JDBCストリーム・セッター(setBinaryStreamなど)によってリプレイが無効になります。
- プライマリ・データベースに戻る読取り/書込みデータベース・リンクを使用してOracle Active Data Guardを使用している場合、リプレイはサポートされません。これはトランザクション・ガードのセキュリティ制限です。
- パラレル問合せコールの失敗の場合、これが文レベルの失敗であるときにはリプレイは開始されません。たとえば、インスタンスやノードの障害またはメモリーの問題で発生したコール失敗に対するORA-12805:parallel query server died unexpectedlyの後には、リプレイは行われません。
- リプレイはJavaのDRCPをサポートしません。専用サーバーと共有サーバーはサポートされます。
- リプレイはISOLATION_LEVEL=SERIALIZABLEをサポートしていません。

ノート:

データベースのクローンを作成するために、ディスク・イメージ(BCV など)を分割する場合や、物理または Oracle Active Data Guard データベースではないロジカル・スタンバイまたはロジカル・コピーを作成するために別のデータベースになるようにデータベースをクローニングする場合は、データベースを区別するために nid ユーティリティを使用して DBID を変更する必要があります。

関連項目

- OCIのアプリケーション・コンティニュイティがフェイルオーバーできる場合
- NIDユーティリティを使用してDBID、DBNAMEを変更する方法(My Oracle SupportドキュメントID 863800.1)

クライアント・フェイルオーバーを向上させるためのトランザクション・ガード

トランザクション・ガードは、アプリケーション・コンティニュイティがリプレイするトランザクションが複数回適用されないようにします。

最後の送信がコミットされたこと、これからコミットされること、または実行が完了しなかったことを認識できないと、アプリケーションで問題となります。これは、再送信するユーザーや独自のリプレイを使用するアプリケーションが重複した要求を発行したり、データベースにすでにコミットされた変更内容が繰り返されたり、その他の形式の論理破損が発生する原因となる可能性があります。トランザクション・ガードを使用すると、この問題を解決できます。

アプリケーション・コンティニュイティは、自動的にトランザクション・ガードを有効化して使用しますが、トランザクション・ガードは個別に有効化することもできます。アプリケーションがアプリケーション・レベルのリプレイを実装している場合は、冪等性を実現するためにアプリケーションをトランザクション・ガードと統合する必要があります。

Oracle Database 12cでは、トランザクション・ガードによって、自動的かつ透過的に基準化された方法で冪等性を達成するために、アプリケーションで使用される新たな完全統合ツールが提供されます。トランザクション・ガードでは、論理トランザクションID (LTXID)を使用して、重複したトランザクションの送信を回避します。これは、トランザクションの冪等性と呼ばれます。LTXIDはコミット時に継続され、ロールバックの後に再使用されます。通常の実行中、LTXIDは、各データベース・トランザクションについてクライアントおよびサーバーの両方で自動的にセッションで保持されます。コミット時に、LTXIDはトランザクションのコミットの一環として継続され、次に使用するLTXIDがクライアントに返されます。

XAトランザクション用のトランザクション・ガード

トランザクション・ガードは、XAベースのトランザクションもサポートします。これは、Oracle WebLogic Server、Oracle Tuxedo、およびMicroSoft Transaction Server (Oracle ODP.NETを通じてOracle Databaseに公開されます)などのトランザクション・マネージャのオプションです。

XAトランザクションのトランザクション・ガード・サポートは、Oracle WebLogic ServerでのXAトランザクションのリカバリ可能な停止の後の安全なリプレイを実現します。XAサポートの追加により、Oracle WebLogic Serverは、トランザクション・ガードを使用した冪等性を持つリプレイを実現できます。

この項には次のトピックが含まれます:

- トランザクション・ガードの構成チェックリスト
- トランザクション・ガードのサービスの構成
- 透過的アプリケーション・フェイルオーバーによるOCIクライアントのフェイルオーバー

関連項目

- Oracle Database開発ガイド
- Oracle Database開発ガイド
- 『Oracle Database JDBC開発者ガイド』

トランザクション・ガードの構成チェックリスト

トランザクション・ガードのサービスを構成する前に、次の構成チェックリストを使用します。

- 次のように、GET_LTXID_OUTCOMEを呼び出すアプリケーション・ユーザーに権限を付与します。

```
GRANT EXECUTE ON DBMS_APP_CONT to user_name;
```

ノート:

アプリケーション・コンティニュイティを使用する場合は、この文を実行しないでください。

- 最適なパフォーマンスを実現するためにトランザクション履歴表を特定および定義します。

  トランザクション履歴表(LTXID_HIST)は、Oracle Databaseの作成時またはアップグレード時にデフォルトで

SYSAUX表領域に作成されます。最後のパーティションの記憶域を使用して、インスタンスの追加時に新しいパーティションが追加されます。トランザクション履歴表の場所がパフォーマンス上最適でない場合は、別の表領域に移動して、そこでパーティションを作成できます。たとえば、次の文により、FastPaceという名前の表領域にトランザクション履歴表を移動します。

```
ALTER TABLE LTXID_TRANS move partition LTXID_TRANS_1 tablespace FastPace
   storage ( initial 10G next 10G minextents 1 maxextents 121 );
```

- -commit_outcomeおよび-retentionサービス・パラメータの値を設定します。
- Oracle RAC、Oracle Data GuardまたはOracle Active Data Guardを使用している場合は、停止を迅速に通知するためにFANを使用することをお薦めします。

トランザクション・ガードのサービスの構成

トランザクション・ガードを使用するようにサービスを構成するには、次のサービス・パラメータを設定します。

- -commit_outcome: -commit_outcomeサービス・パラメータをTRUEに設定します。このサービス・パラメータにより、COMMITが実行されて停止が発生した後に、トランザクションのコミット結果 にアクセスできるかどうかが決定されます。Oracle DatabaseではCOMMITは常に永続的ですが、トランザクション・ガードでは、COMMITの結果が永続的になり、アプリケーションでは、それを使用して、停止の前に実行された最後のトランザクションのステータスを強制適用します。
- -retention: -retentionサービス・パラメータを-commit_outcomeとともに使用します。このサービス・パラメータにより、COMMIT結果が保持される時間(秒数)が決定されます。ほとんどのインストールではデフォルト値を使用することをお薦めします。

次のSRVCTLコマンドにより、salesという名前のポリシー管理サービスをトランザクション・ガードに構成します。

```
$ srvctl add service -db crm -service sales -serverpool spool_1
  -commit_outcome TRUE -retention 86400 -notification TRUE
```

次のSRVCTLコマンドにより、salesという名前の管理者管理サービスをトランザクション・ガードに構成します。

```
$ srvctl add service -db crm -service sales -preferred crm_1,crm_2
  -available crm_3,crm_4 -commit_outcome TRUE -retention 86400
  -notification TRUE
```

srvctl modify serviceコマンドを使用して、既存のサービスをトランザクション・ガード用に構成するように変更することもできます。

ノート:

デフォルト・データベース・サービス(db_name または db_unique_name の値に名前が設定されているサービス)は使用しないでください。デフォルト・サービスは、管理目的に使用されるものであり、ユーザー作成のサービスと同じプロパティを備えていません。

関連項目

- srvctl add service

- srvctl modify service
- アプリケーション・コンティニュイティについて
- 『Oracle Database JDBC開発者ガイド』
- Oracle Call Interfaceプログラマーズ・ガイド

透過的アプリケーション・フェイルオーバーによるOCIクライアントのフェイルオーバー

Oracle Net Servicesによってインスタンスへの接続が確立されると、Oracle Call Interface (OCI)クライアントが接続をクローズするか、インスタンスが停止するか、または障害が発生するまで、接続はオープン状態のまま維持されます。

接続に透過的アプリケーション・フェイルオーバー(TAF)を構成すると、インスタンスで障害が発生した場合、Oracle Databaseは残りのインスタンスでセッションをリプレイします。

TAFでは、フェイルオーバーが完了すると問合せは再開できますが、INSERT、UPDATE、DELETEなどの他のトランザクションの場合、アプリケーションで、失敗したトランザクションをロールバックして再度送信する必要があります。FAILOVER_RESTOREをLEVEL1またはAUTOに設定しなかった場合は、フェイルオーバーの発生後、セッションのカスタマイズ、つまりALTER SESSION文も再実行する必要があります。ただし、TAFでは、ワークロードが変化しても、通常処理の間は接続が移動されません。

サービスはTAFのデプロイメントを簡素化します。サービスのTAFポリシーを定義でき、このサービスを使用するすべての接続によって、自動的にTAFが有効になります。これには、クライアント側の変更は必要ありません。サービスのTAF設定は、クライアントの接続定義内のTAF設定よりも優先されます。

-failovermethodおよび-failovertypeパラメータを定義して、サービスのすべてのユーザー用のTAFポリシーを定義できます。-failoverretryおよび-failoverdelayパラメータをそれぞれ使用して、失敗したセッションによるサービスへの再接続試行回数、および再接続の試行間での待機時間を設定して、TAFポリシーをさらに詳しく定義できます。

サービスのTAFポリシーを定義するには、次の例に示すようにSRVCTLを使用します(サービス名はtafconn.example.com、データベース名はcrmです)。

```
$ srvctl modify service -db crm -service tafconn.example.com -failovermethod BASIC
  -failovertype SELECT -failoverretry 10 -failoverdelay 30
```

TAFが有効なOCIアプリケーションでは、高速接続フェイルオーバーのためにFAN高可用性イベントを使用します。

トランザクション・ガードとFAILOVER_RESTOREをサポートするTAF

トランザクション・ガードを使用すると、開発者向けのエラーがTAFによって管理されます。TAFとトランザクション・ガードの両方を使用すると、開発者はTAFエラーを使用して、コミットされていないトランザクションをロール・バックして安全に再送信することも、そのトランザクションを返すこともできます(TAFエラー・コードORA-25402、ORA-25408、ORA-25405の場合)。

FAILOVER_RESTOREを使用していると、TAFは自動的に一般的な状態をリストアします。これにより、ほとんどのアプリケーションでコールバックの必要がなくなります。

関連項目

- FAILOVER_RESTORE

- トランザクション・ガードの理解

7 Recovery Managerの構成およびアーカイブ

この章では、Oracle Real Application Clusters (Oracle RAC)環境で使用するためのRecovery Manager (RMAN)の構成方法について説明します。Oracle RAC環境でアーカイブを実行するために使用する方法、オンラインREDOログおよびアーカイブREDOログに関する考慮事項についても説明します。

内容は次のとおりです。

Oracle RACのRMANの構成の概要

RMANを使用すると、データ・ファイル、制御ファイル、サーバー・パラメータ・ファイル(SPFILE)およびアーカイブREDOログ・ファイルのバックアップ、リストアおよびリカバリを実行できます。RMANはOracle Databaseに含まれているため、個別にインストールする必要はありません。RMANは、コマンドラインから実行するか、Oracle Enterprise ManagerのBackup Managerで使用できます。

Oracle RACでのアーカイブ・モード

REDOログ・ファイルがアーカイブされるためには、Oracle RACデータベースがARCHIVELOGモードになっている必要があります。

データベースがローカル・インスタンスによってマウントされていても、いずれのインスタンスでもオープンされていないため、SQL文ALTER DATABASEを実行すると、Oracle RACでアーカイブ・モードを変更できます。この文を実行するために、パラメータの設定を変更する必要はありません。

ノート:

- ARCHIVELOG モードは、インスタンス・レベルではなく、データベース・レベルで設定します。すべてのインスタンスがアーカイブされるか、1 つもされないかのどちらかです。
- Oracle Enterprise Manager の Oracle RAC データベースの「ホーム」ページにある「メンテナンス」タ

ブの「リカバリ設定」ページを使用して、アーカイブ・ログ・モードを変更することもできます。

関連項目

- Oracle Database管理者ガイド

RMANのスナップショット制御ファイルの位置の構成

スナップショット制御ファイルは、RMANによってオペレーティング・システム固有の位置に作成されるデータベース制御ファイルのコピーです。

RMANでは、制御ファイルの一貫性バージョンを保持して、リカバリ・カタログの再同期化時または制御ファイルのバックアップ時に使用できるように、スナップショット制御ファイルを作成します。

クラスタ・ファイル・システムまたはRAWデバイス接続先をスナップショット制御ファイルの位置に指定できます。このファイルはクラスタ内のすべてのノードで共有され、クラスタ内のすべてのノードがアクセスできる必要があります。次のRMANのコマンドを実行して、スナップショット制御ファイルの構成位置を判別します。

```
SHOW SNAPSHOT CONTROLFILE NAME;
```

スナップショット制御ファイルの構成位置は変更できます。たとえば、LinuxシステムおよびUNIXシステムの場合は、RMANのプロンプトで次のコマンドを入力し、スナップショット制御ファイルの位置を$ORACLE_HOME/dbs/scf/snap_prod.cfに指定できます。

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '$ORACLE_HOME/dbs/scf/snap_prod.cf';
```

このコマンドは、クラスタ・データベースのすべてのインスタンスのスナップショット制御ファイルの位置構成をグローバルに設定します。したがって、バックアップを実行するすべてのノードによって$ORACLE_HOME/dbs/scfディレクトリが共有されるようにしてください。

CONFIGUREコマンドを使用すると、複数のRMANセッションにまたがる永続的な設定を作成できます。したがって、スナップショット制御ファイルの位置を変更しないかぎり、再度このコマンドを実行する必要はありません。

スナップショット制御ファイルを削除するには、次に示すように、最初にスナップショット制御ファイルの位置を変更してから、古い位置のファイルを削除する必要があります。

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO 'new_name';
DELETE COPY OF CONTROLFILE;
```

関連項目

- 『Oracle Database Recovery Managerリファレンス』

制御ファイルおよびSPFILEを自動的にバックアップするようなRMANの構成

CONFIGURE CONTROLFILE AUTOBACKUPをONに設定すると、BACKUPコマンドまたはCOPYコマンドの実行後に、制御ファイルとSPFILEのバックアップがRMANによって自動的に作成されます。

リカバリを実行する際のインスタンスの起動にSPFILEが必要な場合(SPFILEのデフォルト位置がOracle RACデータベースの

すべてのノードで使用可能である必要があるため)、RMANはSPFILEを自動的にリストアすることもできます。

ノート:

SQL*Plus ALTER DATABASE コマンドを使用して制御ファイルをバックアップする場合は、すべてのノードによって共有されるデバイスで制御ファイルのバックアップを作成することも必要となります。

これらの機能は、リカバリ・カタログがない場合でもRMANが制御ファイルをリストアできるため、障害時リカバリでは重要です。リカバリ・カタログと現行の制御ファイルの両方が失われた場合でも、RMANによって、自動バックアップされた制御ファイルをリストアできます。CONFIGURE CONTROLFILE AUTOBACKUP FORMATコマンドを使用すると、RMANによって指定されたこのファイルのデフォルト名を変更できます。このコマンドで絶対パス名を指定する場合、そのパスはバックアップに使用するすべてのノードにも存在している必要があります。

RMANは、最初に割り当てられたチャネルで制御ファイルの自動バックアップを実行します。したがって、異なるパラメータを使用して複数のチャネルを割り当てる場合、特にCONNECTコマンドを使用してチャネルを割り当てる場合は、どのチャネルで制御ファイルの自動バックアップを実行するかを決定する必要があります。そのノードに、常に最初にチャネルを割り当ててください。

RMANの制御ファイルを使用する他に、Oracle Enterprise Managerを使用してRMAN機能を使用することもできます。

関連項目

- 『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

複数のOracle RACノードでのクロスチェック

複数のノードでクロスチェックを実行する場合(およびRMAN全般を操作する場合)、どのノードでバックアップを作成したかに関係なく、すべてのノードからすべてのバックアップにアクセスできるようにクラスタを構成します。

クラスタをこのように構成すると、リストアまたはクロスチェックの操作中に、クラスタ内のすべてのノードにチャネルを割り当てられます。

各ノードからすべてのバックアップにアクセスできるようにクラスタを構成できない場合は、リストアおよびクロスチェックの操作中に、CONFIGURE CHANNELコマンドのCONNECTオプションを使用して複数のノードでチャネルを割り当てて、1つ以上のノードからすべてのバックアップにアクセスできるようにする必要があります。バックアップにアクセス可能なノードでチャネルを構成していなかったために、クロスチェック中にバックアップにアクセスできなかった場合、クロスチェック後にRMANリポジトリ内でそのバックアップにEXPIREDのマークが付けられます。

たとえば、クラスタ内の様々なノードでテープ・バックアップが作成され、各バックアップがバックアップを作成したノードでのみ使用可能なOracle RAC構成で、CONFIGURE CHANNEL ... CONNECTを使用できます。

関連項目

- 特定のノードを使用するようなチャネルの構成
- 『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

Oracle RACでのRMANのチャネルの構成

この項では、RMANでチャネルを構成する方法を説明します。次の各項目で説明するように、チャネルは、自動ロード・バランシングを使用するか、特定のインスタンスに特定のチャネルを指定して構成できます。

- 自動ロード・バランシングを使用するようなチャネルの構成
- 特定のノードを使用するようなチャネルの構成

自動ロード・バランシングを使用するようなチャネルの構成

自動ロード・バランシングを使用するようにチャネルを構成するには、次の構文を使用します。

```
CONFIGURE DEVICE TYPE [disk | sbt] PARALLELISM number_of_channels;
...
```

number_of_channelsは、この操作に使用するチャネル数です。このワンタイム構成を完了した後、BACKUPコマンドまたはRESTOREコマンドを発行できます。

特定のノードを使用するようなチャネルの構成

ポリシー管理Oracle RACデータベース・インスタンスごとに1つのRMANチャネルを構成するには、次の構文を使用します。

```
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@racinst_1'
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@racinst_2'
...
```

このワンタイム構成ステップを実行した後、BACKUPコマンドまたはRESTOREコマンドを発行できます。

Oracle RACでのRMANを使用したアーカイブREDOログの管理

ノードでアーカイブREDOログが生成されると、Oracle Databaseはそのログのファイル名を常にターゲット・データベースの制御ファイルに記録します。リカバリ・カタログを使用している場合、アーカイブREDOログのファイル名は、RMANによって再同期化の実行時にリカバリ・カタログにも記録されます。

あるノードが特定のファイル名でログをファイル・システムに書き込む場合、このアーカイブREDOログにアクセスするすべてのノードに対してそのファイルが読取り可能になっている必要があるため、使用するアーカイブREDOログのネーミング・スキームが重要となります。たとえば、node1が/oracle/arc_dest/log_1_100_23452345.arcにログをアーカイブしている場合、node2は、自身のファイル・システムで/oracle/arc_dest/log_1_100_23452345.arcを読み取ることができる場合にのみ、このアーカイブREDOログをバックアップできます。

選択するバックアップとリカバリの計画は、各ノードのアーカイブ先を構成する方法によって異なります。アーカイブREDOログのバックアップを1つのノードのみで実行するか、全ノードで実行するかに関係なく、すべてのアーカイブREDOログがバックアップされることを確認する必要があります。リカバリ中にRMANのパラレル化を使用する場合、リカバリを実行するノードにはクラスタ内のすべてのアーカイブREDOログに対する読取りアクセス権が必要です。

複数のノードが、アーカイブ・ログをパラレルでリストアできます。ただし、リカバリ中にアーカイブ・ログを適用できるノードは1つのみです。したがって、リカバリを実行中のノードは、リカバリ操作に必要なすべてのアーカイブ・ログにアクセスできる必要があります。

デフォルトでは、データベースはパラレル・スレッドの最適数を判断し、リカバリ操作に使用します。RECOVERコマンドのPARALLEL句を使用して、パラレル・スレッドの数を変更できます。

アーカイブREDOログに関するガイドラインおよび考慮事項

アーカイブREDOログに関しては、リカバリ時に(可能な場合はバックアップ時にも)、各ノードからすべてのアーカイブREDOログが読み取れるようにすることが重要です。リカバリ中、アーカイブ・ログの宛先がリカバリを実行するノードから認識可能であれば、Oracle Databaseはアーカイブ・ログのデータを正常にリカバリできます。

Oracle RACのアーカイブREDOログ・ファイルの表記規則

アーカイブREDOログの構成では、LOG_ARCHIVE_FORMATパラメータでアーカイブREDOログを一意に識別します。

このパラメータのフォーマットは、オペレーティング・システム固有で、テキスト文字列、1つ以上の変数およびファイル名拡張子を指定できます。

表7-1 アーカイブREDOログ・ファイル名のフォーマット・パラメータ

パラメータ	説明	例
%r	埋込みなしのリセットログ識別子	log_1_62_23452345
%R	左端に 0 が埋め込まれたリセット・ログ識別子	log_1_62_0023452345
%s	埋込みなしのログ順序番号	log_251
%S	左端に 0 が埋め込まれたログ順序番号	log_0000000251
%t	埋込みなしのスレッド番号	log_1
%T	左端に 0 が埋め込まれたスレッド番号	log_0001

アーカイブREDOログのすべてのファイル名形式パラメータは、大/小文字のいずれも、Oracle RACに必須です。これらのパラメータによって、Oracle Databaseは、すべてのインカネーションのアーカイブ・ログに対して一意の名前を作成できます。この要件は、COMPATIBLEパラメータが10.0以上に設定されている場合に適用されます。

%Rまたは%rパラメータを使用してリセットログ識別子を含め、以前のインカネーションでログが上書きされないようにします。ログのフォーマットを指定しない場合、デフォルトでオペレーティング・システム固有のフォーマットが使用され、%t、%sおよび%rが含まれます。

たとえば、REDOスレッド番号1に対応付けられたインスタンスによって、LOG_ARCHIVE_FORMATがlog_%t_%s_%r.arcに設定されると、そのアーカイブREDOログ・ファイル名は次のようになります。

```
log_1_1000_23435343.arc
log_1_1001_23452345.arc
log_1_1002_23452345.arc
```

. . .

関連項目

- Oracle Database管理者ガイド

RMANのアーカイブ構成使用例

この項では、Oracle RACデータベースでのアーカイブの使用例について説明します。この章の2つの構成使用例では、Oracle RACデータベース用の3ノードのUNIXクラスタについて説明します。いずれの使用例でも、リカバリを実行するインスタンスに指定するLOG_ARCHIVE_FORMATは、REDOログ・ファイルをアーカイブするインスタンスに指定したフォーマットと同じである必要があります。

この項には次のトピックが含まれます:

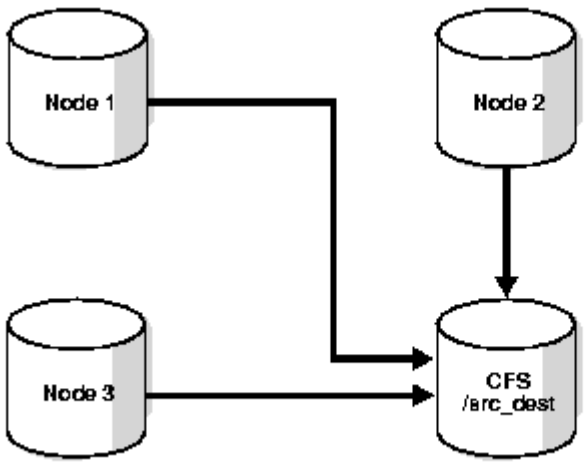
- Oracle Automatic Storage Managementおよびクラスタ・ファイル・システムのアーカイブ・スキーム
- 非クラスタ・ファイル・システムのローカル・アーカイブ・スキーム

Oracle Automatic Storage Managementおよびクラスタ・ファイル・システムのアーカイブ・スキーム

Oracle RACで推奨する構成は、データファイル用とは異なるリカバリ・セット用のディスク・グループを使用して、Oracle Automatic Storage Management(Oracle ASM)をリカバリ領域に使用する構成です。

Oracle ASMを使用する場合は、Oracle Managed Filesの名前のフォーマットが使用されます。この構成のかわりに、クラスタ・ファイル・システムのアーカイブ・スキームを使用することもできます。クラスタ・ファイル・システムを使用する場合、各ノードは、REDOログ・ファイルをアーカイブする際にクラスタ・ファイル・システムの1つの場所に書き込みます。各ノードは、他のノードのアーカイブREDOログ・ファイルを読み取ることができます。たとえば、図7-1に示すように、ノード1がREDOログ・ファイルをクラスタ・ファイル・システムの/arc_dest/log_1_100_23452345.arcにアーカイブすると、クラスタの他のノードもこのファイルを読み取ることできます。

図7-1 クラスタ・ファイル・システムのアーカイブ・スキーム



ノート:

この例のアーカイブ・ログ名のフォーマットは、クラスタ・ファイル・システムの例のみに適用されます。

クラスタ・ファイル・システムを使用しない場合は、アーカイブREDOログ・ファイルをRAWデバイス上に置くことができません。これは、RAWデバイスでは、連続したアーカイブ・ログ・ファイルの順次書込みができないためです。

関連項目

- 『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

クラスタ・ファイル・システムのアーカイブ・スキームのメリット

このスキームには、いずれのノードもログのアーカイブでネットワークを使用しないというメリットがあります。あるノードが書き込んだファイル名はクラスタ内の任意のノードで読み取ることができるため、RMANは、クラスタ内の任意のノードからすべてのログをバックアップできます。各ノードには、すべてのアーカイブREDOログに対するアクセス権があるため、バックアップとリストアのスクリプトが簡素化されます。

クラスタ・ファイル・システムのアーカイブ・スキームに関する初期化パラメータの設定

クラスタ・ファイル・システムのスキームでの各ノードは、クラスタ・データベース内のすべてのインスタンスで同じ名前を使用して識別されるディレクトリにアーカイブします(次の例では/arc_dest)。このディレクトリを構成するには、次の例のようにLOG_ARCH_DEST_1パラメータに値を設定します。

```
*.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
```

次のリストは、前述の例に基づいて、RMANのカタログまたは制御ファイルに表示されるアーカイブREDOログ・エントリの例を示しています。すべてのノードが任意のスレッドを使用してログをアーカイブできることに注意してください。

```
/arc_dest/log_1_999_23452345.arc
/arc_dest/log_1_1000_23435343.arc
/arc_dest/log_1_1001_23452345.arc <- thread 1 archived in node3
/arc_dest/log_3_1563_23452345.arc <- thread 3 archived in node2
/arc_dest/log_2_753_23452345.arc <- thread 2 archived in node1
/arc_dest/log_2_754_23452345.arc
/arc_dest/log_3_1564_23452345.arc
```

クラスタ・ファイル・システムのアーカイブ・スキームでのアーカイブ・ログの位置

どのノードがログを作成したかに関係なく、どのノードでもアーカイブ・ログを読み取ることができます。

ファイル・システムは共有化されており、各ノードはそのノード自体のアーカイブREDOログをクラスタ・ファイル・システムの/arc_destディレクトリに書き込むため、各ノードはそのノード自体、および他のノードが書き込んだログを読み取ることができます。

非クラスタ・ファイル・システムのローカル・アーカイブ・スキーム

非クラスタ・ファイル・システムにローカルにアーカイブする場合、各ノードは、一意の名前のローカル・ディレクトリにアーカイブします。リカバリが必要な場合は、他のノードのディレクトリにリモートでアクセスできるように、リカバリ・ノードを構成できます。たとえば、LinuxコンピュータおよびUNIXコンピュータのNFSまたはWindowsシステムのマップされたドライブを使用します。したがって、各

ノードはローカルの宛先にのみ書き込みますが、他のノード上のリモート・ディレクトリにあるアーカイブREDOログ・ファイルを読み取ることもできます。

非クラスタ・ファイル・システムのローカル・アーカイブの使用に関する考慮事項

メディア・リカバリに非クラスタ・ファイル・システムのローカル・アーカイブを使用する場合、ノードで他のノード上のアーカイブ・ディレクトリにあるアーカイブREDOログ・ファイルを読み取れるようにするために、他のノードへのリモート・アクセスのリカバリを実行するノードを構成する必要があります。

また、リカバリを実行する際に、使用可能なすべてのアーカイブ・ログがない場合は、アーカイブREDOログの順序番号が欠落している最初の地点まで不完全リカバリを実行する必要があります。このスキームのために特定の構成を使用する必要はありません。ただし、バックアップ処理を複数のノードに分散する最も簡単な方法は、「バックアップおよびリカバリの管理」のバックアップの例に示すとおり、チャネルを構成することです。

ノート:

非クラスタの場合は異なるファイル・システムが使用されるため、アーカイブ・ログ・ディレクトリは各ノードで一意である必要があります。たとえば、/arc_dest_1 は node1 でのみ使用可能で、/arc_dest_2 は node2 にのみ直接マウントされます。

その後、node1 は、NFS を介して node2 から/arc_dest_2 をマウントし、node3 から/arc_dest_3 をマウントします。

関連項目

- バックアップおよびリカバリの管理

非クラスタ・ファイル・システムのローカル・アーカイブに関する初期化パラメータの設定

ポリシー管理データベースまたは管理者管理のデータベースの初期化パラメータ・ファイルで、アーカイブ先の値を次のように、設定できます。

次の例に示すように、SID指定子を使用して、各インスタンスにSID.LOG_ARCH_DESTパラメータを設定します。

```
sid1.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_1"
sid2.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_2"
sid3.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_3"
```

ポリシー管理データベースでは、ノードおよびインスタンス・バインディングを手動で作成し、次のように、sid1が常に同じノードで動作するようにします。

```
$ srvctl modify database -d mydb -n node1 -i sid1
$ srvctl modify database -d mydb -n node2 -i sid2
$ srvctl modify database -d mydb -n node3 -i sid3
```

次のリストは、データベース制御ファイルのアーカイブREDOログ・エントリを示しています。障害発生後にデータベースをリカバリするために、すべてのノードは任意のスレッドからアーカイブREDOログの読取りができる必要があることに注意してください。

```
/arc_dest_1/log_1_1000_23435343.arc
/arc_dest_2/log_1_1001_23452345.arc <- thread 1 archived in node2
/arc_dest_2/log_3_1563_23452345.arc <- thread 3 archived in node2
/arc_dest_1/log_2_753_23452345.arc <- thread 2 archived in node1
/arc_dest_2/log_2_754_23452345.arc
/arc_dest_3/log_3_1564_23452345.arc
```

非クラスタ・ファイル・システムのローカル・アーカイブでのアーカイブ・ログの位置

表7-2に示すように、3つのノードそれぞれにはローカルのアーカイブREDOログを含むディレクトリがあります。また、NFSまたはマップされたドライブを介して他のノード上のディレクトリをリモートでマウントしている場合、各ノードには、残りのノードがアーカイブしたアーカイブREDOログ・ファイルをRMANで読み取ることができる、2つのリモート・ディレクトリがあります。

ノート:

表 7-2 に記載されているようなアーカイブ・ログ先は、NFS ディレクトリを別のノードにマウントする場合に、既存のアーカイブ・ログ・ディレクトリと競合しないように、各ノードで異なっている必要があります。

表7-2 UNIX/NFSのログの位置の例: 非クラスタ・ファイル・システムのローカル・アーカイブ

ノード	アーカイブREDOログ・ファイルを読み取るディレクトリ	ノードがアーカイブするログ
1	/arc_dest_1	1
1	/arc_dest_2	2 (NFS を介して)
1	/arc_dest_3	3 (NFS を介して)
2	/arc_dest_1	1 (NFS を介して)
2	/arc_dest_2	2
2	/arc_dest_3	3 (NFS を介して)
3	/arc_dest_1	1 (NFS を介して)
3	/arc_dest_2	2 (NFS を介して)
3	/arc_dest_3	3

非クラスタ・ファイル・システムのローカル・アーカイブに関するファイル・システムの構成

リカバリを実行していて、障害が発生しなかったインスタンスが、まだバックアップされていないディスク上のログをすべて読み取る必

要がある場合は、表7-3に示すようにNFSを構成する必要があります。

表7-3 共有読取りローカル・アーカイブに関するUNIX/NFSの構成

ノード	ディレクトリ	構成	マウント先	ノード
1	/arc_dest_1	ローカルの読取り/書込み	該当なし	該当なし
1	/arc_dest_2	NFS 読取り	/arc_dest_2	2
1	/arc_dest_3	NFS 読取り	/arc_dest_3	3
2	/arc_dest_1	NFS 読取り	/arc_dest_1	1
2	/arc_dest_2	ローカルの読取り/書込み	該当なし	該当なし
2	/arc_dest_3	NFS 読取り	/arc_dest_3	3
3	/arc_dest_1	NFS 読取り	/arc_dest_1	1
3	/arc_dest_2	NFS 読取り	/arc_dest_2	2
3	/arc_dest_3	ローカルの読取り/書込み	該当なし	該当なし

ノート:

Windows ユーザーは、マップされたドライブを使用してこの例と同じ結果を得ることができます。

アーカイバ・プロセスの監視

RMAN構成がOracle RAC環境で操作可能になった後、GV$ARCHIVE_PROCESSESビューとV$ARCHIVE_PROCESSESビューを使用してアーカイバ・プロセスのステータスを判断します。これらのビューには、問合せ対象がグローバル・ビューかローカル・ビューのいずれであるかに従って、すべてのデータベース・インスタンスに関する情報または接続先のインスタンスのみに関する情報がそれぞれ表示されます。

ノート:

kill コマンドを使用してアーカイバ・プロセスを停止した場合、そのデータベース・インスタンスは失敗します。

関連項目

- Oracle Database管理者ガイド

- Oracle Databaseリファレンス

8 バックアップおよびリカバリの管理

この章では、インスタンス・リカバリおよびRecovery Manager (RMAN)を使用したOracle Real Application Clusters (Oracle RAC)データベースのバックアップおよびリストアの方法について説明します。また、Oracle RACインスタンス・リカバリ、パラレル・バックアップ、SQL*Plusを使用したリカバリ、およびOracle RACでの高速リカバリ領域の使用についても説明します。内容は次のとおりです。

- 非クラスタ・ファイル・システムでのRMANのバックアップ機能の使用例
- Oracle RACでのRMANのリストア機能の使用例
- Oracle RACでのインスタンス・リカバリ
- Oracle RACでのメディア・リカバリ
- Oracle RACでのパラレル・リカバリ
- Oracle RACでの高速リカバリ領域の使用

ノート:

Oracle RAC 環境でのリストアおよびリカバリでは、リカバリを実行するインスタンスを、すべてのデータ・ファイルをリストアする唯一のインスタンスとしても構成する必要はありません。Oracle RAC では、クラスタのすべてのノードからデータ・ファイルにアクセスできるため、すべてのノードでアーカイブ REDO ログ・ファイルをリストアできます。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

関連項目:

Oracle Cluster Registry(OCR)などのOracle Clusterwareコンポーネントおよび投票ディスクのバックアップおよびリストアについては、『*Oracle Clusterware*管理およびデプロイメント・ガイド』を参照してください。

非クラスタ・ファイル・システムでのRMANのバックアップ機能の使用例

非クラスタ・ファイル・システム環境では、各ノードはローカルにマウントされる非クラスタ・ファイル・システムのディレクトリにのみバックアップできます。たとえば、リモート・アクセスに対応するようにネットワーク・ファイル・システムを構成しないかぎり、node1は、node2またはnode3のアーカイブREDOログ・ファイルへはアクセスできません。バックアップに対応するようにネットワーク・ファイル・システム・ファイルを構成した場合、各ノードはアーカイブREDOログをローカル・ディレクトリにバックアップします。

Oracle RACでのRMANのリストア機能の使用例

この項では、次の一般的なRMANのリストア機能の使用例について説明します。

- クラスタ・ファイル・システムからのバックアップのリストア

- 非クラスタ・ファイル・システムからのバックアップのリストア
- RMANまたはOracle Enterprise Managerを使用したサーバー・パラメータ・ファイル(SPFILE)のリストア

ノート:

クラスタ・ファイル・システムのスキームでのリストアおよびリカバリの手順は、非クラスタの Oracle の場合とほぼ同じです。

クラスタ・ファイル・システムからのバックアップのリストア

この項で説明するスキームでは、「Oracle Automatic Storage Managementおよびクラスタ・ファイル・システムのアーカイブ・スキーム」が使用されていると仮定します。このスキームでは、node3がクラスタ・ファイル・システムへのバックアップを実行したとします。リストアおよびリカバリ操作にnode3が使用可能で、すべてのアーカイブ・ログがバックアップ済か、ディスク上にある場合は、次のコマンドを実行して完全リカバリを実行します。

```
RESTORE DATABASE;
RECOVER DATABASE;
```

バックアップを実行したnode3 が使用できない場合は、残りのノードの1つに対してメディア管理デバイスを構成し、このノードでnode3のバックアップ・メディアを使用可能にします。

ノート:

「自動ロード・バランシングを使用するようなチャネルの構成」で説明したとおり、RMAN が構成されている場合、ノード間でチャネルをロード・バランスを実行するには、少なくとも 1 つのインスタンスがデータベースを正常にオープンするまでロード・バランスができないことに注意してください。つまり、データベース全体のリストア中はノード間でチャネルにロード・バランスが実行されることはありません。RESTORE および RECOVER コマンド用にチャネルのロード・バランシングをアーカイブするには、次のようなコマンドを実行して、チャネルを一時的に再割当てできます。

```
run {
ALLOCATE CHANNEL DEVICE TYPE sbt C1 CONNECT '@racinst_1'
ALLOCATE CHANNEL DEVICE TYPE sbt C2 CONNECT '@racinst_2'
...
}
```

関連項目

- Oracle Automatic Storage Managementおよびクラスタ・ファイル・システムのアーカイブ・スキーム
- 自動ロード・バランシングを使用するようなチャネルの構成

非クラスタ・ファイル・システムからのバックアップのリストア

この項で説明するスキームでは、「非クラスタ・ファイル・システムのローカル・アーカイブ・スキーム」が使用されていると仮定します。このスキームでは、各ノードが異なるディレクトリにローカルでアーカイブします。たとえば、node1は/arc_dest_1に、node2は/arc_dest_2に、node3は/arc_dest_3にアーカイブします。リカバリ・ノードが残りのノードでアーカイブ・ディレクトリを読み取るこ

とができるように、ネットワーク・ファイル・システム・ファイルを構成する必要があります。

すべてのノードが使用可能で、すべてのアーカイブREDOログがバックアップされている場合は、データベースをマウントして任意のノードで次のコマンドを実行することで、完全なリストアおよびリカバリを実行できます。

```
RESTORE DATABASE;
RECOVER DATABASE;
```

ネットワーク・ファイル・システム構成では、各ノードにその他のノードのREDOログ・ファイルに対する読取りアクセス権があるため、リカバリ・ノードは、ローカルおよびリモート・ディスクにあるアーカイブREDOログの読取りおよび適用が可能です。手動によるアーカイブREDOログの転送は不要です。

関連項目

- 非クラスタ・ファイル・システムのローカル・アーカイブ・スキーム

RMANまたはOracle Enterprise Managerを使用したサーバー・パラメータ・ファイル(SPFILE)のリストア

RMANでは、サーバー・パラメータ・ファイルをデフォルト位置または指定された位置にリストアできます。

Oracle Enterprise Managerを使用して、SPFILEをリストアすることもできます。「メンテナンス」タブの「バックアップ/リカバリ」セクションで、「リカバリの実行」をクリックします。「リカバリの実行」リンクは状況依存のリンクであり、データベースがクローズしている場合にのみ、SPFILEのリストアにナビゲートされます。

Oracle RACでのインスタンス・リカバリ

Oracle RACでのインスタンス・リカバリについて学習します。

インスタンス障害は、ソフトウェアまたはハードウェアの問題によってインスタンスが無効になった場合に発生します。インスタンス障害の後、Oracle DatabaseはオンラインREDOログ・ファイルを使用して、次の各項で説明するデータベース・リカバリを自動的に実行します。

Oracle RACでの単一ノード障害

Oracle RACでのインスタンス・リカバリでは、障害が発生したインスタンス上で実行していたアプリケーションのリカバリは実行されません。Oracle Clusterwareがインスタンスを自動的に再起動します。

障害発生前にノードで実行中のアプリケーションは、障害の認識とリカバリの機能を使用して実行を継続します。これによって、ハードウェアまたはソフトウェアに障害が発生しても、一貫性のある連続的なサービスが提供されます。あるインスタンスが別のインスタンスのリカバリを実行する場合、障害が発生しなかったインスタンスは、障害が発生しているインスタンスによって生成されたオンラインREDOログ・エントリを読み取り、その情報を使用して、コミットされたすべてのトランザクションがデータベースに記録されるようにします。したがって、コミットされたトランザクションのデータが失われることはありません。 リカバリを実行中のインスタンスは、障害発生時にアクティブだったトランザクションをロールバックし、それらのトランザクションによって使用されたリソースを解放します。

ノート:

すべてのオンライン REDO ログは、インスタンスのリカバリのためにアクセスできる必要があります。オンライン REDO ログをミラー化することをお薦めします。

Oracle RACでの複数ノード障害

複数ノード障害が発生した場合、障害を受けなかったインスタンスが1つでもあれば、Oracle RACは、障害が発生したすべてのインスタンスに対してインスタンス・リカバリを実行します。Oracle RACデータベースのすべてのインスタンスに障害が発生した場合、Oracle Databaseは、次のインスタンスがデータベースをオープンするときにリカバリを自動的に実行します。リカバリを実行するインスタンスは、Oracle RACデータベースのどのノードからでも、クラスタ・データベースまたは排他モードでデータベースをマウントできます。このリカバリ手順は、1つのインスタンスが、障害が発生したすべてのインスタンスのリカバリを実行するという点以外は、共有モードで実行しているOracle Databaseでも、排他モードで実行しているOracle Databaseでも同じです。

Oracle RACでのRMANを使用したバックアップの作成

Oracle Databaseには、データベースのバックアップおよびリストアを行うRMANがあります。

RMANを使用すると、データ・ファイル、制御ファイル、SPFILEおよびアーカイブREDOログのバックアップ、リストアおよびリカバリを実行できます。RMANはOracle Databaseサーバーに含まれているため、デフォルトでインストールされます。RMANは、コマンドラインから実行するか、またはEnterprise ManagerのBackup Managerから使用できます。また、Oracle Automatic Storage Management(Oracle ASM)を使用している場合、バックアップおよびリカバリ・ツールとしてRMANを使用することをお薦めします。Oracle RAC環境でRMANを使用する手順は、非クラスタのOracle環境の場合とほぼ同じです。

関連項目

- 『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

RMANを使用したクラスタ・インスタンスへのチャネル接続

インスタンスへのチャネル接続は、チャネル構成で定義された接続文字列を使用して判別されます。たとえば、次の構成では、dbauser/pwd@service_nameを使用して3つのチャネルが割り当てられています。ロード・バランシングが有効の状態でSQL Netサービス名を構成した場合、ロード・バランシング・アルゴリズムによって決定されたノードにチャネルが割り当てられます。

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE SBT CONNECT 'dbauser/pwd@service_name'
```

ただし、接続文字列で使用されるサービス名がロード・バランシング用ではない場合、次のようにチャネル構成ごとに個別の接続文字列を使用して、どのインスタンスにチャネルを割り当てるかを制御できます。

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL 1.. CONNECT 'dbauser/pwd@mydb_1';
CONFIGURE CHANNEL 2.. CONNECT 'dbauser/pwd@mydb_2';
CONFIGURE CHANNEL 3.. CONNECT 'dbauser/pwd@mydb_3';
```

前述の例では、Oracle RAC環境の定義済ノードに接続するSQL*Netサービス名は、mydb_1、mydb_2およびmydb_3である

と仮定しています。また、手動で割り当てたチャネルを使用してデータベース・ファイルをバックアップすることもできます。たとえば、次のコマンドを実行すると、SPFILE、制御ファイル、データ・ファイルおよびアーカイブREDOログがバックアップされます。

```
RUN
{
    ALLOCATE CHANNEL CH1 CONNECT 'dbauser/pwd@mydb_1';
    ALLOCATE CHANNEL CH2 CONNECT 'dbauser/pwd@mydb_2';
    ALLOCATE CHANNEL CH3 CONNECT 'dbauser/pwd@mydb_3';
    BACKUP DATABASE PLUS ARCHIVED LOG;
}
```

少なくとも1つの割り当てられたチャネルからアーカイブ・ログにアクセス可能であれば、バックアップ操作中、RMANはそのチャネル上の特定のログのバックアップを自動的にスケジュールします。制御ファイル、SPFILEおよびデータ・ファイルはどのチャネルからでもアクセス可能であるため、これらのファイルのバックアップ操作は、割り当てられたチャネル間で分散されます。

ローカル・アーカイブ・スキームの場合、ローカル・アーカイブ・ログに記録するすべてのノードに1つ以上のチャネルが割り当てられている必要があります。クラスタ・ファイル・システムのアーカイブ・スキームでは、すべてのノードがアーカイブ・ログを同じクラスタ・ファイル・システムに書き込む場合、そのアーカイブ・ログのバックアップ操作は、割り当てられたチャネル間で分散されます。

バックアップの実行中は、チャネルの接続先インスタンスは、すべてマウントされているか、すべてオープン状態である必要があります。たとえば、node2とnode3のインスタンスにはオープン状態のデータベースがあるが、node1のインスタンスにマウントされたデータベースがある場合は、バックアップに失敗します。

関連項目

- 『Oracle Database Recovery Managerリファレンス』

関連項目:

CONFIGURE CHANNEL文のCONNECT句の詳細は、『*Oracle Database*バックアップおよびリカバリ・リファレンス』を参照してください。

高速接続のノード・アフィニティの認識

一部のクラスタ・データベース構成では、クラスタの一部のノードは、他のデータファイルに対するアクセスよりもより高速に特定のデータファイルにアクセスします。RMANはこの状況を自動的に検出し、これはノード・アフィニティの認識と呼ばれます。特定のデータファイルのバックアップに使用するチャネルを決定する際に、RMANは、バックアップするデータファイルに高速にアクセスするノードを優先します。たとえば、3ノードのクラスタがあり、node1がデータファイル7、8および9に対して他のノードより高速に読取りおよび書込みアクセスを行う場合、node1はnode2およびnode3に比べて、これらのファイルに対するノード・アフィニティが高いと言えます。

バックアップ完了後のアーカイブREDOログの削除

「RMANを使用したクラスタ・インスタンスへのチャネル接続」の説明に従って自動チャネルを構成した場合、次の例を使用してn回バックアップしたアーカイブ・ログを削除できます。デバイス・タイプは、DISKまたはSBTになります。

```
DELETE ARCHIVELOG ALL BACKED UP n TIMES TO DEVICE TYPE device_type;
```

少なくとも1つの割り当てられたチャネルからアーカイブ・ログにアクセス可能であれば、削除操作中、RMANはそのチャネル上の特定のログの削除を自動的にスケジュールします。ローカル・アーカイブ・スキームの場合、アーカイブ・ログを削除できる1つ以上のチャネルが割り当てられている必要があります。クラスタ・ファイル・システムのアーカイブ・スキームでは、すべてのノードが同じクラスタ・ファイル・システムのアーカイブ・ログに記録する場合、割り当てられた任意のチャネルからアーカイブ・ログを削除できます。

自動チャネルを構成していない場合、次のようにメンテナンス・チャネルを手動で割り当てて、アーカイブ・ログを削除できます。

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node1';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node2';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node3';
DELETE ARCHIVELOG ALL BACKED UP n TIMES TO DEVICE TYPE device_type;
```

関連項目

- RMANを使用したクラスタ・インスタンスへのチャネル接続

バックアップ・コマンドとリストア・コマンドのオートロケーション

RMANは、バックアップまたはリストアが必要なすべてのファイルのオートロケーションを自動的に実行します。非クラスタ・ファイル・システムのローカル・アーカイブ・スキームを使用している場合、各ノードが読み取ることができるのは、そのノードのインスタンスによって生成されたアーカイブREDOログのみです。RMANは、アーカイブREDOログを読み取れない場合、チャネルでのログのバックアップを試行しません。

リストアの操作時に、RMANはバックアップのオートロケーションを自動的に実行します。ノードにバックアップされたファイルのリストアが試行されるのは、特定のノードに接続されているチャネルのみです。たとえば、ログ順序番号1001はnode1に連結されているドライブにバックアップされ、ログ1002はnode2に連結されているドライブにバックアップされるとします。各ノードに接続するチャネルを割り当てる場合、node1に接続されたチャネルは(ログ1002ではなく)ログ1001をリストアでき、node2に接続されたチャネルは(ログ1001ではなく)ログ1002をリストアできます。

Oracle RACでのメディア・リカバリ

メディア・リカバリは、クライアント・アプリケーションを介してユーザーが起動する必要がありますが、インスタンス・リカバリは、データベースによって自動的に実行されます。この場合、RMANを使用してデータファイルのバックアップをリストアしてから、データベースをリカバリします。Oracle RAC環境でのRMANのメディア・リカバリ手順は、非クラスタ環境のメディア・リカバリ手順とほぼ同じです。

リカバリを実行するノードは、必要なデータ・ファイルをすべてリストアできることが必要です。また、このノードは、ディスク上の必要なアーカイブREDOログをすべて読み取ることができるか、バックアップしたデータ・ファイルをリストアできることが必要です。

暗号化された表領域を使用してデータベースをリカバリする場合(SHUTDOWN ABORTまたはデータベース・インスタンスをダウンさせた重大なエラーの発生後など)、リカバリ・プロセスでデータ・ブロックおよびREDOを復号できるように、データベースのマウント後、データベースを開く前にOracleウォレットを開く必要があります。

Oracle RACでのパラレル・リカバリ

Oracle Databaseでは、インスタンス・リカバリ、クラッシュ・リカバリ、メディア・リカバリの最適な並列度は自動的に選択されます。

CPUの可用性に基づいたパラレル・プロセスの最適な数を使用して、アーカイブREDOログが適用されます。次の項目で説明するように、Oracle RACデータベースでは、パラレル・インスタンス・リカバリおよびパラレル・メディア・リカバリを使用できます。

- RMANを使用したパラレル・リカバリ
- パラレル・リカバリの無効化

関連項目

- 『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

RMANを使用したパラレル・リカバリ

RMANのRESTOREおよびRECOVERコマンドを使用すると、次に示す3段階のリカバリ・プロセスが自動的にパラレル化されます。

データ・ファイルのリストア

データ・ファイルをリストアする場合、RMANのリカバリ・スクリプトに割り当てられているチャネル数によって、RMANが使用するパラレル化が効果的に設定されます。たとえば、5つのチャネルを割り当てると、最大5つのパラレル・ストリームでデータ・ファイルをリストアできます。

増分バックアップの適用

同様に、増分バックアップを適用する場合、割り当てるチャネル数によって潜在的なパラレル化が決定されます。

アーカイブREDOログの適用

RMANでは、アーカイブREDOログの適用はパラレルに実行されます。Oracle Databaseでは、使用可能なCPUリソースに基づいて最適な並列度が自動的に選択されます。

パラレル・リカバリの無効化

次の項目の手順を使用して、パラレル・リカバリを無効にできます。

- パラレル・インスタンス・リカバリおよびパラレル・クラッシュ・リカバリの無効化
- パラレル・メディア・リカバリの無効化

パラレル・インスタンス・リカバリおよびパラレル・クラッシュ・リカバリの無効化

複数CPUのシステムでパラレル・インスタンス・リカバリおよびパラレル・クラッシュ・リカバリを無効にするには、データベースの初期化パラメータ・ファイル(SPFILE)の RECOVERY_PARALLELISMパラメータを0(ゼロ)または1に設定します。

パラレル・メディア・リカバリの無効化

RMANのRECOVERコマンドまたはALTER DATABASE RECOVER文のNOPARALLEL句を使用すると、Oracle Databaseで強制的に非パラレル・メディア・リカバリが使用されます。

Oracle RACでの高速リカバリ領域の使用

Oracle RACで高速リカバリ領域を使用するには、Oracle ASMディスク・グループ、クラスタ・ファイル・システムまたは各

Oracle RACインスタンスのネットワーク・ファイル・システム・ファイルで構成される共有ディレクトリに配置される必要があります。

つまり、高速リカバリ領域はOracle RACデータベースのすべてのインスタンス間で共有される必要があります。また、すべてのインスタンスに対してDB_RECOVERY_FILE_DESTパラメータに同じ値を設定します。

Oracle Enterprise Managerを使用すると、高速リカバリ領域を設定できます。この機能を使用するには:

1. クラスタ・データベースの「ホーム」ページで、「メンテナンス」タブをクリックします。
2. 「バックアップ/リカバリ」オプション・リストから、「リカバリ設定の構成」をクリックします。
3. ページの「高速リカバリ領域」セクションで、要件を指定します。
4. 詳細は、このページの「ヘルプ」をクリックしてください。

関連項目

- 『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

9 新規クラスタのノードへのOracle RACのクローニング

この章では、LinuxおよびUNIXシステムで、Oracle Real Application Clusters(Oracle RAC)データベース・ホームを新規クラスタのノードにクローニングする方法について説明します。

この章では、スクリプトを使用して実装する非対話式のクローニング技術について説明します。この章で説明するクローニング技術は、複数の同時クラスタ・インストールを実行する場合に最適です。スクリプトの作成は手動プロセスであり、間違いが発生する可能性があります。1つのクラスタのみをインストールする場合は、Oracle Universal InstallerやOracle Enterprise ManagerのProvisioning Pack機能などの従来の自動化された対話式のインストール方法を使用してください。

ノート:

クローニングは、Provisioning Pack に含まれる Oracle Enterprise Manager クローニングにかわるものではありません。Oracle Enterprise Manager クローニングの間、プロビジョニング・プロセスでは、Oracle ホームに関する詳細(クローニングのデプロイ先の位置、Oracle Database ホームの名前、クラスタ内のノードのリストなど)が対話式で確認されます。

Oracle Enterprise Manager Cloud Control の Provisioning Pack 機能のフレームワークによって、新規ノードおよびクラスタのプロビジョニングを簡単に自動化できます。多数の Oracle RAC クラスタを持つデータ・センターの場合、既存のクラスタへの新規クラスタおよび新規ノードのプロビジョニングを簡素化するクローニング手順を作成しておくと有効です。

この章の内容は次のとおりです。

- Oracle RACのクローニングの概要
- Oracle RACのクローニングの準備
- クラスタのノードへのOracle RACのクローニングのデプロイ
- クローニング時に生成されたログ・ファイルの検索および表示

関連項目

- クローニングを使用した同じクラスタのノードへのOracle RACの拡張

Oracle RACのクローニングの概要

クローニングとは、既存のOracle RACインストールを別の位置にコピーし、コピーしたインストールを新しい環境で動作するように更新するプロセスのことです。ソースOracleホームで適用された1回限りのパッチによって行われた変更は、クローニング操作後も存在します。ソース・パスと宛先パス(クローニング対象のホスト)は同じである必要はありません。

クローニングは、次のような状況で役立ちます。

- クローニングによって、Oracleホームを一度準備してから、同時に多数のホストにデプロイできます。非対話形式のプロセスとして、インストールをサイレントに完了できます。グラフィカル・ユーザー・インタフェース(GUI)のコンソールを使用する必要がなく、必要に応じて、セキュア・シェル(SSH)・ターミナル・セッションからクローニングを実行できます。

- クローニングにより、すべてのパッチが適用されたインストール(本番、テスト、または開発用インストールのコピー)を1回のステップで作成できます。ベースとなるインストールを実行し、すべてのパッチ・セットとパッチをソース・システムに適用した後、これらの個々のステップのすべてを1つの手順としてクローニングで実行します。これにより、クラスタの各ノードですべてのインストール・プロセスを通して行い、インストール、構成およびパッチ適用を個別のステップで実行するのとは、対照的な効果が得られます。
- クローニングによるOracle RACのインストールは、非常に迅速なプロセスです。たとえば、3つ以上のノードの新規クラスタにOracleホームをクローニングする場合、Oracleベース・ソフトウェアのインストールに数分と各ノードに対して数分ずつ必要となります(root. shスクリプトの実行に要する時間とほぼ同じ)。

クローン・インストールは、ソース・インストールと同様に動作します。たとえば、クローンOracleホームは、Oracle Universal Installerを使用して削除したり、OPatchを使用してパッチを適用することができます。また、クローンOracleホームを別のクローニング操作のソースとして使用することもできます。コマンドラインのクローニング・スクリプトを使用して、テスト用、開発用または本番用のインストールをクローニングしたコピーを作成できます。デフォルトのクローニング手順は、ほとんどの使用例に適しています。ただし、カスタム・ポート割当ての指定やカスタム設定の保存など、クローニングの様々な側面をカスタマイズすることもできます。

クローニング・プロセスでは、ソースOracleホームから宛先Oracleホームにすべてのファイルがコピーされます。このため、ソースOracleホームのディレクトリ構造外にあるソース・インスタンスによって使用されるすべてのファイルは、宛先位置にコピーされません。

ソースと宛先でバイナリのサイズが異なる場合がありますが、これは、これらがクローン操作の一部として再リンクされており、これら2つの位置のオペレーティング・システムのパッチ・レベルが異なっていることがあるためです。また、いくつかのファイルがソースからコピーされることによって、特にインスタンス化されたファイルはクローン操作の一部としてバックアップされるため、クローン・ホームのファイルの数が増える場合があります。

Oracle RACのクローニングの準備

この概要を使用して、Oracle RACのクローニングに使用する手順について理解してください。

準備フェーズでは、Oracleホームのコピーを作成し、次にそのコピーを使用して1つ以上のノードでクローニング手順を実行します。Oracle Clusterwareもインストールします。

Oracle RACのインストール

ご使用のプラットフォーム用の『*Oracle Real Application Clusters*インストレーション・ガイド』の詳細な手順に従って、Oracle RACソフトウェアおよびパッチをインストールします。

1. Oracle RACをインストールし、ソフトウェアのみインストール・オプションを選択します。
2. リリースに必要なレベル(12.1.0.nなど)のパッチを適用します。
3. 必要に応じて、1回限りのパッチを適用します。

ソース・ホームのバックアップの作成

Oracle RACホームのコピーを作成します。このファイルを使用して、クラスタ内の各ノードにOracle RACホームをコピーします。

バックアップ(tar)ファイルを作成する場合は、ファイルの名前にリリース番号を含めることをお薦めします。たとえば:

```
# cd /opt/oracle/product/12c/db_1
# tar -zcvf /pathname/db1120.tgz .
```

Oracle Clusterwareのインストールおよび起動

クローニングを使用してOracle RACホームを作成する前に、クローニングされたOracle RACホームをコピーするノードにOracle Clusterwareをインストールし、起動しておく必要があります。つまり、元のノードでOracle ClusterwareとOracle RACソフトウェア・コンポーネントをインストールしたときと同じ順序で、ソース・クラスタからターゲット・クラスタのノードにクローニングしたOracle RACホームを構成します。

関連項目

- 『Oracle Real Application Clustersインストレーション・ガイド』
- クラスタのノードへのOracle RACのクローニングのデプロイ
- Oracle Clusterware管理およびデプロイメント・ガイド

クラスタのノードへのOracle RACのクローニングのデプロイ

「Oracle RACのクローニングの準備」の項で説明した前提条件のタスクを完了した後、クローニングしたOracleホームをデプロイできます。

次のように、Oracle RACデータベース・ホームをクラスタにデプロイします。

1. ご使用のプラットフォームに固有のOracle RACインストレーション・ガイドの説明に従い、次のようなOracle RACのインストール前のタスクを実行して新しいクラスタ・ノードを準備します。
    - カーネル・パラメータを指定します。
    - Oracle Clusterwareがアクティブであることを確認します。
    - Oracle ASMがアクティブであり、少なくとも1つのOracle ASMディスク・グループが存在し、マウント済であることを確認します。
2. 次のように、Oracle RACデータベース・ソフトウェアをデプロイします。
    - Oracleホームのクローンをすべてのノードにコピーします。たとえば:

      ```
      [root@node1 root]# mkdir -p /opt/oracle/product/12c/db
      [root@node1 root]# cd /opt/oracle/product/12c/db
      [root@node1 db]# tar -zxvf /path_name/db1120.tgz
      ```

      ホームの位置とpath_nameの指定では、ホームの位置を、tarの作成に使用したソース・ホームと同じディレクトリ・パスにすることも、異なるディレクトリ・パスにすることもできます。
    - oracleユーザーまたはoinstallグループ(あるいはその両方)が、ソース・ノードと宛先ノード間で異なる場合は、次のように、Oracle Inventoryファイルの所有権を変更します。

      ```
      [root@node1]# chown -R oracle:oinstall /opt/oracle/product/12c/db
      ```

      Oracle RACホームで前述のコマンドを実行すると、Oracleバイナリからsetuidとsetgidの情報がクリアされます。

ノート:

このステップは、各クラスタ・ノードで clone.pl スクリプトおよび$ORACLE_HOME/root.sh スクリプトを実行するステップ 3 および 4 と同時に実行できます。

3. 次のように、主なOracle RACクローニング・タスクを実行するclone.plスクリプトを各ノードで実行します。

- 表9-2と表9-3に示すように、start.shスクリプトに環境変数とクローニング・パラメータを指定します。clone.plスクリプトでは、渡されるパラメータが厳密に識別されるため、括弧、一重引用符および二重引用符を正確に使用する必要があります。
- oracleまたはOracle RACソフトウェアを所有するユーザーとしてスクリプトを実行します。

次の表に、clone.plスクリプト・パラメータとその説明を示します。

表9-1 clone.plスクリプト・パラメータ

パラメータ	説明
ORACLE_HOME=Oracle_home	クローニングする Oracle ホームへの完全なパス。無効なパスを指定すると、スクリプトは終了します。このパラメータは必須です。
ORACLE_BASE=ORACLE_BASE	クローニングする Oracle ベースへの完全なパス。無効なパスを指定すると、スクリプトは終了します。このパラメータは必須です。
ORACLE_HOME_NAME=Oracle_home_name | -defaultHomeName	クローニングするホームの Oracle ホーム名。必要に応じて、-defaultHomeName フラグを指定できます。このパラメータは省略可能です。
ORACLE_HOME_USER=Oracle_home_user	Windows サービス用の Oracle ホーム・ユーザー。このパラメータは Windows にのみ適用でき、オプションです。
OSDBA_GROUP=group_name	OSDBA 権限が付与されたグループとして使用するオペレーティング・システム・グループを指定します。このパラメータは省略可能です。
OSOPER_GROUP=group_name	OSOPER 権限が付与されたグループとして使用するオペレーティング・システム・グループを指定します。このパラメータは省略可能です。
OSASM_GROUP=group_name	OSASM 権限が付与されたグループとして使用するオペレーティング・システム・グループを指定します。このパラメータは省略可能です。
OSBACKUPDBA_GROUP=group_name	OSBACKUPDBA 権限が付与されたグループとして使用するオペレーティング・システム・グループを指定します。このパラメータは省略可能です。
OSDGDBA_GROUP=group_name	OSDGDBA 権限が付与されたグループとして使用するオペレーティング・システム・グループを指定します。このパラメータは省略可能です。

パラメータ	説明
OSKMDBA_GROUP=group_name	OSKMDBA 権限が付与されたグループとして使用するオペレーティング・システム・グループを指定します。このパラメータは省略可能です。
-debug	このオプションは、clone.pl スクリプトをデバッグ・モードで実行する場合に指定します。
-help	このオプションは、clone.pl スクリプトのヘルプを表示する場合に指定します。

次の例に、clone.plスクリプトをコールするstart.shスクリプトからの抜粋を示します。

```
ORACLE_BASE=/opt/oracle
ORACLE_HOME=/opt/oracle/product/12c/db
cd $ORACLE_HOME/clone
THISNODE='host_name'
E01=ORACLE_HOME=/opt/oracle/product/12c/db
E02=ORACLE_HOME_NAME=OraDBRAC
E03=ORACLE_BASE=/opt/oracle
C01="-O CLUSTER_NODES={node1,node2}"
C02="-O LOCAL_NODE=$THISNODE"
perl $ORACLE_HOME/clone/bin/clone.pl $E01 $E02 $E03 $C01 $C02
```

次の表に、前の例に太字で示されている環境変数E01、E02およびE03とその説明を示します。

表9-2 clone.plスクリプトに渡される環境変数

記号	変数	説明
E01	ORACLE_HOME	Oracle RAC データベース・ホームの位置。このディレクトリ位置が存在し、Oracle オペレーティング・システム・グループ oinstall によって所有されている必要があります。
E02	ORACLE_HOME_NAME	Oracle RAC データベースの Oracle ホームの名前。Oracle インベントリに格納されています。
E03	ORACLE_BASE	Oracle ベース・ディレクトリの位置。

次の表に、前の例に太字で示されているクローニング・パラメータC01およびC02とその説明を示します。

表9-3 clone.plスクリプトに渡されるクローニング・パラメータ。

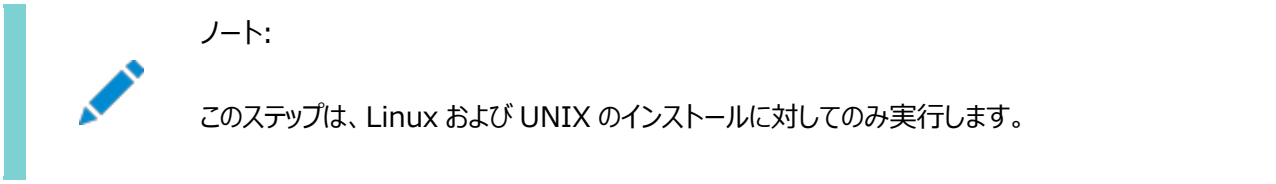
変数	名前	パラメータ	説明
C01	クラスタ・ノード	CLUSTER_NODES	クラスタ内のノードのリストを表示します。
C02	ローカル・ノード	LOCAL_NODE	ローカル・ノードの名前。

次の例に、ユーザーが作成する必要があり、clone.plスクリプトをコールするstart.batスクリプトからの抜粋を示しま

す。

```
set ORACLE_home=C:¥oracle¥product¥12c¥db1
cd %ORACLE_home%¥clone¥bin
set THISNODE=%hostname%
set E01=ORACLE_HOME=%ORACLE_home%
set E02=ORACLE_HOME_NAME=OraDBRAC
set E03=ORACLE_BASE=Oracle_Base
set C01="CLUSTER_NODES={node1,node2}"
set C02="-O LOCAL_NODE=%THISNODE%"
perl clone.pl %E01% %E02% %E03% %C01% %C02%
```

4.



ノート:

このステップは、Linux および UNIX のインストールに対してのみ実行します。

ノードでclone.plプロシージャが完了した後、すぐにrootオペレーティング・システム・ユーザーとして$ORACLE_HOME/root.shを実行します。

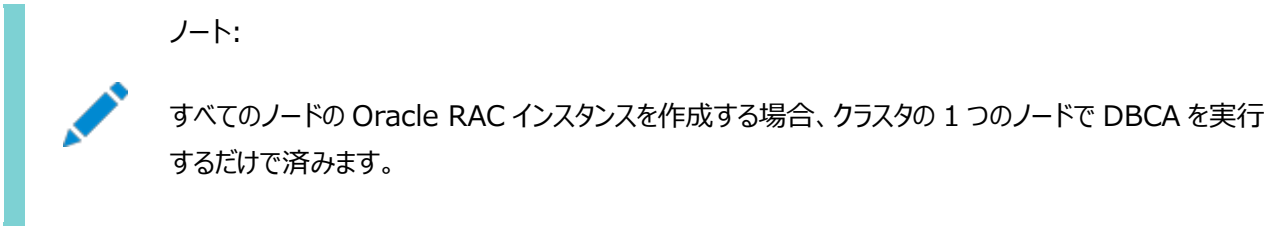
```
[root@node1 root]# /opt/oracle/product/12c/db/root.sh -silent
```

各ノードでスクリプトを同時に実行できます。

```
[root@node2 root]# /opt/oracle/product/12c/db/root.sh -silent
```

各ノードでスクリプトが完了していることを確認し、次のステップに進みます。

5.



ノート:

すべてのノードの Oracle RAC インスタンスを作成する場合、クラスタの 1 つのノードで DBCA を実行するだけで済みます。

このステップでは、DBCAをサイレント・モードで実行し、レスポンス・ファイルに入力してOracle RACインスタンスを作成する方法を示します。

次の例では、ERIという名前のOracle RACデータベースを各ノードで作成し、各ノードにデータベース・インスタンスを作成し、そのインスタンスをOCRに登録し、DATAと呼ばれるOracle ASMディスク・グループにデータベース・ファイルを作成し、サンプル・スキーマを作成します。また、SYS、SYSTEM、SYSMANおよびDBSNMPパスワードをpassword(各アカウントのパスワード)に設定します。

```
[oracle@node1 oracle]$ export ORACLE_HOME=/opt/oracle/product/12c/db
[oracle@node1 oracle]$ cd $ORACLE_HOME/bin/
[oracle@node1 bin]$./dbca -silent -createDatabase -templateName General_Purpose.dbc ¥
-gdbName ERI -sid ERI ¥
-sysPassword password -systemPassword password ¥
-sysmanPassword password -dbsnmpPassword password ¥
-emConfiguration LOCAL ¥
-storageType ASM -diskGroupName DATA ¥
-datafileJarLocation $ORACLE_HOME/assistants/dbca/templates ¥
-nodelist node1,node2 -characterset WE8ISO8859P1 ¥
-obfuscatedPasswords false -sampleSchema true
```

関連項目

- Oracle RACのクローニングの準備
- 『Oracle Real Application Clustersインストレーション・ガイド』
- Oracle Database 2日でデータベース管理者

クローニング時に生成されたログ・ファイルの配置と表示

クローニング・スクリプトは複数のツールを実行し、それぞれが独自にログ・ファイルを生成する可能性があります。

clone.plスクリプトの実行が終了した後、ログ・ファイルを参照して、クローニング・プロセスに関する詳細情報を入手できます。

クローニング時に生成された次のログ・ファイルは、診断のための主要なログ・ファイルです。

- Central_Inventory/logs/cloneActionstimestamp.log

  クローニングのOracle Universal Installerの部分で発生したアクションの詳細なログが含まれます。
- Central_Inventory/logs/oraInstalltimestamp.err

  Oracle Universal Installerの実行時に発生したエラーに関する情報が含まれます。
- Central_Inventory/logs/oraInstalltimestamp.out

  Oracle Universal Installerによって生成されたその他のメッセージが含まれます。
- $ORACLE_HOME/clone/logs/clonetimestamp.log

  クローニング前とクローニング操作中に発生したアクションの詳細なログが含まれます。
- $ORACLE_HOME/clone/logs/errortimestamp.log

  クローニング前とクローニング操作中に発生したエラーに関する情報が含まれます。

表9-4に、Oracleインベントリ・ディレクトリの位置を確認する方法を示します。

表9-4 Oracleインベントリ・ディレクトリの位置の確認

システムのタイプ	Oracleインベントリ・ディレクトリの位置
Linux および IBM AIX を除くすべての UNIX コンピュータ	/var/opt/oracle/oraInst.loc
IBM AIX および Linux	/etc/oraInst.loc ファイル。
Windows	C:¥Program Files¥Oracle¥Inventory

10 クローニングを使用した同じクラスタのノードへのOracle RACの拡張

この章では、クローニングを使用してOracle Real Application Clusters(Oracle RAC)を既存クラスタのノードに拡張する方法について説明します。

Oracle RACを新規クラスタのノードに追加する場合は、「新規クラスタのノードへのOracle RACのクローニング」を参照してください。

この章のトピックは、次のとおりです:

- Oracle RAC環境でのクローニングを使用したノードの追加について
- LinuxおよびUNIXシステムでのローカルOracleホームのクローニング
- LinuxおよびUNIXシステムでの共有Oracleホームのクローニング
- WindowsシステムでのOracleホームのクローニング

関連項目

- 新規クラスタのノードへのOracle RACのクローニング
- Oracle RACのクローニングの概要
- Oracle Clusterware管理およびデプロイメント・ガイド

Oracle RAC環境でのクローニングを使用したノードの追加について

クローニング手順は、ノードおよびインスタンスを追加するOracle RAC環境が正常にインストールされ、構成されていることを前提としています。クローニングを使用してOracle RAC環境にノードを追加するには、まず、Oracle Clusterware構成を拡張し、次にOracle RACによってOracle Databaseソフトウェアを拡張し、Oracle Assistantを実行してリスナーとインスタンスを追加します。

クローニング・スクリプトは複数のツールを実行し、それぞれが独自にログ・ファイルを生成する可能性があります。clone.plスクリプトの実行が終了した後、ログ・ファイルを参照して、クローニング・プロセスに関する詳細情報を入手できます。詳細は、「クローニング時に生成されたログ・ファイルの配置と表示」を参照してください。

LinuxおよびUNIXシステムでのローカルOracleホームのクローニング

LinuxおよびUNIX環境でローカルの非共有Oracleホームをクローニングして、Oracle RAC環境にノードを追加します。

Oracle RACとともにOracle Databaseをクローニングするには、次のステップを実行します。

1. Oracleホームのコピーを作成し、このコピーを使用して1つ以上のノードでクローニング手順を実行するには、トピック「Oracle RACのクローニングの準備」のステップに従ってください。
2. tarユーティリティを使用して、既存ノードでOracleホームのアーカイブを作成し、それを新規ノードにコピーします。ソース・ノードのOracleホームの場所が$ORACLE_HOMEである場合は、これと同じディレクトリを新規ノード上のコピー先として使用する必要があります。

3. 新規ノードで、環境変数ORACLE_HOMEおよびORACLE_BASEを構成します。その後で、新しいノードのGrid_home/clone/binディレクトリに移動して、次のコマンドを実行します。existing_nodeはクローニングするノードの名前、new_node2とnew_node3は新しいノードの名前、Oracle_home_nameはOracleホームの名前です。

```
perl clone.pl ORACLE_HOME=$ORACLE_HOME ORACLE_HOME_NAME=Oracle_home_name
  ORACLE_BASE=$ORACLE_BASE "'CLUSTER_NODES={existing_node,new_node2,new_node3}'"
  "'LOCAL_NODE=new_node2'" CRS=TRUE INVENTORY_LOCATION=/u01/app/oraInventory
```

4. 次のコマンドを実行し、新規ノードでOracle RACを構成するためのConfiguration Assistantを実行します。

```
$ORACLE_HOME/cfgtoollogs/configToolFailedCommands
```

このスクリプトには、インストール中に失敗、スキップまたはキャンセルされたすべてのコマンドが含まれています。このスクリプトは、Oracle Universal Installer外でデータベース構成アシスタントを実行するために使用できます。スクリプトの実行前に、スクリプト内のパスワードを更新する必要があるかどうかを確認してください。

5. 既存ノードで、$ORACLE_HOME/oui/binディレクトリから次のコマンドを実行し、Oracle RACを含むOracle Databaseホーム(Oracle_homeで指定)のインベントリを更新します。existing_nodeはクローニング元のノードの名前、new_node2およびnew_node3は新規ノードの名前です。

```
./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME -O "CLUSTER_NODES=
{existing_node,new_node2,new_node3}"
```

6. それぞれの新規ノードで、$ORACLE_HOMEディレクトリに移動し、次のコマンドを実行します。

```
./root.sh
```

7. クローニングしたノードから、Oracle Database Configuration Assistant (Oracle DBCA)を実行して、新規ノードにOracle RACデータベース・インスタンスを追加します。

関連項目

- Oracle RACのクローニングの準備

LinuxおよびUNIXシステムでの共有Oracleホームのクローニング

LinuxおよびUNIXシステム環境で共有Oracleホームをクローニングして、既存のOracle RAC環境にノードを追加します。

Oracle DatabaseとOracle RACソフトウェアをクローニングするには、次のステップを実行します。

1. Oracleホームのコピーを作成し、このコピーを使用して1つ以上のノードでクローニング手順を実行するには、「Oracle RACのクローニングの準備」のステップに従ってください。
2. 新規ノードで、環境変数ORACLE_HOMEおよびORACLE_BASEを構成します。次に、$ORACLE_HOME/clone/binディレクトリに移動し、次のコマンドを実行します(ここで、existing_nodeはクローニングするノードの名前、new_node2,およびnew_node3は新規ノードの名前、Oracle_home_nameはOracleホームの名前で、-cfsオプションはOracleホームが共有されることを示します)。

```
perl clone.pl -O 'CLUSTER_NODES={existing_node,new_node2,new_node3}'
-O LOCAL_NODE=new_node2 ORACLE_BASE=$ORACLE_BASE ORACLE_HOME=$ORACLE_HOME
 ORACLE_HOME_NAME=Oracle_home_name [-cfs]
```

ノート:

このコマンドの内容は次のとおりです。

- -cfs オプションは、Oracle RAC を含む共有 Oracle Database ホームに使用します。
- ORACLE_HOME_NAME パラメータの値は、クローニングするノードの値である必要があります。

3. 既存ノードで、$ORACLE_HOME/oui/binディレクトリから次のコマンドを実行し、Oracle RACを含むOracle Databaseホーム(Oracle_homeで指定)のインベントリを更新します。existing_nodeはクローニング元のノードの名前、new_node2およびnew_node3は新規ノードの名前です。

```
./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME "CLUSTER_NODES=
{existing_node,new_node2,new_node3}"
```

4. それぞれの新規ノードで、$ORACLE_HOMEディレクトリに移動し、次のコマンドを実行します。

```
./root.sh
```

5. クローニングしたノードから、Database Configuration Assistant(DBCA)を実行して、新規ノードにOracle RACデータベース・インスタンスを追加します。

関連項目

- Oracle RACのクローニングの準備

WindowsシステムでのOracleホームのクローニング

Windowsシステム環境で共有またはローカルOracleホームをクローニングして、既存のOracle RAC環境にノードを追加します。

Oracle DatabaseとOracle RACソフトウェアをクローニングするには、次のステップを実行します。

1. ローカルOracleホームがある場合は、ZIPユーティリティを使用して、既存ノードでOracle RACを含むOracle Databaseホームのアーカイブを作成し、それを新規ノードにコピーします。それ以外の場合は、次のステップに進みます。

   ZIPファイルのOracle DatabaseとOracle RACのホームのファイルを、新規ノードで同じディレクトリ(既存ノードでOracle RACとともにOracle Databaseホームが配置されていた場所)に解凍します。たとえば、新規ノードのコピー先のOracle RACのホームの場所を%ORACLE_HOME%とします。

2. 新規ノードで、%ORACLE_HOME%¥clone¥binディレクトリに移動し、次のコマンドを実行します(ここで、Oracle_HomeはOracle Databaseホーム、Oracle_Home_NameはOracle Databaseホームの名前、Oracle_BaseはOracleベース・ディレクトリ、user_nameはクローニングされるOracleホームのOracleホーム・ユーザー(管理者以外のユーザー)の名前、existing_nodeは既存のノードの名前、new_nodeは新規ノードの名前です)。

```
perl clone.pl ORACLE_HOME=Oracle_Home ORACLE_BASE=Oracle_Base
ORACLE_HOME_NAME=Oracle_Home_Name ORACLE_HOME_USER=user_name
-O 'CLUSTER_NODES={existing_node,new_node}'
-O LOCAL_NODE=new_node
```

   Oracle RACを含む共有Oracle Databaseホームがある場合は、コマンドに-cfsオプションを追加し、Oracleホー

ムが共有されていることを示します。次に例を示します。

```
perl clone.pl ORACLE_HOME=Oracle_Home ORACLE_BASE=Oracle_Base
ORACLE_HOME_NAME=Oracle_Home_Name ORACLE_HOME_USER=user_name
-O 'CLUSTER_NODES={existing_node,new_node}' -O LOCAL_NODE=new_node
[-cfs -noConfig]
```

ノート:

- セキュアな Oracle ホームをクローニングする場合にのみ、ORACLE_HOME_USER が必須です。
- -cfs および-noConfig オプションは、Oracle RAC を含む共有 Oracle Database ホームに使用します。
- ORACLE_HOME_NAME パラメータの値は、クローニングするノードの値である必要があります。ORACLE_HOME_NAME を取得するには、HKEY_LOCAL_MACHINE¥SOFTWARE¥oracle¥KEY_OraCRs12c_home1 の下の ORACLE_HOME_NAME パラメータ・キーを、クローニングするノードのレジストリで検索します。

3. 既存ノードで、%ORACLE_HOME%¥oui¥binディレクトリから次のコマンドを実行し、Oracle_homeで指定されるOracle RACを含むOracle Databaseホームのインベントリを更新します。existing_nodeは既存ノードの名前、new_nodeは新規ノードの名前です。

```
setup.exe -updateNodeList ORACLE_HOME=Oracle_home "CLUSTER_NODES=
{existing_node,new_node}" LOCAL_NODE=existing_node
```

4. クローニングしたノードから、DBCAを実行して、新規ノードにOracle RACデータベース・インスタンスを追加します。

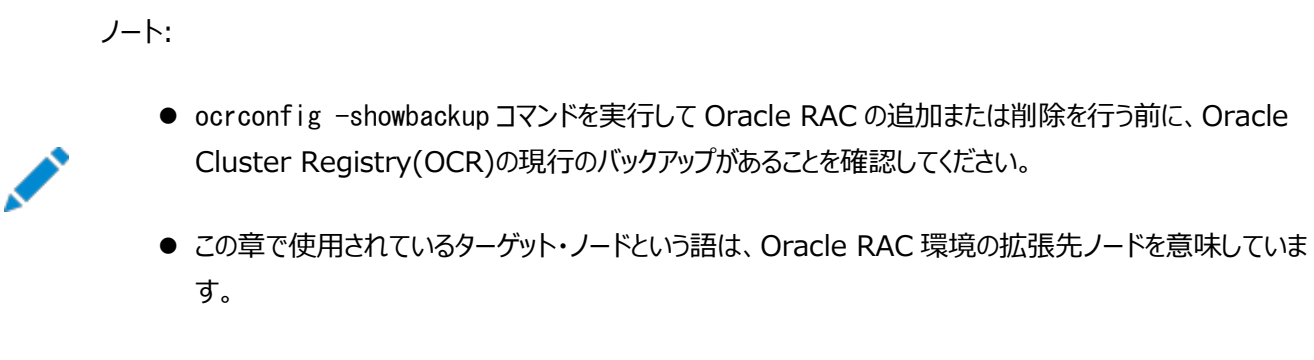
11 LinuxおよびUNIXシステムのノードでのOracle RACの追加と削除

既存のOracle Real Application Clusters(Oracle RAC)ホームを他のノードおよびクラスタ内のインスタンスに拡張し、Oracle RACをノードおよびクラスタ内のインスタンスから削除します。

既存のOracle RACホームをクローニングし、クラスタ全体で複数の新規Oracle RACインストールを作成する場合は、「新規クラスタのノードへのOracle RACのクローニング」に示すクローニング手順に従います。

内容は次のとおりです。

- Oracle Clusterwareがインストールされたノードへの Oracle RACの追加
- クラスタ・ノードからのOracle RACの削除



ノート:

- ocrconfig -showbackup コマンドを実行して Oracle RAC の追加または削除を行う前に、Oracle Cluster Registry(OCR)の現行のバックアップがあることを確認してください。
- この章で使用されているターゲット・ノードという語は、Oracle RAC 環境の拡張先ノードを意味しています。

関連項目

- 新規クラスタのノードへのOracle RACのクローニング
- WindowsシステムのノードでのOracle RACの追加と削除

Oracle Clusterwareがインストールされたノードへの Oracle RACの追加

この手順を開始する前に、既存ノードのGrid_homeへのパスが正しいこと、および$ORACLE_HOME環境変数がOracle RACホームに設定されていることを確認します。

- ローカル(非共有)のOracleホームを使用している場合は、クラスタの既存のノード(この手順ではnode1)にあるOracle RACデータベース・ホームをターゲット・ノード(この手順ではnode3)に拡張する必要があります。
  1. node1のOracle_home/addnodeディレクトリに移動し、addnode.shスクリプトを実行します。
  2. サイレント・インストールを実行する場合は、次の構文を使用してaddnode.shスクリプトを実行します。

     ```
     $ ./addnode.sh -silent "CLUSTER_NEW_NODES={node3}"
     ```

  3. rootとして、node3でOracle_home/root.shスクリプトを実行します。
  4. SQL*Plusセッションで次のコマンドを使用して、新しく追加したノードでプラガブル・データベース(PDB)をオープンします。

     ```
     SQL> CONNECT / AS SYSDBA
     SQL> ALTER PLUGGABLE DATABASE pdb_name OPEN;
     ```

- Oracle Automatic Storage Management Cluster File System (Oracle ACFS)を使用して共有されている共有Oracleホームを持つ場合は、次の手順を実行し、Oracle Databaseホームをnode3に拡張します。

  1. rootとしてGrid_home/binディレクトリから次のコマンドを実行して、新しいノードでOracle ACFSリソースを起動します。

     ```
     # srvctl start filesystem -device volume_device [-node node_name]
     ```

     ノート:

     Oracle ホームが格納されている Oracle ACFS レジストリ・リソースおよび Oracle ACFS ファイル・システム・リソースを含む Oracle ACFS リソースが、新しく追加されたノードでオンラインであることを確認します。

  2. Oracle RACをインストールしたユーザーとして、追加したノードのOracle_home/oui/binディレクトリから次のコマンドを実行し、Oracle RACデータベース・ホームを追加します。

     ```
     $ ./runInstaller -attachHome ORACLE_HOME="ORACLE_HOME" "CLUSTER_NODES={node3}"
       LOCAL_NODE="node3" ORACLE_HOME_NAME="home_name" -cfs
     ```

  3. node1のOracle_home/addnodeディレクトリに移動し、次の構文を使用してOracle RACをインストールしたユーザーとしてaddnode.shスクリプトを実行します。

     ```
     $ ./addnode.sh -noCopy "CLUSTER_NEW_NODES={node3}"
     ```

     ノート:

     宛先ノードの Oracle ホームはすでにソフトウェアに完全に移入されているため、-noCopy オプションを使用します。

- Oracle ACFS以外の共有ファイル・システムで共有Oracleホームを持つ場合は、まずターゲット・ノードでOracle RACデータベース・ホームのマウント・ポイントを作成し、Oracle RACデータベース・ホームをマウントおよびアタッチし、次のようにOracle Inventoryを更新します。

  1. クラスタ内の既存ノードでsrvctl config database -db db_nameコマンドを実行して、マウント・ポイント情報を取得します。
  2. 次のコマンドをrootとしてnode3で実行し、マウント・ポイントを作成します。

     ```
     # mkdir -p mount_point_path
     ```

  3. Oracle RACデータベース・ホームをホストするファイル・システムをマウントします。
  4. Oracle RACをインストールしたユーザーとして、追加したノードのOracle_home/oui/binディレクトリから次のコマンドを実行し、Oracle RACデータベース・ホームを追加します。

     ```
     $ ./runInstaller -attachHome ORACLE_HOME="ORACLE_HOME" "CLUSTER_NODES=
       {local_node_name}" LOCAL_NODE="node_name" ORACLE_HOME_NAME="home_name"
     ```

  5. Oracle RACをインストールしたユーザーとして、次のようにOracle Inventoryを更新します。

```
$ ./runInstaller -updateNodeList ORACLE_HOME=mount_point_path "CLUSTER_NODES=
  {node_list}"
```

前述のコマンドで、node_listはOracle RACデータベース・ホームがインストールされたすべてのノード(追加したものも含む)のリストです。

rootとして、node3でOracle_home/root.shスクリプトを実行します。

ノート:

ノード追加プロセスの終了後は、OCR をバックアップすることをお薦めします。

これで、次の項のいずれかの手順を使用して、ターゲット・ノードにOracle RACデータベース・インスタンスを追加できます。

- ターゲット・ノードへのポリシー管理Oracle RACデータベース・インスタンスの追加
- ターゲット・ノードへの管理者管理Oracle RACデータベース・インスタンスの追加

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

ターゲット・ノードへのポリシー管理Oracle RACデータベース・インスタンスの追加

ポリシー管理データベースをOracle Automatic Storage Management(Oracle ASM)に格納しないかぎり、およびOracle Managed Filesが有効でないかぎり、UNDOログおよびREDOログを手動で追加する必要があります。

サーバー・プールにノードを追加するための空き領域があり、データベースが1回以上起動されている場合、Oracle ClusterwareはOracle RACデータベース・インスタンスを新しく追加したノードに追加するため、それ以上の処置は必要ありません。

ノート:

新しく追加したノードにデータベース・インスタンスを追加する前に、データベースは 1 回以上起動されている必要があります。

サーバー・プールに空き領域がない場合は、新しく追加したノードは空きサーバー・プールに移動します。srvctl modify srvpoolコマンドを使用して、新しく追加したノードを格納するサーバー・プールのカーディナリティを上げた後に、ノードは空きサーバー・プールから修正済サーバー・プールに移動し、Oracle ClusterwareはOracle RACデータベース・インスタンスをそのノードに追加します。

ターゲット・ノードへの管理者管理Oracle RACデータベース・インスタンスの追加

ノート:

この項の手順は、管理者管理データベースに対してのみ実行してください。ポリシー管理データベースは、そのデー

タベースのサーバー・プールでノードが使用可能な場合にノードを使用します。

Oracle Enterprise ManagerまたはDBCAを使用して、Oracle RACデータベース・インスタンスをターゲット・ノードに追加できます。

この項では、DBCAを使用してOracle RACデータベース・インスタンスを追加する方法について説明します。

これらのツールは、次のタスクをガイドします。

- 各ターゲット・ノードでの新規データベース・インスタンスの作成
- 高可用性コンポーネントの作成および構成
- Oracleホームからのデフォルト以外のリスナー用のOracle Net構成の作成
- 新規インスタンスの起動
- サービス構成ページでサービス情報を入力した場合、サービスの作成および起動

ターゲット・ノードにインスタンスを追加した後で、「動的データベース・サービスによるワークロード管理」の説明に従って、必要なサービス構成手順を実行する必要があります。

関連項目

- 動的データベース・サービスによるワークロード管理

対話モードでのDBCAによるターゲット・ノードへのデータベース・インスタンスの追加

対話モードでDBCAを使用してターゲット・ノードにデータベース・インスタンスを追加するには、次のステップを実行します。

1. 既存ノードの$ORACLE_HOME環境変数がOracle RACホームに設定されていることを確認します。
2. Oracle_home/binディレクトリから、システム・プロンプトでdbcaを入力してDBCAを起動します。

   DBCAの実行中に、CVUの特定のチェックが実行されます。ただし、コマンドラインからCVUを実行して、様々な検証を実行することもできます。

   DBCAにOracle RAC用の「ようこそ」ページが表示されます。DBCAの各ページで「ヘルプ」をクリックすると、追加情報を参照できます。
3. 「インスタンス管理」を選択して「次へ」をクリックすると、DBCAによって「インスタンス管理」ページが表示されます。
4. 「インスタンスの追加」を選択し、「次へ」をクリックします。DBCAによって「クラスタ・データベースのリスト」ページが表示され、データベースおよび現在のステータス(ACTIVEやINACTIVEなど)が表示されます。
5. 「クラスタ・データベースのリスト」ページで、インスタンスを追加するアクティブなOracle RACデータベースを選択します。「次へ」をクリックすると、DBCAによって選択したOracle RACデータベースの既存のインスタンスの名前を示す「クラスタ・データベース・インスタンスのリスト」ページが表示されます。
6. 新規インスタンスを追加するには、「次へ」をクリックします。DBCAによって「インスタンスの追加」ページが表示されます。
7. 「インスタンスの追加」ページで、DBCAに表示されるインスタンス名が既存のインスタンス名スキームと合致しない場合には、このページの一番上のフィールドにインスタンス名を入力します。
8. 「サマリー」ダイアログ・ボックスに表示された情報を確認し、「OK」をクリックするか、またはインスタンス追加操作を終了する場合は「取消」をクリックします。DBCAがインスタンス追加操作を実行中であることを示す進捗ダイアログ・ボックス

が表示されます。

9. DBCAセッションの終了後、次のコマンドを実行して、ターゲット・ノードでの管理権限を確認し、それらの権限に関する詳細情報を取得します(node_listは、データベース・インスタンスを追加したノードの名前で構成されます)。

```
cluvfy comp admprv -o db_config -d Oracle_home -n node_list [-verbose]
```

10. 「動的データベース・サービスによるワークロード管理」で説明したように、必要なサービス構成手順を実行します。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- 動的データベース・サービスによるワークロード管理

サイレント・モードでのDBCAによるターゲット・ノードへのデータベース・インスタンスの追加

DBCAのサイレント・モードを使用して、Oracle ClusterwareホームおよびOracle Databaseホームを拡張したノードにインスタンスを追加できます。

dbcaコマンドを実行する前に、既存ノードでORACLE_HOME環境変数が正しく設定されていることを確認します。次の構文を使用して変数に値を指定し、DBCAを実行します。

```
dbca -silent -addInstance -nodeName node_name -gdbName gdb_name
  [-instanceName instance_name -sysDBAUserName sysdba -sysDBAPassword
  password]
```

次の表に、各変数に指定する必要がある値を示します。

表11-1 DBCAサイレント・モード構文の変数

変数	説明
node_name	インスタンスの追加(または削除)対象のノード。
gdb_name	グローバル・データベース名。
instance_name	インスタンスの名前。Oracle RAC インスタンス名に関する Oracle ネーミング規則を使用せずにインスタンス名を指定する場合にのみ、インスタンス名を指定します。
sysdba	SYSDBA 権限を持つ Oracle ユーザーの名前。
password	SYSDBA ユーザーのパスワード。

「動的データベース・サービスによるワークロード管理」で説明したように、必要なサービス構成手順を実行します。

関連項目

- 動的データベース・サービスによるワークロード管理

クラスタ・ノードからのOracle RACの削除

クラスタ・ノードからOracle RACを削除するには、データベース・インスタンスおよびOracle RACソフトウェアを削除した後で、クラスタからそのノードを削除する必要があります。

ノート:

削除するノードにデータベース・インスタンスがない場合は、「Oracle RAC の削除」に進みます。

この項では、次の手順を実行して、Oracle RAC環境のクラスタからノードを削除します。

- Oracle RACデータベースからのインスタンスの削除
- Oracle RACの削除
- クラスタからのノードの削除

関連項目

- Oracle RACの削除

Oracle RACデータベースからのインスタンスの削除

データベース・インスタンスの削除手順は、ポリシー管理データベースと管理者管理データベースとで異なります。

ポリシー管理データベース・インスタンスを削除すると、データベース・インスタンスが存在するサーバー・プールのサーバーの数が減少します。管理者管理データベース・インスタンスの削除では、DBCAを使用したデータベース・インスタンスの削除が必要です。

ポリシー管理データベースの削除

ポリシー管理データベースを削除するには、データベース・インスタンスが存在するサーバーを別のサーバー・プールに再配置することで、データベース・インスタンスが存在するサーバー・プールのサーバーの数を減らします。この操作によって、ノードからOracle RACソフトウェアを削除しなくても、またはクラスタからノードを削除しなくても、事実上インスタンスは削除されます。

たとえば、クラスタ内の任意のノードで次のコマンドを実行すると、ポリシー管理データベースを削除できます。

```
$ srvctl stop instance -db db_unique_name -node node_name
$ srvctl relocate server -servers "server_name_list" -serverpool Free
```

最初のコマンドは特定のノードのデータベース・インスタンスを停止し、2つ目のコマンドはノードを現行のサーバー・プールから空きサーバー・プールに移動します。

管理者管理データベースからのインスタンスの削除

ノート:

Oracle RAC データベースからインスタンスを削除する前に、SRVCTL を使用して次の操作を実行します。

- サービスが構成されている場合は、サービスを再配置します。
- 各サービスを残りのインスタンスの 1 つで実行できるように、サービスを変更します。
- 管理者管理データベースから削除されるインスタンスが、何らかのサービスの優先インスタンスまたは使用可能インスタンスではないことを確認します。

関連項目

- Oracle RACの削除
- SRVCTLを使用したサービスの管理
- 対話モードでのDBCAによるノードからのインスタンスの削除

対話モードでのDBCAによるノードからのインスタンスの削除

この項の手順では、対話モードでDBCAを使用してOracle RACデータベースからインスタンスを削除する方法について説明します。

対話モードでDBCAを使用してインスタンスを削除するには、次のステップを実行します。

1. DBCAを起動します。

   削除するインスタンスを保持しているノード以外のノードで、DBCAを起動します。削除するデータベースおよびインスタンスは、このステップの間、実行し続けている必要があります。

2. DBCAの「操作」ページで「インスタンス管理」を選択し、「次へ」をクリックします。DBCAによって「インスタンス管理」ページが表示されます。

3. DBCAの「インスタンス管理」ページで、削除するインスタンスを選択し、「インスタンスの削除」を選択し、「次へ」をクリックします。

4. 「クラスタ・データベースのリスト」ページで、次のように、インスタンスを削除するOracle RACデータベースを選択します。

   a. この「クラスタ・データベース・インスタンスのリスト」ページには、DBCAによって選択したOracle RACデータベースに関連付けられたインスタンスと各インスタンスのステータスが表示されます。インスタンスを削除するクラスタ・データベースを選択します。

   b. 「確認」ダイアログ・ボックスで「OK」をクリックし、インスタンスの削除を続行します。

      DBCAがインスタンスを削除していることを示す進捗ダイアログ・ボックスが表示されます。この操作の中で、DBCAはインスタンスとそのインスタンスのOracle Net構成を削除します。

      DBCAを終了する場合は「いいえ」を、別の操作を実行する場合は「はい」をクリックします。「はい」をクリックすると、DBCAによって「操作」ページが表示されます。

5. 既存ノードでSQL*Plusを使用してGV$LOGビューを問い合せ、削除したインスタンスのREDOスレッドが削除されていることを確認します。REDOスレッドが無効になっていない場合は、スレッドを無効にします。たとえば:

   ```
   SQL> ALTER DATABASE DISABLE THREAD 2;
   ```

6. 次のコマンドを実行して、OCRからインスタンスが削除されていることを確認します(db_unique_nameはOracle RACデータベースの一意のデータベース名です)。

   ```
   $ srvctl config database -db db_unique_name
   ```

7. 複数のノードを削除する場合は、このステップを繰り返し、削除するすべてのノードからインスタンスを削除します。

サイレント・モードでのDBCAによるノードからのインスタンスの削除

DBCAのサイレント・モードを使用して、ノードからデータベース・インスタンスを削除できます。

次のコマンドを実行します(変数は、表11-1に示したインスタンスを追加するDBCAコマンドの変数と同じです)。次の例に示すとおり、DBCAが実行されているノード以外からインスタンスを削除する場合のみ、ノード名を指定します(passwordはパスワードです)。

```
dbca -silent -deleteInstance [-nodeList node_name] -gdbName gdb_name
-instanceName instance_name [-sysDBAUserName sysdba -sysDBAPassword password]
```

この時点で、次の作業が完了しました。

- 関連付けられたOracle Net Servicesリスナーからの選択されたインスタンスの登録解除
- インスタンスの構成ノードからの選択されたデータベース・インスタンスの削除
- Oracle Net構成の削除
- インスタンスの構成ノードからのOracle Flexible Architectureディレクトリ構造の削除

Oracle RACの削除

この手順では、クラスタから削除するノードからOracle RACソフトウェアを削除し、残りのノードのインベントリを更新します。

1. 削除するノードのOracle RACホームにリスナーが存在する場合、Oracle RACソフトウェアを削除する前にリスナーを無効にして停止する必要があります。リスナーの名前および削除するノード名前を指定して、次のコマンドをクラスタ内の任意のノードで実行します。

   ```
   $ srvctl disable listener -l listener_name -n name_of_node_to_delete
   $ srvctl stop listener -l listener_name -n name_of_node_to_delete
   ```

2. Oracle_home¥deinstallディレクトリから次のコマンドを実行することによって、削除するノードからOracleホームをアンインストールします(Oracleホームが共有されていない場合のみ)。

   ```
   deinstall -local
   ```

警告:

Oracle ホームが共有されている場合、共有ソフトウェアを削除してしまうため、このコマンドは実行できません。そのかわり、次のステップに進みます。

クラスタからのノードの削除

データベース・インスタンスおよびOracle RACソフトウェアを削除した後で、クラスタからそのノードを削除するプロセスを開始できます。削除するノードでスクリプトを実行してOracle Clusterwareインストールを削除し、残りのノードでスクリプトを実行してノード・リストを更新し、このプロセスを完了します。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

12 WindowsシステムのノードでのOracle RACの追加と削除

この章では、既存のOracle Real Application Clusters(Oracle RAC)ホームを他のノードおよびクラスタ内のインスタンスに拡張し、Oracle RACをノードおよびクラスタ内のインスタンスから削除する方法について説明します。Windowsシステムの場合の手順について説明します。

ノート:

この章では、Grid_home のエントリは Oracle Grid Infrastructure ホームのフル・パス名を指し、Oracle_home のエントリは Oracle RAC がある Oracle ホームの環境変数の置き換えを指します。

既存のOracle RACホームをクローニングし、クラスタ全体で複数の新規Oracle RACインストールを作成する場合は、「新規クラスタのノードへのOracle RACのクローニング」に示すクローニング手順に従います。

この章の内容は次のとおりです。

- Oracle Clusterwareがインストールされたノードへの Oracle RACの追加
- クラスタ・ノードからのOracle RACの削除

ノート:

- ocrconfig -showbackup コマンドを実行して Oracle RAC の追加または削除を行う前に、Oracle Cluster Registry(OCR)の現行のバックアップがあることを確認してください。
- ノードの追加とノードの削除のすべての手順で、%TEMP%、C:¥Temp などの一時ディレクトリは、共有ディレクトリにしないでください。一時ディレクトリが共有されていると、%TEMP%などの一時環境変数にはローカル・ノードにある場所が設定されます。また、すべてのノードに存在するディレクトリ・パスを使用してください。

関連項目

- 新規クラスタのノードへのOracle RACのクローニング

Oracle Clusterwareがインストールされたノードへの Oracle RACの追加

この手順を開始する前に、既存ノードのGrid_homeへのパスが正しいこと、およびOracle_home環境変数が正しく設定されていることを確認します。

Oracle Clusterwareがすでにインストールされているノードに Oracle RACデータベース・インスタンスを追加するには、クラスタの既存のノード(この手順ではnode1)にあるOracle RACホームをターゲット・ノードに拡張する必要があります。

1. node1のOracle_home¥addnodeディレクトリに移動し、次の構文を使用してaddnode.batスクリプトを実行します(ここで、node2は追加するノードの名前です)。

```
addnode.bat "CLUSTER_NEW_NODES={node2}"
```

このコマンドをサイレント・モードで実行するには:

```
addNode.bat -silent "CLUSTER_NEW_NODES={node2}"
```

使用するOracleホーム・ディレクトリで、Oracle Databaseソフトウェアをインストールしたときに Oracleホーム・ユーザーを指定した場合、OUIではOracleホーム・ユーザーのパスワードが必要になります。OUIは、ユーザーの(OCRに格納されている)ウォレットをチェックし、そこからパスワードを抽出します。ユーザー情報がウォレットに含まれていない場合は、コマンドラインで-promptPasswdフラグを指定しないかぎり、addnode.batスクリプトによってエラーが生成されます。

2. ポリシー管理データベースをOracle Automatic Storage Management(Oracle ASM)に格納する場合は、Oracle Managed Filesが有効になり、node2のサーバー・プールに領域があれば、crsdがOracle RACデータベース・インスタンスをnode2に追加するため、それ以上の処置は必要ありません。Oracle Managed Filesが有効でない場合は、UNDOログおよびREDOログを手動で追加する必要があります。

   サーバー・プールに空き領域がない場合は、node2は空きサーバー・プールに移動します。srvctl modify srvpoolコマンドを使用して、node2を格納するサーバー・プールのカーディナリティを上げた後に、node2は空きサーバー・プールから修正済サーバー・プールに移動し、crsdはOracle RACデータベース・インスタンスをnode2に追加します。

3. 管理者管理データベースを使用している場合、node2に新しいインスタンスを追加します

Oracle Automatic Storage Management Cluster File System (Oracle ACFS)を使用して共有されている共有Oracleホームを持つ場合は、次の手順を実行し、Oracle Databaseホームをnode2に拡張します。

1. rootとしてGrid_home¥binディレクトリから次のコマンドを実行して、新しいノードでOracle ACFSリソースを起動します。

```
$ srvctl start filesystem -device volume_device_name [-node node_name]
```

ノート:

Oracle ホームが格納されている Oracle ACFS レジストリ・リソースおよび Oracle ACFS ファイル・システム・リソースを含む Oracle ACFS リソースが、新しく追加されたノードでオンラインであることを確認します。

2. Oracle RACをインストールしたユーザーとして、追加したノードのOracle_home¥oui¥binディレクトリから次のコマンドを実行し、Oracle RACデータベース・ホームを追加します。

```
setup.exe -attachHome ORACLE_HOME="ORACLE_HOME" LOCAL_NODE="node2"
  ORACLE_HOME_NAME="home_name" -cfs
```

3. node1のOracle_home¥addnodeディレクトリに移動し、次の構文を使用してOracle RACをインストールしたユーザーとしてaddnode.batスクリプトを実行します。

```
addnode.bat -noCopy "CLUSTER_NEW_NODES={node2}"
```

ノート:

宛先ノードの Oracle ホームはすでにソフトウェアに完全に移入されているため、-noCopy オプションを使用します。

Oracle ACFS以外の共有ファイル・システムで共有Oracleホームを持つ場合は、まずターゲット・ノードでOracle RACデータベース・ホームのマウント・ポイントを作成し、Oracle RACデータベース・ホームをマウントおよびアタッチし、次のようにOracle Inventoryを更新します。

1. クラスタ内の既存ノードでsrvctl config database -db db_nameコマンドを実行して、マウント・ポイント情報を取得します。
2. Oracle RACデータベース・ホームをホストするファイル・システムをマウントします。
3. Oracle RACをインストールしたユーザーとして、追加したノードのOracle_home¥oui¥binディレクトリから次のコマンドを実行し、Oracle RACデータベース・ホームを追加します。

```
setup.exe -attachHome ORACLE_HOME="ORACLE_HOME" "CLUSTER_NODES=
  local_node_name}" LOCAL_NODE="node_name" ORACLE_HOME_NAME="home_name"
```

4. Oracle RACをインストールしたユーザーとして、次のようにOracle Inventoryを更新します。

```
setup.exe -updateNodeList ORACLE_HOME=mount_point_path "CLUSTER_NODES={node_list}"
```

前述のコマンドで、node_listはOracle RACデータベース・ホームがインストールされたすべてのノード(追加したものも含む)のリストです。

ノート:

ノード追加プロセスの終了後は、投票ディスクおよび Oracle Cluster Registry(OCR)ファイルをバックアップすることをお薦めします。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- ターゲット・ノードへの管理者管理Oracle RACデータベース・インスタンスの追加

ターゲット・ノードへの管理者管理Oracle RACデータベース・インスタンスの追加

Oracle Enterprise ManagerまたはDBCAを使用して、Oracle RACデータベース・インスタンスをターゲット・ノードに追加できます。

この項では、DBCAを使用してOracle RACデータベース・インスタンスを追加する方法について説明します。

これらのツールは、次のタスクをガイドします。

- 各ターゲット・ノードでの新規データベース・インスタンスの作成
- 高可用性コンポーネントの作成および構成
- Oracleホームからのデフォルト以外のリスナー用のOracle Net構成の作成
- 新規インスタンスの起動

- サービス構成ページでサービス情報を入力した場合、サービスの作成および起動

ターゲット・ノードにインスタンスを追加した後で、必要なサービス構成手順を実行する必要があります。

関連項目

- 動的データベース・サービスによるワークロード管理

対話モードでのDBCAによるターゲット・ノードへのデータベース・インスタンスの追加

対話モードでDBCAを使用してターゲット・ノードにデータベース・インスタンスを追加するには、次のステップを実行します。

1. 既存ノードのOracleホーム環境変数が正しく設定されていることを確認します。
2. 既存ノードのOracle_home¥binディレクトリから、システム・プロンプトでdbcaを入力してDBCAを起動します。

   DBCAの実行中に、CVUの特定のチェックが実行されます。ただし、コマンドラインからCVUを実行して、様々な検証を実行することもできます。
3. 「データベース操作」ページで、「インスタンス管理」を選択し、「次へ」をクリックすると、DBCAには「インスタンス管理」ページが表示されます。
4. 「インスタンスの追加」を選択し、「次へ」をクリックします。DBCAによって「クラスタ・データベースのリスト」ページが表示され、データベースおよび現在のステータス(ACTIVEやINACTIVEなど)が表示されます。
5. 「クラスタ・データベースのリスト」ページで、インスタンスを追加するアクティブなOracle RACデータベースを選択します。「次へ」をクリックすると、DBCAによって選択したOracle RACデータベースの既存のインスタンスの名前を示す「クラスタ・データベース・インスタンスのリスト」ページが表示されます。
6. 新規インスタンスを追加するには、「次へ」をクリックします。DBCAによって「インスタンスの追加」ページが表示されます。
7. 「インスタンスの追加」ページで、DBCAに表示されるインスタンス名が既存のインスタンス名スキームと合致しない場合には、このページの一番上のフィールドにインスタンス名を入力します。次に、リストから新しいノード名を選択します。

   ノート:

   「Oracle ホーム・ユーザー」オプションとともに Oracle ホームをインストールした場合、DBCA によって、このページでパスワードが求められます。
8. 「サマリー」ページの情報を確認し、「終了」をクリックしてインスタンス追加操作を開始します。DBCAがインスタンス追加操作を実行中であることを示す進捗ダイアログ・ボックスが表示されます。

OraMTS Service for Microsoft Transaction Serverの作成

Oracle Services for Microsoft Transaction Server (OraMTS)を使用すると、Microsoftアプリケーションで調整されるトランザクション内で、リソース・マネージャとしてOracle Databaseを使用できます。OraMTSは、Microsoft分散トランザクション・コーディネータ(MSDTC)に対するOracle Databaseのプロキシとして機能します。この結果、OraMTSによってクライアント側の接続プールが提供され、Oracleを利用するクライアント・コンポーネントを昇格可能なトランザクションおよび分散トランザクションに使用できるようになります。また、サービス自体がWindowsで実行される場合、OraMTSは、任意のオペレーティング・システム上で実行されているOracle Databaseと連携して動作できます。

Oracle Database 12cより前のリリースでは、OraMTSはソフトウェアのみのインストールの一部として作成されました。

Oracle Database 12c以上では、構成ツールを使用してこのサービスを作成する必要があります。

ノードの追加後、またはOracle RACのソフトウェアのみのインストールを実行した後でOraMTSサービスを作成するには、次の手順を実行します。

1. コマンド・ウィンドウを開きます。
2. ディレクトリを%ORACLE_HOME%¥binに変更します。
3. OraMTSCtlユーティリティを実行してOraMTSサービスを作成します(ここで、host_nameはサービスが作成されるノードのリストです)。

```
C:¥..bin> oramtsctl.exe -new -host host_name
```

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- Microsoft Windows用Oracle Services for Microsoft Transaction Server開発者ガイド

サイレント・モードでのDBCAによるターゲット・ノードへのデータベース・インスタンスの追加

Oracle ClusterwareホームおよびOracle Databaseホームを拡張したノードにインスタンスを追加します。

次の構文とともにDBCAのサイレント・モードを使用します。

```
dbca -silent -addInstance -nodeName node_name -gdbName gdb_name
[-instanceName instance_name -sysDBAUserName sysdba -sysDBAPassword password]
```

必要なサービス構成手順を実行します。

関連項目

- サービス管理ポリシー
- 動的データベース・サービスによるワークロード管理

クラスタ・ノードからのOracle RACの削除

クラスタ・ノードからOracle RACを削除するには、データベース・インスタンスおよびOracle RACソフトウェアを削除した後で、クラスタからそのノードを削除する必要があります。

ノート:

削除するノードにデータベース・インスタンスがない場合は、Oracle RAC を削除します。

この項では、次の手順を実行して、Oracle RAC環境のクラスタからノードを削除します。

- Oracle RACデータベースからのインスタンスの削除
- Oracle RACの削除
- クラスタからのノードの削除

関連項目

- Oracle RACの削除

Oracle RACデータベースからのインスタンスの削除

インスタンスの削除手順は、ポリシー管理データベースと管理者管理データベースとでは異なります。

ポリシー管理データベース・インスタンスを削除すると、データベース・インスタンスが存在するサーバー・プールのサイズが減少します。管理者管理データベース・インスタンスの削除では、DBCAを使用したデータベース・インスタンスの削除が必要です。

ポリシー管理データベースの削除

ポリシー管理データベースを削除するには、データベース・インスタンスが存在するサーバー・プールのサイズを小さくします。この操作によって、ノードからOracle RACソフトウェアを削除しなくても、またはクラスタからノードを削除しなくても、事実上インスタンスは削除されます。

たとえば、クラスタ内の任意のノードで次のコマンドを実行すると、ポリシー管理データベースを削除できます。

```
$ srvctl stop instance -db db_unique_name -node node_name
$ srvctl relocate server -servers "server_name_list" -serverpool Free
```

最初のコマンドは特定のノードのインスタンスで停止し、2つ目のコマンドはサーバーのリストを現行のサーバー・プールから空きサーバー・プールに移動します。

管理者管理データベースからのインスタンスの削除

ノート:

Oracle RAC データベースからインスタンスを削除する前に、SRVCTL を使用して次の操作を実行します。

- サービスが構成されている場合は、サービスを再配置します。
- 各サービスを残りのインスタンスの 1 つで実行できるように、サービスを変更します。
- 管理者管理データベースから削除されるインスタンスが、何らかのサービスの優先インスタンスまたは使用可能インスタンスではないことを確認します。

関連項目

- Oracle RACの削除
- SRVCTLを使用したサービスの管理

対話モードでのDBCAによるノードからのインスタンスの削除

対話モードでDBCAを使用してインスタンスを削除するには、次のステップを実行します。

1. OCRの現行のバックアップがあることを確認します。

   ocrconfig -showbackupコマンドを実行して、有効なバックアップがあることを確認します。

2. DBCAを起動します。

削除するインスタンスを保持しているノード以外のノードで、DBCAを起動します。削除するデータベースおよびインスタンスは、このステップの間、起動し、実行し続ける必要があります。

3. DBCAの「操作」ページで「インスタンス管理」を選択して、「次へ」をクリックします。「インスタンス管理」ページが表示されます。
4. 「インスタンス管理」ページで、「インスタンスの削除」を選択して、「次へ」をクリックします。DBCAによって「クラスタ・データベースのリスト」ページが表示されます。
5. インスタンスを削除するOracle RACデータベースを選択します。「次へ」をクリックすると、DBCAによって「クラスタ・データベース・インスタンスのリスト」ページが表示されます。「クラスタ・データベース・インスタンスのリスト」ページには、選択したOracle RACデータベースに関連付けられたインスタンスと各インスタンスのステータスが表示されます。
6. 「クラスタ・データベースのリスト」ページで、次のように、インスタンスを削除するOracle RACデータベースを選択します。
    a. この「クラスタ・データベース・インスタンスのリスト」ページには、DBCAによって選択したOracle RACデータベースに関連付けられたインスタンスと各インスタンスのステータスが表示されます。インスタンスを削除するクラスタ・データベースを選択します。「終了」をクリックします。
    b. 「確認」ダイアログ・ボックスで「OK」をクリックし、インスタンスの削除を続行します。
    c. 次の「確認」ダイアログ・ボックスで「OK」をクリックし、インスタンスおよび関連するOptimal Flexible Architecture(OFA)ディレクトリ構造を削除します。

        DBCAがインスタンスを削除していることを示す進捗ダイアログ・ボックスが表示されます。この操作の中で、DBCAはインスタンスとそのインスタンスのOracle Net構成を削除します。

        DBCAを終了する場合は「いいえ」を、別の操作を実行する場合は「はい」をクリックします。「はい」をクリックすると、DBCAによって「操作」ページが表示されます。
7. 既存インスタンスからSQL*Plusを使用してV$LOGビューを問い合せ、削除したインスタンスのREDOスレッドが削除されていることを確認します。REDOスレッドが無効になっていない場合は、スレッドを無効にします。たとえば:

```
SQL> ALTER DATABASE DISABLE THREAD 2;
```

8. 次のコマンドを実行して、OCRからインスタンスが削除されていることを確認します(db_unique_nameはデータベース名です)。

```
srvctl config database -db db_unique_name
```

9. 複数のノードを削除する場合は、このステップを繰り返し、削除するすべてのノードからインスタンスを削除します。

サイレント・モードでのDBCAによるノードからのインスタンスの削除

DBCAのサイレント・モードを使用して、ノードからデータベース・インスタンスを削除できます。

インスタンスを削除するには、次のコマンド構文を使用します。次の例に示すとおり、DBCAが実行されているノード以外からインスタンスを削除する場合のみ、ノード名を指定します(passwordはSYSDBAパスワードです)。

```
dbca -silent -deleteInstance [-nodeName node_name] -gdbName gdb_name
-instanceName instance_name [-sysDBAUserName sysdba] [-sysDBAPassword password]
```

次の表に、各変数に指定する必要がある値を示します。

表12-1 DBCAサイレント・モード構文の変数

変数	説明
node_name	インスタンスの追加(または削除)対象のノード。
gdb_name	グローバル・データベース名。
instance_name	インスタンスの名前。Oracle RAC インスタンス名に関する Oracle ネーミング規則を使用せずにインスタンス名を指定する場合にのみ、インスタンス名を指定します。
sysdba	SYSDBA 権限を持つ Oracle ユーザーの名前。
password	SYSDBA ユーザーのパスワード。

この時点で、次の作業が完了しました。

- 関連付けられたOracle Net Servicesリスナーからの選択されたインスタンスの登録解除
- インスタンスの構成ノードからの選択されたデータベース・インスタンスの削除
- Oracle Net構成の削除
- インスタンスの構成ノードからのOracle Flexible Architectureディレクトリ構造の削除

Oracle RACの削除

この手順では、クラスタから削除するノードからOracle RACソフトウェアを削除し、残りのノードのインベントリを更新します。

1. 削除するノードのOracle RACホームにリスナーが存在する場合、Oracle RACソフトウェアを削除する前にリスナーを無効にして停止する必要があります。リスナーの名前および削除するノード名前を指定して、次のコマンドをクラスタ内の任意のノードで実行します。

   ```
   C:¥srvctl disable listener -listener listener_name -node name_of_node_to_delete
   C:¥srvctl stop listener -listener listener_name -node name_of_node_to_delete
   ```

2. Oracle_home¥deinstallディレクトリから次のコマンドを実行して、削除するノードからOracleホームをアンインストールします。

   ```
   deinstall -local
   ```

共有Oracle RACホームがある場合、このステップのコマンド例に-cfsオプションを追加し、クラスタ・ファイル・システムの完全なパスの場所を指定します。

クラスタからのノードの削除

インスタンスを削除すると、クラスタからノードを削除するプロセスを開始できます。削除するノードでスクリプトを実行してOracle Clusterwareインストールを削除し、残りのノードでスクリプトを実行してノード・リストを更新し、このプロセスを完了します。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

13 設計およびデプロイメント方法

この章では、Oracle Real Application Clusters(Oracle RAC)環境でのデータベースの設計およびデプロイメント方法について簡単に説明します。また、高可用性を実現するための考慮事項を説明し、Oracle RACの様々なデプロイメントに関する一般的なガイドラインを示します。

この章の内容は次のとおりです。

- 高可用性を実現するOracle RACのデプロイ
- Oracle RACの設計に関する一般的な考慮事項
- Oracle RACでのデータベースの一般的なデプロイメント

高可用性を実現するOracle RACのデプロイ

多数のお客様がOracle RACを実装してそれぞれのOracle Databaseアプリケーションで高可用性を実現しています。真の高可用性を得るには、アプリケーションのインフラストラクチャ全体を高可用性にする必要があります。これには、インフラストラクチャ全体でシングル・ポイント障害の発生をなくすための詳細な計画が必要です。Oracle RACによってデータベースが高可用性になったとしても、重要なアプリケーションが使用不可になれば、ビジネスに悪影響が生じる可能性があります。たとえば、認証にLightweight Directory Access Protocol(LDAP)を使用する場合は、LDAPサーバーを高可用性にする必要があります。データベースが起動していても、LDAPサーバーへのアクセスに障害があってユーザーがデータベースに接続できないと、ユーザーにはシステム全体が停止しているように見えます。

この項には次のトピックが含まれます:

- 高可用性システムの設計について
- 高可用性環境でOracle RACをデプロイするためのベスト・プラクティス
- クラスタ内の単一または複数のデータベースでの複数のアプリケーションの統合
- Oracle RACのスケーラビリティ

高可用性システムの設計について

ミッション・クリティカルなシステムの場合は、フェイルオーバーとリカバリが実行可能であることに加えて、あらゆるタイプの障害に対して環境にリジリエンスがある必要があります。

ミッション・クリティカルなシステムの場合は、フェイルオーバーとリカバリが実行可能であることに加えて、あらゆるタイプの障害に対して環境にリジリエンスがある必要があります。これらの目標に到達するには、ビジネスのサービス・レベルの要件を定義することから始めます。この要件には、データ・センター内の障害(ノード障害など)または障害時リカバリ(データ・センター全体に障害が発生した場合)に対する最大のトランザクション応答時間およびリカバリ予測の定義が含まれる必要があります。通常、サービス・レベルの目標は、障害の内容にかかわらず、作業の目標応答時間になります。各冗長コンポーネントにリカバリ時間を設定します。アクティブ/アクティブ・モードで実行されている複数のハードウェア・コンポーネントがある場合でも、1つのコンポーネントに障害が発生したときに、そのコンポーネントの修理中に、他のハードウェア・コンポーネントが稼働状態を保持できるとは想定しないでください。また、コンポーネントがアクティブ・モードまたはパッシブ・モードで実行されている場合は、通常のテストを実行してフェ

イルオーバー時間を検証します。たとえば、ストレージ・チャネルのリカバリ時間は、何分もかかる場合があります。停止時間がビジネスのサービス・レベル契約の範囲内であることを確認し、この範囲を超えている場合は、ハードウェア・ベンダーと協力して構成および設定をチューニングします。

ミッション・クリティカルなシステムをデプロイする場合は、テストに機能試験、破壊試験および性能試験を含める必要があります。破壊試験では、システム内に様々な障害を発生させてリカバリをテストし、サービス・レベルの要件に適合しているかどうかを確認します。また、破壊試験によって、本番システム用の操作手順を作成できます。

ミッション・クリティカルなシステムまたは高可用性システムの設計と実装を支援するため、Oracleでは規模に関係なくすべての組織に適用可能な様々なソリューションを提供しています。小規模な作業グループとグローバル企業が同じように組織の重要なビジネス・アプリケーションの可用性を拡張できます。Oracleとインターネットを使用することで、現在では常にどこからでも確実にアプリケーションとそのデータにアクセスできます。Oracle Maximum Availability Architecture(MAA)は、Oracleの実証済高可用性テクノロジと推奨事項に基づいたOracleのベスト・プラクティスのブループリントです。MAAの目的は、最適な高可用性アーキテクチャの設計から複雑な仕組みを排除することです。

関連項目

- Oracle Maximum Availability Architecture(MAA)

高可用性環境でOracle RACをデプロイするためのベスト・プラクティス

アプリケーションは、Oracle Database、Oracle ClusterwareおよびOracle RACの多数の機能を使用してOracle RAC環境の障害を最小限に抑えたり、マスクすることができます。たとえば、次のことが可能です。

- VIPアドレスを使用してデータベースに接続することで、TCP/IPのタイムアウト待機時間をなくします。
- 詳細な操作手順を作成し、インフラストラクチャ内のすべてのコンポーネントに対して、定義されたサービス・レベルを満たす適切で有効なサポート契約があることを確認します。
- 接続時フェイルオーバー、高速接続フェイルオーバー、高速アプリケーション通知、ロード・バランシング・アドバイザなどのOracle RACの自動ワークロード管理機能を使用します。
- 個別のボリューム・グループに投票ディスクを配置して、I/Oスループットの低下による停止を減らします。x個の投票デバイスの障害に対応するには、2x + 1個のミラーを構成します。
- Oracle Database Quality of Service Management (Oracle Database QoS Management)を使用して、システムを監視し、パフォーマンスのボトルネックを検出します。
- 約2ミリ秒(ms)以下のI/Oサービス時間でOCRを配置します。
- FAST_START_MTTR_TARGET初期化パラメータを使用してデータベース・リカバリをチューニングします。
- Oracle Automatic Storage Management(Oracle ASM)を使用してデータベース記憶域を管理します。
- 厳密変更制御手順が有効であることを確認します。
- LDAP、NIS、DNSなどの周辺のインフラストラクチャに高可用性およびリジリエンスがあることを確認します。これらのエンティティは、Oracle RACデータベースの可用性に影響します。可能な場合は、ローカルのバックアップ手順を定期的に実行します。
- Oracle Enterprise Managerを使用して、Oracle RACデータベースのみでなく、Oracle RAC環境全体を管理

します。Oracle Enterprise Managerを使用して、サービスの作成と変更、およびクラスタ・データベース・インスタンスとクラスタ・データベースの起動と停止を実行できます。

- Recovery Manager (RMAN)を使用して、データ・ファイル、制御ファイル、サーバー・パラメータ・ファイル(SPFILE)およびアーカイブREDOログ・ファイルのバックアップ、リストアおよびリカバリを実行します。RMANをメディア・マネージャとともに使用すると、ファイルを外部記憶域にバックアップできます。また、Oracle RACデータベースのバックアップまたはリカバリの実行時にパラレル化を構成できます。Oracle RACでは、RMANのチャネルは、すべてのOracle RACインスタンス間で動的に割り当てられます。チャネルのフェイルオーバーによって、1つのノード上で失敗した操作を別のノードで継続できます。RMANは、Oracle Enterprise Manager Backup ManagerまたはコマンドラインからRMANを実行できます。
- 順序番号を使用している場合、常にNOORDERオプションを指定したCACHEを使用することによって、順序番号生成のパフォーマンスを最適化します。ただし、CACHEオプションを使用すると、連続しない順序番号が生成される場合があります。連続しない順序番号を使用できない環境では、NOCACHEオプションを使用するか、または順序番号の事前生成を検討してください。アプリケーションでは順序番号の順序付けが必要で、連続しない順序番号を使用できる場合は、CACHEおよびORDERを使用して、Oracle RACの順序番号のキャッシュおよび順序付けを行います。アプリケーションで連続した順序番号の順序付けが必要な場合は、NOCACHEおよびORDERを使用します。NOCACHEとORDERの組合せは、その他のキャッシュおよび順序付けの組合せと比較すると、パフォーマンスに最も大きな影響を及ぼします。

  ノート:

  連続しない順序番号を使用できない環境では、順序番号の事前生成を検討するか、または ORDER および CACHE オプションを使用してください。

  Oracle Database 18c以降では、大規模な順序キャッシュを構成するかわりに、スケーラブルな順序を使用して、データのロードのスケーラビリティを向上させることができます。特に順序値が表の主キー列の移入に使用される場合、スケーラブルな順序により同時データ・ロード操作のパフォーマンスが向上します。
- 索引を使用する場合は、逆キー索引などの方法を検討してパフォーマンスを最適化します。逆キー索引は、挿入日付に基づいた索引のように、索引の一方に頻繁に挿入を行う場合に特に有効です。

関連項目

- 動的データベース・サービスによるワークロード管理
- Recovery Managerの構成およびアーカイブ
- 順序をスケーラブルにする方法

クラスタ内の単一または複数のデータベースでの複数のアプリケーションの統合

多くのユーザーは、複数のアプリケーションの単一データベースへの統合および複数のデータベースの単一クラスタへの統合を望んでいます。Oracle ClusterwareおよびOracle RACは、両方のタイプの統合をサポートしています。

Oracle ASMによって管理された単一の記憶域のプールでクラスタを作成すると、インフラストラクチャで複数のデータベースを(シングル・インスタンス・データベースか、またはOracle RACデータベースかに関係なく)管理できます。

統合時の容量管理

Oracle RACデータベースの場合は、ワークロードの要件に基づいてインスタンスの数、および指定されたデータベースのインスタンスを実行するノードを調整できます。クラスタで管理されるサービスなどの機能を使用すると、単一データベースまたは複数の

データベース間で複数のワークロードを管理できます。

作業を追加する場合は、クラスタの容量を適切に管理することが重要です。クラスタを管理するプロセス(Oracle Clusterwareとデータベースの両方からのプロセスを含む)は、CPUのリソースを適時に取得できる必要があり、システム内で優先度が高く設定される必要があります。Oracle Database Quality of Service Management(Oracle Database QoS Management)は、パフォーマンス目標を満たすようにCPUリソースを動的に割り当てることで、クラスタまたはデータベース内での複数のアプリケーション統合を支援します。クラスタ構成ポリシーを使用して、クラスタ・レベルのリソースを管理することもできます。

関連項目

- Oracle Database Quality of Service Managementユーザーズ・ガイド

統合時のグローバル・キャッシュ・サービス・プロセスの管理

サーバー上のリアルタイム・グローバル・キャッシュ・サービス・プロセス(LMSn)の数は、プロセッサ数と同じか、それより少なくすることをお薦めします。(これは、コアを含めて認識されるCPUの数です。たとえば、デュアル・コアCPUは、2つのCPUとみなされます。)ノード上にインスタンスを追加するときは、システムで負荷試験を実行して、ワークロードをサポートするのに必要な容量があることを確認することが重要です。

多数の小規模なデータベースをクラスタに統合する場合は、Oracle RACインスタンスによって作成されるLMSnの数を減らす必要がある場合があります。デフォルトでは、サーバーで検出されるCPUの数に基づいてプロセスの数がOracle Databaseによって算出されます。この計算の結果、LMSnプロセスがOracle RACインスタンスで必要とされるプロセス数より多くなることがあります。1つのLMSプロセスは、最大4個のCPUで対応できます。LMSnプロセスの数を減らすには、GC_SERVER_PROCESSES初期化パラメータを手動で最小限度の1の値に設定します。CPU4個ごとに、アプリケーションで必要とされるプロセスを1つ追加します。通常は、ビジー状態のLMSnプロセスを少なくすることをお薦めします。Oracle Databaseはインスタンスの起動時にプロセスの数を算出するため、その値を変更するにはインスタンスを再起動する必要があります。

統合へのDatabase Cloudの使用

データベース・クラウドは、Global Data Servicesフレームワークによって、単一の仮想サーバーに統合された一連のデータベースであり、1つ以上のグローバル・サービスを提供すると同時に、高いパフォーマンス、可用性、およびリソースの使用率の最適化を実現します。

Global Data Servicesでは、これらの仮想化リソースを最小限の管理オーバーヘッドで管理し、追加のクライアント要求を処理するために迅速にデータベース・クラウドを拡張できます。クラウドを構成するデータベースは、グローバルに分散させることができ、クライアントは、サービス名を指定するのみでデータベース・クラウドに接続でき、クラウドのコンポーネントやトポロジを把握する必要はありません。

データベース・クラウドは、複数のデータベース・プールで構成できます。データベース・プールは、データベース・クラウド内の一連のデータベースであり、一意のグローバル・サービス・セットを提供し、特定の管理ドメインに属しています。クラウド・データベースを複数のプールにパーティション化すると、サービス管理が簡素化され、かつ、各プールを異なる管理者が管理できることによって、セキュリティが向上します。データベース・クラウドは、複数の地理的リージョンにまたがることができます。リージョンは、互いに近くに存在すると考えられるデータベース・クライアントおよびサーバーが含まれる論理的な境界です。通常、1つのリージョンが1つの

データ・センターに対応しますが、データ・センター間のネットワーク待機時間が、これらのデータ・センターにアクセスするアプリケーションの品質保証契約を満たす場合、複数のデータ・センターを同じリージョンに配置できます。

グローバル・サービスによって、ローカルまたはグローバルに分散している、疎結合の異機種データベースを、スケーラブルで可用性の高いプライベート・データベース・クラウドに統合できます。このデータベース・クラウドは、地球上のすべてのクライアントで共有できます。プライベート・データベース・クラウドを使用すると、使用可能なリソースの使用率が最適化され、データベース・サービスのプロビジョニングが簡素化されます。

関連項目

- Oracle Database Global Data Services概要および管理ガイド

Oracle RACのスケーラビリティ

Oracle RACでは、複数のシステムからデータの個別コピーへのトランザクションに一貫性のある同時アクセスが可能です。単一のサーバーの容量を超えたスケーラビリティも実現します。アプリケーションが対称マルチプロセッシング(SMP)サーバーで透過的にスケール変更する場合は、そのアプリケーションはアプリケーション・コードを変更しなくてもOracle RACで適切にスケール変更すると想定できます。

従来、データベース・サーバーが容量を超えて実行される場合は、より大きな新しいサーバーに置き換えられてきました。サーバーの容量が大きくなると、価格も上がります。ただし、Oracle RACデータベースには、容量を増やすための別の手段があります。

- 従来はより大きいSMPサーバーで実行していたアプリケーションを移行して、小規模なサーバーのクラスタで実行できます。
- 現行のハードウェアへの投資を維持し、新しいサーバーをクラスタに追加する(あるいは新しいクラスタを作成または追加する)ことで、容量を増やすことができます。

Oracle ClusterwareおよびOracle RACを使用してクラスタにサーバーを追加する場合、停止は必要ありません。新規インスタンスが起動されると同時に、アプリケーションは追加された容量を使用できます。

クラスタ内のすべてのサーバーは、同じオペレーティング・システムと同じバージョンのOracle Databaseを実行する必要がありますが、各サーバーの容量を揃える必要はありません。Oracle RACを使用すると、要件にあったクラスタ(デュアルCPUの汎用サーバーで構成されるクラスタや、32または64のCPUが組み込まれたサーバーで構成されるクラスタなど)を構築できます。Oracleのパラレル実行機能を使用すると、1つのSQL文を複数のプロセスに分割でき、それぞれのプロセスで作業のサブセットが実行されます。Oracle RAC環境では、パラレル・プロセスをユーザーが接続されているインスタンスでのみ実行するように定義したり、クラスタ内の複数のインスタンス間で実行するように定義することができます。

関連項目

- 新規クラスタのノードへのOracle RACのクローニング
- クローニングを使用した同じクラスタのノードへのOracle RACの拡張
- LinuxおよびUNIXシステムのノードでのOracle RACの追加と削除
- Windowsシステムのノードでの Oracle RACの追加と削除

Oracle RACの設計に関する一般的な考慮事項

この項では、Oracle RAC環境でのデータベースの設計およびデプロイメント方法について簡単に説明します。また、高可用性を実現するための考慮事項を説明し、Oracle RACの様々なデプロイメントに関する一般的なガイドラインを示します。

Oracle RACデータベース上にデプロイするアプリケーションの設計および開発時には、次のステップの実行を検討してください。

1. 設計とアプリケーションのチューニング
2. メモリーとI/Oのチューニング
3. 競合のチューニング
4. オペレーティング・システムのチューニング

ノート:

アプリケーションを SMP システムで拡張できない場合は、アプリケーションを Oracle RAC データベースに移動してもパフォーマンスは向上しません。

挿入集中型オンライン・トランザクション処理(OLTP)アプリケーションには、ハッシュ・パーティション化を使用することを検討します。ハッシュ・パーティション化の特長は次のとおりです。

- 単一データベース構造への同時挿入による競合を軽減します。
- 索引がローカルで表を使用してパーティション化され、表が順序ベースのキーでパーティション化されている場合、順序ベースの索引に適用されます。
- アプリケーションに対して透過的です。

OLTP環境に対して表および索引のハッシュ・パーティション化を使用すると、Oracle RACデータベースでのパフォーマンスが大幅に向上します。ハッシュ・パーティション化された索引では、索引レンジ・スキャンは使用できないことに注意してください。

Oracle RACでのデータベースの一般的なデプロイメント

この項では、Oracle RACデータベースをデプロイする際の考慮事項について説明します。ここで説明する方法を採用しない場合でも、Oracle RACデータベースのパフォーマンスが低下することはありません。効率的な非クラスタ設計であれば、アプリケーションはOracle RACデータベースで効率的に実行されます。

この項には次のトピックが含まれます:

- Oracle RACでの表領域の使用
- Oracle RACでのオブジェクトの作成およびパフォーマンス
- Oracle RACでのノードの追加と削除およびSYSAUX表領域
- 分散トランザクションおよびOracle RAC
- Oracle RACでのOLTPアプリケーションのデプロイ
- キャッシュ・フュージョンによる柔軟な実装

- Oracle RACでのデータ・ウェアハウス・アプリケーションのデプロイ
- Oracle RACでのデータ・セキュリティの考慮事項

Oracle RACでの表領域の使用

ローカル管理表領域の使用の他に、自動セグメント領域管理(ASSM)および自動UNDO管理を使用して領域管理をさらに簡単にすることができます。

ASSMでは、挿入を行うためのブロックの各インスタンスのサブセット間にインスタンスのワークロードが分散されます。これによってブロック転送が最小限に抑えられるため、Oracle RACパフォーマンスが向上します。自動UNDO管理をOracle RAC環境にデプロイするには、各インスタンスに固有のUNDO表領域が必要です。

Oracle RACでのオブジェクトの作成およびパフォーマンス

原則として、DDL文の使用はメンテナンス・タスクのみに限定し、システム操作のピーク時には実行しないようにします。ほとんどのシステムでは、新しいオブジェクトの生成量とその他のDDL文に対する制限が必要です。非クラスタのOracle Databaseの場合と同様に、オブジェクトの作成と削除が多くなるとパフォーマンスのオーバーヘッドが増加する可能性があります。

Oracle RACでのノードの追加と削除およびSYSAUX表領域

Oracle RACデータベース環境にノードを追加する場合は、SYSAUX表領域のサイズを大きくする必要があります。また、クラスタ・データベースのノードを削除する場合は、SYSAUX表領域のサイズを小さくできる場合もあります。

関連項目:

複数のインスタンスに対するSYSAUX表領域のサイズの設定方法については、プラットフォーム固有のOracle RACのインストレーション・ガイドを参照してください。

分散トランザクションおよびOracle RAC

Oracle RAC環境でXAトランザクションを実行しているときに、そのパフォーマンスが低い場合は、1つ以上のOracle RACインスタンスでの複数のOracle Distributed Transaction Processing (DTP)サービスの作成により、密結合分散トランザクションのすべてのブランチを同じインスタンスに割り当てます。

各DTPサービスは、1つのみのOracle RACインスタンスで使用可能な単一のサービスです。分散トランザクション処理のためのデータベース・サーバーへのアクセスは、すべてDTPサービスを経由する必要があります。単一のグローバル分散トランザクションのすべてのブランチが、同じDTPサービスを使用することを確認してください。つまり、TNS名やJDBC URLなどのネットワーク接続記述子には、分散トランザクション処理をサポートするためにDTPサービスを使用する必要があります。

関連項目

- Oracle RACの分散トランザクション処理
- Oracle Database開発ガイド

Oracle RACでのOLTPアプリケーションのデプロイ

Oracle RACデータベースはキャッシュ・フュージョンによって、オンライン・トランザクション処理(OLTP)アプリケーションに最適なデプロイメント・サーバーになります。これは、これらのアプリケーションには次の要件があるためです。

- 障害発生時の高可用性
- 増加するシステム需要に対応するためのスケーラビリティ
- 需要変動に応じたロード・バランシング

Oracle DatabaseおよびOracle RACの高可用性機能は、処理を中断することなく、障害が発生していないインスタンスにワークロードを再分散し、ロード・バランシングを実行できます。さらに、Oracle RACは、優れたスケーラビリティも提供するため、ノードを追加または置換した場合、Oracle Databaseは、リソースを再マスター化してロードの処理を再分散します。

キャッシュ・フュージョンによる柔軟な実装

オンライン・トランザクション処理システムの頻繁に変化するワークロードに対応するには、Oracle RACは、システム負荷やシステム可用性が変化しても、柔軟かつ動的に対応します。Oracle RACは、たとえば次のような原因で変動する幅広いサービス・レベルに対応します。

- ユーザーの需要の変化
- 取引の集中(大量取引の発生)など、ピーク時のスケーラビリティの問題
- システム・リソースの可用性の変化

Oracle RACでのデータ・ウェアハウス・アプリケーションのデプロイ

この項では、Oracle RAC環境でのデータ・ウェアハウス・システムのデプロイ方法について説明します。また、共有ディスク・アーキテクチャで使用可能なデータ・ウェアハウス機能についても説明します。

この項には次のトピックが含まれます:

- Oracle RACでのデータ・ウェアハウス・アプリケーションのスピードアップ
- データ・ウェアハウス・システムおよびOracle RACにおけるパラレル実行

Oracle RACでのデータ・ウェアハウス・アプリケーションのスピードアップ

Oracle RACは、Oracle Databaseの非クラスタのメリットを拡大するため、データ・ウェアハウス・アプリケーションには理想的です。Oracle RACは、Oracle RACデータベースに属するすべてのノード上で使用可能な処理能力を最大限に活用して、データ・ウェアハウス・システムをスピードアップおよびスケールアップすることによって、Oracleのシングル・インスタンスのメリットを拡大します。

クエリー・オプティマイザは、最適な実行計画の判断に、パラレル実行を検討します。クエリー・オプティマイザのデフォルトのコスト・モデルはCPU+I/Oで、コスト単位は時間です。Oracle RACでは、クエリー・オプティマイザが、プロセッサ数に基づいてパラレル化の適切なデフォルト値を動的に計算します。代替アクセス・パスのコスト評価(表スキャンと索引アクセスなど)では、操作に使用できる並列度が考慮されます。これによってOracle Databaseは、Oracle RAC構成用に最適化された実行計画を選択します。

データ・ウェアハウス・システムおよびOracle RACにおけるパラレル実行

パラレル実行は、複数のプロセスを使用して1つ以上のCPUでSQL文を実行し、また、非クラスタOracle Databaseと Oracle RACデータベースの両方で使用可能です。

Oracle RACは、パラレル処理をすべての利用可能なインスタンスに分散することによってパラレル実行を十分に活用します。パラレル操作に使用できるプロセスの数は、各表または索引に割り当てられる並列度によって異なります。

関連項目

- Oracle Databaseパフォーマンス・チューニング・ガイド
- Oracle Database概要

Oracle RACでのデータ・セキュリティの考慮事項

この項では、2つのOracle RACセキュリティの考慮事項について説明します。内容は次のとおりです。

- 透過的データ暗号化およびキーストア
- Windowsファイアウォールの考慮事項

透過的データ暗号化およびキーストア

Oracle Databaseでは、Oracle RACノードはキーストア(ウォレット)を共有できます。これにより、すべてのノードにわたってキーストアを手動でコピーし同期化する必要がなくなります。共有ファイル・システム上にキーストアを作成することをお薦めします。これにより、すべてのインスタンスが同じ共有キーストアにアクセスできます。

Oracle RACは、次の方法でキーストアを使用します。

1. 任意の1つのOracle RACインスタンスで実行されるキーストアのオープンやクローズなどの任意のキーストア操作は、他のすべてのOracle RACインスタンスに適用されます。つまり、1つのインスタンスでキーストアをオープンおよびクローズすると、すべてのOracle RACインスタンスでキーストアがオープンおよびクローズされます。
2. 共有ファイル・システムを使用する場合は、すべてのOracle RACインスタンスのENCRYPTION_WALLET_LOCATIONパラメータが、必ず同じ共有キーストアの場所を指すようにします。また、セキュリティ管理者は、適切なディレクトリ権限を割り当てることによって、共有キーストアのセキュリティを確保する必要もあります。

ノート:

オペレーティング・システムで Oracle Automatic Storage Management Cluster File System (Oracle ACFS)が使用可能な場合は、キーストアを Oracle ACFS に格納することをお薦めします。Oracle ASM に Oracle ACFS がない場合は、Oracle ASM コンフィギュレーション・アシスタント (ASMCA)を使用して作成してください。次のように、各インスタンスの sqlnet.ora ファイルに、マウント・ポイントを追加する必要があります。

```
ENCRYPTION_WALLET_LOCATION=
  (SOURCE = (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = /opt/oracle/acfsmounts/data_keystore)))
```

このファイル・システムは、インスタンスの起動時に自動的にマウントされます。キーストアのオープンとクローズ(および TDE マスター暗号化キーの設定またはキー更新およびローテーションを行うコマンド)は、すべてのノード間で同期化されています。

3. ある1つのインスタンスで実行されるmaster key rekeyは、すべてのインスタンスに適用されます。新しいOracle RACノードは、起動されると、現在のキーストアの状態(オープンまたはクローズ)を認識します。
4. マスター・キーを設定または変更している場合は、キーストアのADMINISTER KEY MANAGEMENT SET KEYSTORE OPENまたはCLOSE SQL文は発行しないでください。

Oracleでは、Oracle RACノードごとの個別のTDEウォレットの使用はサポートされていません。かわりに、Oracle RAC環境でTDE用の共有ウォレットを使用してください。これによってすべてのインスタンスが、同じ共有ソフトウェア・キーストアにアクセスできます。

関連項目

- Oracle Database Advanced Securityガイド

Windowsファイアウォールの考慮事項

デフォルトにより、Windows Server 2003 Service Pack 1以上のすべてのインストレーションにおいて、Windowsファイアウォールは、着信接続に対するほとんどすべてのTCPネットワーク・ポートをブロックできます。そのため、TCPポート上で着信接続をリスニングするOracle製品はすべて、これらのどの接続要求も受信せず、これらの接続を行っているクライアントはエラーを報告します。

インストールするOracle製品およびその使用方法によって、Windows Server 2003でファイアウォール製品が機能するように、Windowsのインストール後の追加の構成作業を実行する必要があります。

ウォレットを使用してONSクライアントを安全に実行

SSL証明書を構成および使用して、データベース層のONSサーバーと中間層の通知クライアント間の認証を設定できます。

JDBC接続プールまたはOracle Universal Connection Poolなどで使用する高速接続フェイルオーバーなどのOracle RAC機能は、Oracle RACノードで実行されているOracle Notification Service (ONS)の通知をサブスクライブします。これらの接続は通常認証されません。

1. Oracle Database 18c以降、Oracle Grid Infrastructureのインストールにデフォルトのウォレットが作成されます。
2. 中間層でクライアント側ONSデーモンを実行している場合は、次の2つの構成が可能です。
   - (OracleAS 10.1.3.xなどの) OPMNからONSを起動する場合。この構成にはopmn.xmlを使用します。
   - (ONSCTLの使用時など)スタンドアロンでONSを起動する場合。この構成にはons.configを使用します。

   最初の構成については、Oracle Application ServerリリースのOPMN管理者ガイドを参照してください。この構成では、ウォレットの場所を指定するために、opmn.xmlファイルを変更します。

   2つ目の構成については、クライアント側ONSデーモンが異なるサーバー上で実行される可能性があります。ステップ1のウォレットをクライアント側サーバーにコピーし、そのクライアント側サーバー上のパスをons.configファイルまたはopmn.xmlファイルに指定します。
3. クライアント側のONSデーモンなしでリモートONS構成を実行する場合は、クライアント側のサーバーを構成します。

- クライアント・クラスタにONSリソースをエクスポートします。
  次のようなコマンドを使用します。cluster_nameはリモート・クラスタの名前です。また、filenameは、資格証明のデータを書き込むファイルの名前です。

  ```
  $ srvctl export ons -clientcluster cluster_name -clientdata filename
  ```

- クライアント側サーバーでパスを指定します。
  ons.configファイルまたはopmn.xmlファイルを変更して、コピーしたファイルの場所を指すようにします。

関連項目

- クライアント側のONSデーモンの構成
- ONSのリモート構成

ハング・マネージャの概要

ハング・マネージャは、ハングを自律的に解決し、リソースの可用性を維持するOracle Real Application Clusters (Oracle RAC)環境の機能です。

ハング・マネージャはデフォルトで有効になっており、次の処理を実行します。

- データベースのハングおよびデッドロックの確実な検出
- データベースのハングおよびデッドロックの自律的な解決
- Oracle Database QoSのパフォーマンス・クラス、ランクおよびSLAを維持するためのポリシーのサポート
- すべての検出および解決のログ記録
- 感度(Normal/High)およびトレース・ファイルのサイズを構成するためのSQLインタフェースの提供

データベースは、セッションによって1つ以上のセッションのチェーンがブロックされた場合にハングします。ブロックしているセッションによってロックやラッチなどのリソースが保持され、ブロックされているセッションの進捗が妨げられます。セッションのチェーンには、チェーン内のその他すべてのセッションをブロックするルートまたは最終ブロッカ・セッションがあります。ハング・マネージャは、ハングを検出して解決することによって、これらの問題を自律的に解決します。

ハング・マネージャのアーキテクチャ

ハング・マネージャは、データベース内のDIA0タスクとして自律的に実行されます。

ハング・マネージャは、次の3つのフェーズで機能します。

- 検出: このフェーズでは、ハング・マネージャによってすべてのノードのデータが収集され、別のセッションで保持されているリソースを待機しているセッションが検出されます。
- 分析: このフェーズでは、ハング・マネージャによって検出フェーズで検出されたセッションが分析され、セッションが潜在的な遅延の一部であるかどうかが判別されます。セッションが遅延している可能性がある場合、ハング・マネージャは特定のしきい値期間の間待機して、セッションが遅延していることを確認します。
- 検証: このフェーズでは、しきい値期間の経過後に、ハング・マネージャによってセッションが遅延していることが検証され、遅延の原因となっているセッションが選択されます。

遅延の原因となっているセッションを選択すると、ハング・マネージャによってそのセッションに解決方法が適用されます。セッションのチェーンまたは遅延が自動的に解決された場合は、ハング・マネージャによって遅延の解決方法は適用されません。ただし、遅

延が自動的に解決されない場合、ハング・マネージャは遅延の原因となっているセッションを終了することで遅延を解決します。セッションの終了が失敗した場合は、ハング・マネージャによってセッションのプロセスが終了されます。このプロセス全体は自律型であり、リソースを長期間ブロックせず、パフォーマンスに影響を与えません。

たとえば、遅延したセッションのチェーンに高ランクのセッションが含まれている場合は、ハング・マネージャにより、遅延の原因となっているセッションの終了が迅速に実行されます。遅延の原因となっているセッションの終了によって、高ランクのセッションが長時間待機することが回避され、高ランクのセッションのパフォーマンス目標が維持されます。

ハング・マネージャのオプションの構成

感度を調整し、ハング・マネージャで使用されるログ・ファイルのサイズと数を制御できます。

感度

ハング・マネージャが遅延を検出した場合、セッションが遅延していることを確認するために、特定のしきい値期間の間待機します。しきい値期間を変更するには、DBMS_HANG_MANAGERを使用してsensitivityパラメータをNormalまたはHighに設定します。sensitivityパラメータがNormalに設定されている場合、ハング・マネージャはデフォルトの期間待機します。ただし、sensitivityがHighに設定されている場合は、期間が50%削減されます。

デフォルトでは、sensitivityパラメータはNormalに設定されています。ハング・マネージャの感度を設定するには、SQL*PlusでSYSユーザーとして次のコマンドを実行します。

- sensitivityパラメータをNormalに設定するには、次のようにします。

```
exec dbms_hang_manager.set(dbms_hang_manager.sensitivity,
dbms_hang_manager.sensitivity_normal);
```

- sensitivityパラメータをHighに設定するには、次のようにします。

```
exec dbms_hang_manager.set(dbms_hang_manager.sensitivity, dbms_hang_manager.sensitivity_high);
```

トレース・ログ・ファイルのサイズ

ハング・マネージャは、ファイル名に_base_を含むトレース・ファイルに、遅延の詳細な診断を記録します。base_file_size_limitパラメータを使用して、トレース・ファイルのサイズ(バイト単位)を変更します。SQL*Plusで次のコマンドを実行して、たとえば、トレース・ファイルのサイズを100 MBに設定します。

```
exec dbms_hang_manager.set(dbms_hang_manager.base_file_size_limit, 104857600);
```

トレース・ログ・ファイルの数

ハング・マネージャのベース・トレース・ファイルはトレース・ファイル・セットの一部です。base_file_set_countパラメータを使用して、トレース・ファイル・セット内のトレース・ファイルの数を変更します。SQL*Plusで次のコマンドを実行して、たとえば、トレース・ファイル・セット内のトレース・ファイルの数を6に設定します。

```
exec dbms_hang_manager.set(dbms_hang_manager.base_file_set_count,6);
```

デフォルトでは、base_file_set_countパラメータは5に設定されています。

ハング・マネージャの診断およびロギング

ハング・マネージャは自律的に遅延を解決し、解決内容をデータベースのアラート・ログに、診断をトレース・ファイルに継続的に記録します。

ハング・マネージャは、インシデント・コードORA-32701の自動診断リポジトリ(ADR)インシデントとして、解決をデータベースのアラート・ログに記録します。

また、遅延の検出に関する詳細な診断をトレース・ファイルで確認できます。トレース・ファイルとアラート・ログのファイル名はdatabase instance_dia0_で始まります。

- トレース・ファイルは$ ADR_BASE/diag/rdbms/database name/database instance/incident/incdir_xxxxxxディレクトリに格納されます。
- アラート・ログは$ ADR_BASE/diag/rdbms/database name/database instance/traceディレクトリに格納されます。

例13-1 ローカル・インスタンスのハング・マネージャ・トレース・ファイル

この例では、ローカル・データベース・インスタンスのハング・マネージャで表示される出力例を示します

```
Trace Log File .../oracle/log/diag/rdbms/hm1/hm11/incident/incdir_111/hm11_dia0_11111_i111.trc
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
...
*** 2016-07-16T12:39:02.715475-07:00
HM: Hang Statistics - only statistics with non-zero values are listed
            current number of active sessions 3
              current number of hung sessions 1
  instance health (in terms of hung sessions) 66.67%
       number of cluster-wide active sessions 9
         number of cluster-wide hung sessions 5
   cluster health (in terms of hung sessions) 44.45%
*** 2016-07-16T12:39:02.715681-07:00
Resolvable Hangs in the System
                     Root       Chain Total               Hang
   Hang Hang         Inst Root  #hung #hung  Hang   Hang  Resolution
     ID Type Status  Num  Sess   Sess  Sess  Conf   Span  Action
  ----- ---- -------- ---- ----- ----- ----- ------ ------ --------------------
      1 HANG RSLNPEND    3    44     3     5   HIGH GLOBAL Terminate Process
  Hang Resolution Reason: Although hangs of this root type are typically
    self-resolving, the previously ignored hang was automatically resolved.
```

例13-2 ハングしたセッションを示すアラート・ログ内のエラー・メッセージ

この例では、プライマリ・インスタンスに関するハング・マネージャのアラート・ログの例を示します

```
2016-07-16T12:39:02.616573-07:00
Errors in file .../oracle/log/diag/rdbms/hm1/hm1/trace/hm1_dia0_i1111.trc  (incident=1111):
ORA-32701: Possible hangs up to hang ID=1 detected
Incident details in: .../oracle/log/diag/rdbms/hm1/hm1/incident/incdir_1111/hm1_dia0_11111_i1111.trc
2016-07-16T12:39:02.674061-07:00
DIA0 requesting termination of session sid:44 with serial # 23456 (ospid:34569) on instance 3
     due to a GLOBAL, HIGH confidence hang with ID=1.
     Hang Resolution Reason: Although hangs of this root type are typically
    self-resolving, the previously ignored hang was automatically resolved.
DIA0: Examine the alert log on instance 3 for session termination status of hang with ID=1.
```

例13-3 ハング・マネージャによって解決されたセッションの遅延を示すアラート・ログ内のエラー・メッセージ

この例は、解決された遅延に関するローカル・インスタンスのハング・マネージャのアラート・ログの例を示します

```
2016-07-16T12:39:02.707822-07:00
Errors in file .../oracle/log/diag/rdbms/hm1/hm11/trace/hm11_dia0_11111.trc  (incident=169):
ORA-32701: Possible hangs up to hang ID=1 detected
Incident details in: .../oracle/log/diag/rdbms/hm1/hm11/incident/incdir_169/hm11_dia0_30676_i169.trc
2016-07-16T12:39:05.086593-07:00
DIA0 terminating blocker (ospid: 30872 sid: 44 ser#: 23456) of hang with ID = 1
     requested by master DIA0 process on instance 1
     Hang Resolution Reason: Although hangs of this root type are typically
    self-resolving, the previously ignored hang was automatically resolved.
     by terminating session sid:44 with serial # 23456 (ospid:34569)
...
DIA0 successfully terminated session sid:44 with serial # 23456 (ospid:34569) with status 0.
```

14 パフォーマンスの監視

この章では、Oracle Real Application Clusters(Oracle RAC)のパフォーマンスを監視およびチューニングする方法について説明します。

この章の内容は次のとおりです。

- Oracle RACデータベースの監視およびチューニングの概要
- Oracle RACのインターコネクト設定の検証
- インターコネクト処理への影響
- Oracle RACのパフォーマンス・ビュー
- CATCLUST.SQLによるOracle RACのデータ・ディクショナリ・ビューの作成
- Oracle RACのパフォーマンス統計
- Oracle RAC環境における自動ワークロード・リポジトリ
- Oracle RACのアクティブ・セッション履歴レポート
- Oracle RACの統計および待機イベントの監視

Oracle RACデータベースの監視およびチューニングの概要

この項には次のトピックが含まれます:

- Oracle RACおよびOracle Clusterwareの監視
- Oracle RACデータベースのチューニング
- データベース信頼性フレームワーク

関連項目

- Oracle Database 2日でデータベース管理者
- Oracle Database 2日でパフォーマンス・チューニング・ガイド
- Oracle Clusterware管理およびデプロイメント・ガイド

Oracle RACおよびOracle Clusterwareの監視

Oracle RACおよびOracle Clusterwareの監視には、Oracle Enterprise Managerを使用することをお薦めします。Oracle Enterprise Managerは、コンピューティング環境を監視および管理するためのOracleのWebベースの統合管理ソリューションです。Webブラウザを使用できる場所であれば、どこからでも、OracleのRACデータベース、アプリケーション・サーバー、ホスト・コンピュータおよびWebアプリケーションと、それらに関連するハードウェアやソフトウェアを管理できます。たとえば、Webブラウザが使用可能な場合、オフィス、自宅またはリモート環境からOracle RACデータベースのパフォーマンスを監視できます。

Oracle Enterprise Manager Cloud Controlはクラスタ対応で、クラスタ・データベースを管理するセントラル・コンソールを用意しています。クラスタ・データベースの「ホーム」ページから、次のすべての操作を実行できます。

- クラスタ内にあるノードの数や現在のステータスなど、全体的なシステム・ステータスの表示。この高レベルの表示機能を利用することで、包括的で集計的な情報のみを確認する場合に、個々のデータベース・インスタンスにアクセスして詳細を確認する必要がなくなります。
- すべてのインスタンスから集計されたアラート・メッセージと各アラート・メッセージのソースのリストの表示。アラート・メッセージとは、特定のメトリックの条件に一致したことを表すインジケータです。メトリックとは、システムの状態の報告に使用される測定の単位です。
- クラスタ全体に影響している問題および個々のインスタンスに影響している問題を確認します。
- クラスタ・キャッシュ一貫性の統計を監視して、処理の傾向の識別やOracle RAC環境のパフォーマンスの最適化に役立てます。キャッシュ一貫性の統計によって、複数のインスタンスのキャッシュ内にあるデータがどの程度適切に同期化されているかが測定されます。データ・キャッシュが相互に完全に同期化されている場合、どのインスタンスのキャッシュからメモリーの場所を読み取っても、その場所に対して任意のインスタンスのキャッシュから書き込まれた最新のデータが戻されます。

Oracle Enterprise Managerは、特定期間のデータ(収集ベース・データという)を累積します。また、Oracle Enterprise Managerは、現在のデータ(リアルタイム・データ)も提供します。

クラスタ・データベースの「ホーム」ページ

クライアント・ブラウザでOracle Enterprise Managerを使用して、Oracle ClusterwareおよびOracle RACの両方の環境を監視できます。監視には次のタスクが含まれます。

- VIPの再配置が行われた場合の通知
- クラスタ検証ユーティリティ(cluvfy)により取得した情報を使用する、クラスタの各ノードのOracle Clusterwareのステータス
- ノード・アプリケーション(nodeapps)が起動または停止した場合の通知
- Oracle ClusterwareのOCR用アラート・ログに記録された問題、投票ディスクの問題(ある場合)およびノード削除の通知

クラスタ・データベースの「ホーム」ページは、非クラスタ・データベースの「ホーム」ページと似ています。ただし、クラスタ・データベースの「ホーム」ページには、Oracle Enterprise Managerにより、システムの状態と可用性が表示されます。これには、アラート・メッセージおよびジョブ・アクティビティのサマリーと、すべてのデータベースおよびOracle Automatic Storage Management (Oracle ASM)インスタンスへのリンクも含まれます。たとえば、すべての優先インスタンスでサービスが実行されていない場合、またはサービスの応答時間のしきい値条件が満たされていない場合などに、クラスタでのサービスに関する問題を追跡できます。

「インターコネクト」ページ

Oracle Enterprise Managerの「インターコネクト」ページを使用して、Oracle Clusterware環境を監視できます。「インターコネクト」ページには、次のような、クラスタのパブリック・インタフェースおよびプライベート・インタフェースや、インターコネクトのデータベース・インスタンスによるロードが表示されます。

- プライベート・インターコネクトでの全体的なスループット
- 構成ミスのためデータベース・インスタンスがパブリック・インタフェースを使用している場合の通知
- インターコネクトのスループットおよびエラー(発生した場合)

- インスタンスごとのインターコネクトのスループット

これらの情報はすべて、履歴表示を含む収集としても使用することができ、このことは、クラスタの待機イベント関連の問題を診断する場合などに、クラスタ・キャッシュ一貫性と併用すると有効です。クラスタ・データベースの「ホーム」ページで「InterConnect」タブをクリックするか、またはOracle RACデータベースの「ホーム」ページで診断結果の「インターコネクト・アラート」リンクをクリックすると、「インターコネクト」ページにアクセスできます。

「クラスタ・データベース」の「パフォーマンス」ページ

Oracle Enterprise Managerクラスタ・データベースの「パフォーマンス」ページには、データベースのパフォーマンス統計のサマリーが表示されます。

統計は、グラフにあるクラスタ・データベース内のすべてのインスタンス間でロールアップされます。グラフの横のリンクを使用すると、より詳細な情報を取得したり、次のタスクを実行することができます。

- パフォーマンスの問題の原因の特定。
- リソースを追加または再分散する必要があるかどうかの判別。
- SQL計画およびスキーマのチューニングによる最適化。
- パフォーマンスの問題の解決

クラスタ・データベースの「パフォーマンス」ページには、次のグラフが含まれます。

- 「クラスタ・ホストのロード平均」グラフ: クラスタ・データベースの「パフォーマンス」ページの「クラスタ・ホストのロード平均」グラフには、データベース外部で発生する可能性がある問題が表示されます。このグラフには、クラスタ内で使用可能なノードについて、過去1時間のロードの最大値、平均値および最小値が表示されます。
- 「グローバル・キャッシュ・ブロックのアクセス待機時間」グラフ: 各クラスタ・データベース・インスタンスのシステム・グローバル領域(SGA)には、独自のバッファ・キャッシュが存在します。キャッシュ・フュージョンの使用によって、Oracle RAC環境で各インスタンスのバッファ・キャッシュが論理的に結合され、論理的に結合された単一のキャッシュにデータが存在する場合と同様に、データベース・インスタンスでデータを処理できます。
- 「平均アクティブ・セッション」グラフ: クラスタ・データベースの「パフォーマンス」ページの「平均アクティブ・セッション」グラフには、データベース内で発生する可能性がある問題が表示されます。「カテゴリ」は待機クラスとも呼ばれ、データベース内でCPUやディスクI/Oなどのリソースを使用している部分が表示されます。CPU時間を待機時間と比較すると、レスポンス時間のうちどれくらいの時間が、他のプロセスに保持されている可能性のあるリソースの待機ではなく有効な作業に消費されているかを確認できます。
- 「データベース・スループット」グラフ: 「データベース・スループット」グラフは、「平均アクティブ・セッション」グラフに表示される任意のリソース競合を要約する他、ユーザーやアプリケーションのためにデータベースが実行中の作業の量を示します。「1秒当たり」ビューは、1秒当たりのログオン数に対するトランザクションの数、REDOサイズに対する物理読取りの量を示します。「1トランザクション当たり」ビューは、トランザクション当たりのREDOサイズに対する物理読取りの量を示します。「ログオン」は、データベースにログオンしているユーザー数を示します。

さらに、クラスタ・データベースの「パフォーマンス」ページにある「トップ・アクティビティ」ドリルダウン・メニューでは、待機イベント、サービスおよびインスタンス単位でアクティビティを表示できます。また、グラフのスライダを使用して以前の時点に移動することによって、SQL/セッションの詳細を表示できます。

クラスタ・データベースの「パフォーマンス」ページには、Oracle RACデータベースのパフォーマンス統計のサマリーが表示されます。ユーザーがすべてのインスタンスを調べなくてもパフォーマンスの問題を特定できるように、統計はクラスタ・データベース内のすべて

のインスタンス間でロールアップされます。サービスに関連したパフォーマンスの問題の優先順位を決定しやすくするために、Oracle Enterprise Managerでは次のレベルでアクティビティ・データを集計します。

- 待機単位の集計

  すべてのアクティビティ・データが12のカテゴリ(CPU、スケジューラ、ユーザーI/O、システムI/O、同時実行性、アプリケーション、コミット、構成、管理、ネットワーク、クラスタおよびその他)で表示されます。表示されたデータは、実行中のすべてのインスタンスからロールアップされます。

- サービス単位の集計

  すべてのアクティビティ・データが、サービスごとにロールアップされます。この方法でアクティビティ・データを表示すると、最もアクティブなサービスおよび詳細な分析が必要なサービスを簡単に特定できます。

- インスタンス単位の集計

  同様の結果として、サービスが対象のサービスでない場合、アクティビティ・データはインスタンスごとにロールアップされます。

集計はアクティビティ・データが表示されているページ(「データベース・パフォーマンス」ページ、「トップ・アクティビティ」ページ、「待機の詳細」ページ、「サービスの詳細」ページなど)に表示されます。

Oracle RACデータベースのチューニング

非クラスタのOracle Databaseのすべてのチューニング方法は、Oracle RACデータベースに適用されます。

関連項目

- Oracle Database 2日でパフォーマンス・チューニング・ガイド
- Oracle Databaseパフォーマンス・チューニング・ガイド

データベース信頼性フレームワーク

データベース信頼性フレームワーク(DRF)は、プロアクティブで自動的な監視および修正のフレームワークです。

データベース信頼性フレームワークは、サービスの中断が発生する前に問題を検出するために、データベースの様々な階層で各種のメトリックを継続的に監視します。DRFは、データベースの重要なイベントを監視し、それらの重要なイベントが特定のしきい値を超えた場合に修正アクションを取ることによって、データベースの可用性を向上させます。

問題が特定されると、アクションが自動的に実装されます。識別された問題に応じて、内部メモリー構造のサイズ変更、Oracle RACプロセスの優先度の変更などのアクションがあります。たとえば、一定の期間に収集されたメトリックに基づくとI/Oの競合はないが`redo waits`が高いシステムについて考えてみます。使用可能なCPUリソースが十分にある場合は、`redo waits`を減らすためのアクション計画として、LGWRプロセスの優先度を高くして十分なCPUが確保されるようにすることが考えられます。DRFでは、すべてのメトリックを慎重に検討した後に、この処理が行われます。これにより、時間の経過に従って問題が悪化してデータベースの可用性に影響を与える前に、最小限のサービス中断で問題が解決されます。

Oracle RACのインターコネクト設定の検証

SQL文を使用して、Oracle RACのインターコネクト設定を検証します。

インターコネクトとノード間通信のプロトコルは、キャッシュ・フュージョンのパフォーマンスに影響を与える場合があります。さらに、インターコネクトの帯域幅、その待機時間およびIPCプロトコルの効率によって、キャッシュ・フュージョンがブロック転送を処理する速度が決まります。

接続したOracle RACデータベース・インスタンスのインターコネクト設定を検証するには、V$CLUSTER_INTERCONNECTSおよびV$CONFIGURED_INTERCONNECTSビューを問い合せます。たとえば:

例14-1 V$CLUSTER_INTERCONNECTSを使用したインターコネクト設定の検証

```
SQL> SELECT * FROM V$CLUSTER_INTERCONNECTS;
NAME            IP_ADDRESS       IS_PUBLIC SOURCE
--------------- ---------------  --- -------------------------------
eth2            10.137.20.181    NO        Oracle Cluster Repository
```

ノート:

GV$CLUSTER_INTERCONNECTS ビューを問い合せて、クラスタ内のすべてのインスタンスのエントリを表示できます。

例14-2 V$CONFIGURED_INTERCONNECTSを使用したインターコネクト設定の検証

```
SQL> SELECT * FROM V$CONFIGURED_INTERCONNECTS;
NAME            IP_ADDRESS       IS_PUBLIC   SOURCE
--------------- ---------------  --- ---------------------------------
eth2            10.137.20.181    NO          Oracle Cluster Repository
eth0            10.137.8.225     YES         Oracle Cluster Repository
```

インターコネクト処理への影響

インターコネクトが動作可能になると、そのパフォーマンスに大きな影響を与えることはできません。ただし、プロセス間通信(IPC)のバッファ・サイズを調整することで、インターコネクト・プロトコルの効率を変更できます。

Oracle Clusterwareでは、Oracle Cluster Registry (OCR)にシステムのインターコネクト情報が格納されます。クラスタのインターコネクトを特定するには、Oracle Interface Configuration (OIFCFG)コマンドライン・ユーティリティのoifcfg getifコマンドまたはOCRDUMPユーティリティを使用します。その後、OIFCFGコマンドを実行して、使用しているインターコネクトを変更できます。

CLUSTER_INTERCONNECTSパラメータを設定する必要はほとんどありませんが、このパラメータを使用して、プライベート・ネットワークIPアドレスまたはネットワーク・インタフェース・カード(NIC)を割り当てることができます。たとえば:

```
CLUSTER_INTERCONNECTS=10.0.0.1
```

オペレーティング・システム固有のベンダーIPCプロトコルを使用している場合は、トレース情報からIPアドレスが判明しない場合があります。

ノート:

- OIFCFG コマンドを使用しても、プライベート・ネットワーク IP アドレスまたはプライベート IP アドレスを割り当てることができます。
- Oracle Clusterware 12c リリース 2 (12.2)以降の Oracle Clusterware リリースでは、IPv4 アドレスまたは IPv6 アドレスを複数のプライベート・ネットワークに割り当てることができます。ただし、いずれか一方のプロトコルを選択し、クラスタ内のすべてのプライベート・ネットワークでそのプロトコルを使用する必要があります。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle Databaseリファレンス

Oracle RACのパフォーマンス・ビュー

Oracle Real Application Clusters (Oracle RAC)データベースに関するパフォーマンス情報を取得するには、インスタンス固有のビューを問い合せるか、クラスタ全体の動的パフォーマンス・ビューを問い合せることができます。

Oracle Real Application Clusters (Oracle RAC)データベース内の各インスタンスには、インスタンス固有の一連のビューがあり、これらにはV$という接頭辞が付いています。また、グローバル動的パフォーマンス・ビューを問い合せて、すべての修飾インスタンスからパフォーマンス情報を取得することもできます。グローバル動的パフォーマンス・ビュー名には、GV$という接頭辞が付きます。

GV$ビューを問い合せると、すべてのインスタンスからV$ビュー情報が取り出されます。V$情報に加えて、各GV$ビューには、NUMBERデータ型のINST_IDという追加の列が含まれています。INST_ID列には、関連するV$ビュー情報の取得元のインスタンス番号が表示されます。

フィルタとしてINST_ID列を使用すると、使用可能なインスタンスのサブセットからV$情報を取り出せます。たとえば、次の問合せでは、インスタンス2および5のV$LOCKビューから情報を取り出します。

```
SQL> SELECT * FROM GV$LOCK WHERE INST_ID = 2 OR INST_ID = 5;
```

関連項目

- Oracle Databaseリファレンス

CATCLUST.SQLによるOracle RACのデータ・ディクショナリ・ビューの作成

Oracle RACデータベースの作成にOracle DBCAを使用しなかった場合は、CATCLUST. SQLスクリプトを実行してOracle RAC関連のビューおよび表を作成する必要があります。

Oracle Database Configuration Assistant (Oracle DBCA)を使用してOracle Real Application Clusters (Oracle RAC)データベースを作成しなかった場合、Oracle RACのデータ・ディクショナリの設定は完了していません。Oracle RACに関連するビューおよび表を作成するには、CATCLUST. SQLスクリプトを実行する必要があります。CATCLUST. SQLスクリプト

を実行するには、使用するユーザー・アカウントにSYSDBA権限を付与する必要があります。

Oracle RACのパフォーマンス統計

Oracle Real Application Clusters (Oracle RAC)の統計は、メッセージ・リクエスト・カウンタまたは定期的な統計として表示されます。

メッセージ要求カウンタには、特定のタイプのブロック・モード変換の数を示す統計が含まれます。定期的な統計は、特定のタイプの操作での読取りおよび書込みI/Oに対する、合計または平均の待機時間を示します。

Oracle RAC環境における自動ワークロード・リポジトリ

自動ワークロード・リポジトリを使用して、Oracle RACデータベースに関連するパフォーマンスの統計を監視できます。

自動ワークロード・リポジトリ(AWR)では、パフォーマンス・データのスナップショットを1時間ごとに自動生成し、統計をワークロード・リポジトリに収集します。Oracle RAC環境では、AWRの各スナップショットが、クラスタ内のすべてのアクティブなインスタンスからのデータを取得します。取得される各スナップショット・セットのデータは、同じ時点のものです。AWRでは、すべてのインスタンスのスナップショット・データが同じ表に格納され、データはインスタンス修飾子で識別されます。たとえば、BUFFER_BUSY_WAIT統計では、各インスタンスのバッファ待機の数が示されます。AWRでは、クラスタ全体から集計されるデータは格納されません。つまり、データは各インスタンスごとに個別に格納されます。

自動データベース診断モニター(ADDM)を使用すると、Oracle Databaseについて考えられるパフォーマンス上の問題がないか、AWRで収集された情報を分析できます。ADDMでは、クラスタ全体の観点からパフォーマンス・データを表示するため、全体的なパフォーマンス分析が可能です。Oracle RAC環境では、ADDMは、すべてのインスタンスから収集されたデータを使用してパフォーマンスを分析し、次のような様々なレベルの粒度で表示できます。

- クラスタ全体の分析
- 特定のデータベース・インスタンスの分析
- データベース・インスタンスのサブセットの分析

これらの分析を行うには、ADDMアドバイザをOracle RACのADDMのモードで実行してクラスタ全体の分析を行うか、ローカルADDMのモードで個々のインスタンスのパフォーマンスを分析するか、または部分ADDMモードでインスタンスのサブセットを分析します。Oracle Enterprise Managerのアドバイザ・セントラルまたはDBMS_ADVISORパッケージおよびDBMS_ADDM PL/SQLパッケージを介したアドバイザ・フレームワークを使用して、ADDM分析をアクティブ化します。

関連項目

- Oracle Databaseパフォーマンス・チューニング・ガイド
- Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス

Oracle RACのアクティブ・セッション履歴レポート

この項では、Oracle RACのアクティブ・セッション履歴(ASH)レポートについて説明します。内容は次のとおりです。

- Oracle RACのASHレポートの概要

- Oracle RACのASHレポート: トップ・クラスタ・イベント
- Oracle RACのASHレポート: トップ・リモート・インスタンス

関連項目

- Oracle Databaseパフォーマンス・チューニング・ガイド

Oracle RACのASHレポートの概要

ASHはOracle Databaseの自己管理フレームワークにとって不可欠な部分であり、Oracle RAC環境のパフォーマンスの問題の診断に役立ちます。ASHレポート統計は、Oracle Databaseのセッション・アクティビティに関する詳細を示します。Oracle Databaseでは、Oracle RACのすべてのアクティブ・インスタンスのアクティブ・セッションに関する情報を記録して、このデータをシステム・グローバル領域(SGA)に格納します。データベースに接続してCPUを使用しているすべてのセッションが、アクティブ・セッションとみなされます。ただし、アイドル状態の待機クラスに属するイベントを待機しているセッションは例外です。

ASHレポートは、アクティブ・セッションの情報のみを取り込むことによって、管理可能なデータのセットを表示します。データの量は、システムで許可されているセッションの数ではなく、実行されている作業に直接関連します。

指定した期間中に収集されたASH統計をASHレポートに含めることができます。各ASHレポートは、ADDM分析には表示されない短時間のパフォーマンスの問題を特定できるように、複数のセクションに分割されています。Oracle RAC固有の2つのASHレポート・セクションは、次の2つの項で説明するとおり、トップ・クラスタ・イベントとトップ・リモート・インスタンスです。

Oracle RACのASHレポート: トップ・クラスタ・イベント

ASHレポートのトップ・クラスタ・イベント・セクションは、Oracle RAC固有のトップ・イベント・レポートの一部です。トップ・クラスタ・イベント・レポートでは、クラスタ待機クラスのイベントのうちセッション・アクティビティの割合が最も高いイベントと、影響を受けるインスタンスのインスタンス番号が表示されます。この情報を使用して、クラスタ待機イベントの割合を上げているイベントおよびインスタンスを特定できます。

Oracle RACのASHレポート: トップ・リモート・インスタンス

ASHレポートのトップ・リモート・インスタンス・セクションは、Oracle RAC固有のトップ・ロード・プロファイル・レポートの一部です。トップ・リモート・インスタンス・レポートでは、クラスタ待機イベントと、セッション・アクティビティの割合が最も高いインスタンスのインスタンス番号が表示されます。この情報を使用して、クラスタ待機期間を長くしているインスタンスを特定できます。

Oracle RACの統計および待機イベントの監視

この項では、Oracle RAC固有の待機イベントおよび統計を説明し、自動ワークロード・リポジトリ(AWR)、Statspackまたは動的パフォーマンス・ビューの非定型問合せによって生成されたパフォーマンス・データを評価する場合の、待機イベントおよび統計の解析方法も説明します。

この項には次のトピックが含まれます:

- AWRおよびStatspackレポートでのOracle RAC統計およびイベント
- Oracle RACの待機イベント

- GCS統計とGES統計の分析によるパフォーマンス監視
- GCS統計を使用したキャッシュ・フュージョンによる転送の影響の分析
- 待機イベントに基づく応答時間の分析

関連項目

- Oracle Databaseパフォーマンス・チューニング・ガイド

AWRおよびStatspackレポートでのOracle RAC統計およびイベント

AWRおよびStatspackによって生成された統計スナップショットは、サマリー・データ(定期的な統計に基づくロード・プロファイルおよびクラスタ・プロファイル、インスタンスごとに収集された待機イベントなど)を表示するレポートを作成して評価します。

ほとんどの関連データは、Oracle RACの「統計」ページに集約されます。次のような情報があります。

- グローバル・キャッシュのロード・プロファイル
- グローバル・キャッシュの効率(パーセント)—ワークロード特性
- グローバル・キャッシュおよびエンキュー・サービス(GES)—メッセージ機能の統計

レポートの後半には、次の追加のOracle RACセクションが表示されます。

- グローバル・エンキュー統計
- グローバルCR統計
- 実行済のグローバルCURRENT統計
- グローバル・キャッシュの送信統計。

Oracle RACの待機イベント

セッション待機の原因の分析および解析は、時間がかかっている場所を判別するための重要な手段です。

Oracle RACの場合、要求の結果を正確に反映するイベントにより待機時間が発生します。たとえば、インスタンスのセッションがグローバル・キャッシュ内のブロックを検索している場合、このセッションは、別のインスタンスがキャッシュしたデータを受け取るかどうか、ディスクから読み込むためのメッセージを受け取るかどうかは判別できません。グローバル・キャッシュに関する待機イベントは正確な情報を伝達し、グローバル・キャッシュのブロックまたはメッセージを待機中のイベントは次の状態になります。

- クラスタ待機クラスと呼ばれる広範囲のカテゴリに集約されます。
- ブロック待機中にアクティブになるプレースホルダ・イベントによって一時的に表されます。たとえば:
  - `gc current block request`
  - `gc cr block request`
- 要求の結果がわかっている場合は、正確なイベントを提供します。たとえば:
  - `gc current block 3-way`
  - `gc current block busy`
  - `gc cr block grant 2-way`
- すべてのディスク読取りが優先される場合の複数ブロック読取りリクエスト・イベント。たとえば:

- gc cr multi block grant
- gc cr multi block mixed

つまり、Oracle RACの待機イベントは、パフォーマンス分析にとって重要な情報を伝達します。これらのイベントは、キャッシュ・フュージョンの影響を正確に診断するために、Automatic Database Diagnostic Monitor(ADDM)で使用されます。

GCS統計とGES統計の分析によるパフォーマンス監視

インスタンス間のメッセージ交換および競合に関連する作業量およびコストを判断するには、次の項の説明に従って、ブロック転送率、各トランザクションで発生したリモート要求、グローバル・キャッシュ・イベントの待機数および待機時間を調べます。

- キャッシュ・フュージョンがOracle RACに与える影響の分析
- GCS統計とGES統計を使用したパフォーマンス分析

キャッシュ・フュージョンがOracle RACに与える影響の分析

グローバル・キャッシュ内のブロックへのアクセス、および一貫性を保持した場合の効果は、次の情報で示されます。

- gc current blocks received、gc cr blocks receivedなどのcurrentおよびcrブロックのグローバル・キャッシュ・サービス(GCS)統計
- gc current block 3-way、gc cr grant 2-wayなどのGCS待機イベント

キャッシュ・フュージョンによる転送の応答期間は、物理的なインターコネクト・コンポーネント、IPCプロトコルおよびGCSプロトコルの制限が適用されるメッセージングおよび処理時間によって決まります。不定期に発生するログ書込み以外のディスクI/O要因による影響は受けません。キャッシュ・フュージョン・プロトコルの場合、キャッシュ一貫性 を保証するためのデータファイルへのI/Oを行う必要はなく、基本的にOracle RACでは、非クラスタ・インスタンスよりも多くのディスクI/Oは発生しません。

GCS統計とGES統計を使用したパフォーマンス分析

すべてのインスタンスによる使用頻度の高い(ホットな)データ・ブロックとオブジェクトを識別して、GCSのパフォーマンスを監視できます。

並行処理回数の多いブロックは、GCSの待機イベントおよび待機数によって特定できます。

gc current block busyの待機イベントは、リモート・キャッシュまたはローカル・キャッシュがビジーなため、キャッシュ・データ・ブロックへのアクセスが遅延状態であることを示します。この原因として、次のいずれかが考えられます。

- ブロックが確保されている
- ブロックがセッションによって保留されている
- ブロックがリモート・インスタンス側のログ書込みにより遅延されている
- 同一インスタンス上のセッションが、インスタンス間を移動中のブロックにアクセスしているため、現行のセッションが待機状態(gc current block busyなど)になっている

V$SESSION_WAITビューを使用して、競合するオブジェクトおよびデータ・ブロックを識別します。GCS待機イベントには、p1およびp2のブロック・リクエストのファイルおよびブロック番号がそれぞれ含まれています。

前述のV$SESSION_WAITへの問合せなしで、ビジー・オブジェクトを迅速に判別するために、セグメント統計gc buffer busyが追加されています。

AWRインフラストラクチャは、最近の待機イベントおよびその引数のトレースにも使用できるアクティブ・セッション履歴のビューを提供します。そのため、ホット・ブロック分析に有効です。AWRおよびStatspackで使用されるほとんどのレポート機能に、オブジェクト統計およびクラスタ待機クラスのカテゴリが含まれるため、前述のビューのサンプリングが必要になることはほとんどありません。

ノート:

ADDM および AWR を使用することをお薦めします。ただし、Statspack には下位互換性があります。Statspack には、レポート機能のみが用意されています。ブロック競合およびセグメント・ブロックの待機に関連する統計を収集するには、Statspack をレベル 7 で実行する必要があります。

AWRインフラストラクチャで収集されたスナップショット・データに対してADDMを実行し、グローバル・キャッシュが与える影響の全体的な評価を取得することをお薦めします。この評価では、ビジー・オブジェクトおよびSQLでの待機時間が最も長いクラスタも特定されます。

GCS統計を使用したキャッシュ・フュージョンによる転送の影響の分析

読込み頻度と変更頻度の高いオブジェクトおよびリモート・アクセスにより発生するサービス時間を識別して、GCSのパフォーマンスを監視する方法について説明します。

ディスク・アクセス待機時間よりもキャッシュ・フュージョンによる転送時間の方が通常は短いという事実はあるものの、ディスクからの読取りによってブロック・アクセス遅延が増加する場合と同様に、ブロックの到着待ちが応答時間のほとんどを占める場合があります。

次の待機イベントは、ブロックの待機、確保またはログ・フラッシュなしでリモート・キャッシュ・ブロックがローカル・インスタンスへ送信されたことを示します。

- gc current block 2-way
- gc current block 3-way
- gc cr block 2-way
- gc cr block 3-way

gc current blocks receivedおよびgc cr blocks receivedについてのオブジェクト統計により、アクティブ・インスタンスで共有される索引および表が簡単に特定できます。前述のとおり、通常は、ADDMによる分析を行うと、インスタンス間の競合によって影響を受けるSQL文およびデータベース・オブジェクトが特定されます。

前述のイベントの平均待機時間を増加させる原因は、次のとおりです。

- 高負荷: CPUの容量不足、長い実行キュー、スケジュールの遅延
- 設定の問題: メッセージおよびブロック通信量にプライベート・インターコネクトではなくパブリック・インターコネクトを使用している

平均待機時間は適切で、インターコネクトまたは負荷には問題がないと診断された場合は、SQL文が累計待機時間の原因になっている可能性があります。アクセスするブロック数を最小にするために、SQL文をチューニングする必要があります。

V$SQLAREAのCLUSTER_WAIT_TIME列は、グローバル・キャッシュ・イベントの個々のSQL文によって発生する待機時間を表し、

チューニングを必要とするSQLを特定します。

待機イベントに基づく応答時間の分析

AWRおよびStatspackレポートまたは動的パフォーマンス・ビューで高い合計時間が示されるほとんどのグローバル・キャッシュ待機イベントは正常で、実際に問題があるのではなく、データベース時間の上位の使用者として表示されることがあります。

この項では、パフォーマンス・データを解析する際に注意が必要な、発生頻度の高い待機イベントについて説明します。

ユーザーの応答時間が長くなり、グローバル・キャッシュでの待機時間の比率が高い場合は、その原因を特定する必要があります。ほとんどのレポートには、合計時間に対する待機時間の割合の順で待機イベントが表示されます。

最初に、定期的に収集されるパフォーマンス統計の影響と、待機時間のほとんどを占めるオブジェクトおよびSQLを特定するADDMレポートを分析します。その後、AWRおよびStatspackにより生成される詳細なレポートを分析することをお薦めします。

Oracle RACの待機イベントは、次のカテゴリに分類されます。

- ブロック関連の待機イベント
- メッセージ関連の待機イベント
- 競合関連の待機イベント
- ロード関連の待機イベント

ブロック関連の待機イベント

ブロック関連待機の主な待機イベントは、次のとおりです。

- gc current block 2-way
- gc current block 3-way
- gc cr block 2-way
- gc cr block 3-way

ブロック関連待機イベント統計は、ブロックが2方向または3方向メッセージの結果として受信されたことを示します。つまり、1メッセージおよび1転送を必要とするリソース・マスターから、ブロックが送信されたか、または2メッセージおよび1ブロック転送を必要とする別のノード(送信元)へブロックが転送されたことを示します。

メッセージ関連の待機イベント

メッセージ関連待機の主な待機イベントは、次のとおりです。

- gc current grant 2-way
- gc cr grant 2-way

メッセージ関連待機イベント統計は、インスタンスにブロックがキャッシュされなかったために、ブロックが受信されなかったことを示します。かわりに、要求側インスタンスがディスクからのブロックを読み取ったり、ブロックを変更できるグローバルな権限が付与されます。

これらのイベントの処理時間が長い場合は、使用頻度の高いSQLによるディスクI/O回数の増加(cr grantの場合)、またはワークロードによって大量のデータが挿入されたため、頻繁に新しいブロックを検索しフォーマットする必要があると(current grantの場合)考えられます。

競合関連の待機イベント

競合関連待機の主な待機イベントは、次のとおりです。

- gc current block busy
- gc cr block busy
- gc buffer busy acquire/release

競合関連の待機イベント統計は、別のノードのセッションによって確保されたブロックが受信されたものの、ディスクへの変更のフラッシュが完了していないため、または並行性が高いために保留され、すぐに送信できない状態になっていることを示します。セッションがキャッシュ・フュージョン操作をすでに開始している場合は、バッファは、ローカルでもビジーになる可能性があり、同一ノード上の別のセッションが、同一データの読取りまたは変更を行う場合、バッファはそれが完了するまで待機します。グローバル・キャッシュ内でやり取りされるブロックのサービス時間が長くなると、競合状態が悪化する可能性がありますが、これは、同一データに対して頻繁に行われる同時読取りおよび書込みアクセスが原因の可能性もあります。

gc current block busyおよびgc cr block busyの待機イベントは、要求を作成しているローカル・インスタンスが、カレント・ブロックまたはCRブロックをすぐに受信しなかったことを示します。これらのイベント名にあるbusyという用語は、リモート・インスタンスでブロックの送信が遅延状態であったことを示します。たとえば、ブロックの変更のREDOがOracle Databaseによってまだログ・ファイルに書き込まれていない場合は、ブロックをすぐに送信することはできません。

block busy待機イベントと比較した場合、gc buffer busyイベントには、Oracle Databaseではローカル・バッファ・キャッシュに格納されているデータへのアクセス権をすぐに付与できないことが示されています。これは、バッファに対するグローバル操作が保留中で、操作がまだ完了していないためです。つまり、バッファはビジー状態であり、ローカル・バッファにアクセスしようとしている他のすべてのプロセスで完了を待機する必要があります。

また、gc buffer busyイベントの存在は、ブロック競合が存在し、結果としてローカル・ブロックに対する複数のアクセス要求が発生していることも意味します。Oracle Databaseはこれらの要求をキューに入れる必要があります。Oracle Databaseによるキューの処理に必要な時間の長さは、ブロックの残りのサービス時間に依存します。サービス時間は、ネットワーク待機時間によって加算される処理時間、リモートおよびローカル・インスタンスの処理時間および待機キューの長さによって影響を受けます。

これらの待機による影響が大きく、パフォーマンスに問題が発生するというアラートを受けた場合は、平均待機時間および合計待機時間を検討する必要があります。通常は、インターコネクトかロードの問題、または大きい共有作業セットに対して実行されるSQLが根本的な原因と考えられます。

ロード関連の待機イベント

ロード関連待機の主な待機イベントは、次のとおりです。

- gc current block congested
- gc cr block congested

ロード関連待機イベントは、GCSで処理遅延が発生したことを示します。通常、その原因となるのは、高負荷、飽和状態のCPUです。この問題は、CPUの追加、ロード・バランシング、別のラウンドトリップ時間帯または新しいクラスタ・ノードへの処理のオフロード処理によって解決できます。前述のイベントについての待機時間には、セッションが開始し、ブロック要求開始後、ブロックの到着を待機するまでのラウンドトリップ全体が含まれます。

15 シングル・インスタンスOracle DatabaseのOracle RACおよびOracle RAC One Nodeへの変換

Oracle Databaseシングル・インスタンス・データベースからOracle Real Application Clusters(Oracle RAC)およびOracle RAC One Nodeデータベースに変換するための手順。

この付録の手順は、元のシングル・インスタンス・データベースとターゲットのOracle RACデータベースが同じリリースを使用して、同じプラットフォーム上で実行されていることを前提としています。以前のバージョンのOracle RACからOracle RAC 12cにアップグレードする場合は、Oracle Database Upgrade Assistant (DBUA)を使用します。

この章の内容は次のとおりです。

- データベースをOracle RACに変換する場合の管理上の問題点
- DBCAを使用したOracle RACおよびOracle RAC One Nodeへの変換
- rconfigおよびOracle Enterprise Managerを使用して変換するための準備
- rconfigを使用したデータベースのOracle RACへの変換
- ConvertToRAC用のrconfig XML入力ファイルの例
- 変換後のステップ

ノート:

Oracle RAC データベースでは、クラスタ化された Oracle Automatic Storage Management(Oracle ASM)インスタンスを使用する必要があります。

関連項目

- Oracle Databaseのオプションおよびそれらに許可されている機能

データベースをOracle RACに変換する場合の管理上の問題点

シングル・インスタンスのデータベースをOracle RACに変換する前に、管理上の考慮事項に対処しておく必要があります。

- 単一インスタンスのOracle DatabaseからOracle RACに変換する前に、バックアップを実行可能にする必要があります。これには、Oracle RACへの変換前に既存のデータベースのバックアップを作成することと、変換直後にOracle RACデータベースのバックアップを作成する準備をすることも含まれます。
- Oracle RAC環境でアーカイブする場合、アーカイブ・ファイル形式にはスレッド番号が必要です。
- メディア・リカバリには、Oracle RACデータベースのすべてのインスタンスのアーカイブ・ログが必要です。この要件のため、ファイルにアーカイブして、クラスタ・ファイル・システムを使用しない場合または共有ファイル・システムを使用するための他の方法を採用しない場合、クラスタ・データベースのインスタンスがあるすべてのノードからアーカイブ・ログにアクセスするなんらかの方法が必要です。
- デフォルトでは、すべてのデータベース・ファイルはOracle Managed Filesに移行されます。この機能によって、表領域の作成が簡単になり、データ・ファイルの場所の一貫性およびOracle Flexible Architecture規則への準拠が確

保されるため、データ・ファイル管理での人為的なミスが減少します。

DBCAを使用したOracle RACおよびOracle RAC One Nodeへの変換

Database Configuration Assistant (DBCA)を使用して、シングル・インスタンスのOracle DatabaseをOracle RACまたはOracle RAC One Nodeデータベースに変換できます。

DBCAを使用すると、制御ファイル属性が自動的に構成され、UNDO表領域とREDOログが作成されて、クラスタ対応環境用の初期化パラメータ・ファイルのエントリが作成されます。また、DBCAは、Oracle Enterprise Managerまたはサーバー制御ユーティリティ(SRVCTL)を使用して、Oracle Net ServicesとOracle Clusterwareリソースの構成およびOracle RACデータベース管理用の構成を行います。

DBCAを使用してシングル・インスタンスのデータベースをOracle RACまたはOracle RAC One Nodeデータベースに変換する前に、システムが次の条件を満たしていることを確認します。

- システムで、サポートされているハードウェアおよびオペレーティング・システム・ソフトウェアが使用されている。システムが、Oracle RACデータベースをサポートするように適切に構成されている。
- ノードから共有記憶域にアクセスでき、たとえば、Oracle Cluster File SystemまたはOracle ASMが使用可能で、すべてのノードからアクセスできる。Linux on POWER Systemsの場合は、GPFSが使用可能で、すべてのノードからアクセスできる。
- 使用しているアプリケーションが、その特性によりクラスタ・データベース・プロセスで使用不可能になることがない。

ご使用のプラットフォームがクラスタ・ファイル・システムをサポートしている場合は、Oracle RACでそのクラスタ・ファイル・システムを使用できます。Oracle RACに変換して、非共有ファイル・システムを使用することもできます。いずれの場合も、Oracle Universal Installerを使用してOracle Database 12cをインストールし、クラスタで選択された各ノード上の同じ場所にOracleホームおよびインベントリを設定することをお薦めします。

この項には次のトピックが含まれます:

- DBCAを使用したOracle DatabaseのインストールのOracle RACへの変換
- DBCAを使用したクラスタ上のシングル・インスタンスのOracle RAC One Nodeへの変換
- DBCAを使用したクラスタ上のシングル・インスタンスのOracle RACへの変換

関連項目

- データベースの変換

DBCAを使用したOracle DatabaseのインストールのOracle RACへの変換

クラスタ・コンピュータ以外のコンピュータ上にあるシングル・インスタンスのOracle DatabaseをOracle RACに変換するには、次の項に説明する手順を、その順序で実行します。

- DBCAを使用したシングル・インスタンス・データベースのイメージの作成
- Oracle Clusterwareのインストールの完了
- クラスタの検証
- 事前構成済データベース・イメージのコピー

- Oracle Database 12cソフトウェアおよびOracle RACのインストール

DBCAを使用したシングル・インスタンス・データベースのイメージの作成

次の手順に従い、DBCAを使用してシングル・インスタンス・データベースの事前構成済イメージを作成します。

1. $ORACLE_HOMEの下のbinディレクトリに移動して、DBCAを起動します。
2. 「ようこそ」ページで「次へ」をクリックします。
3. 「操作」ページで、「テンプレートの管理」を選択して「次へ」をクリックします。
4. 「テンプレート管理」ページで、「データベース・テンプレートの作成」および「既存のデータベースを使用(データおよび構造)」を選択して「次へ」をクリックします。
5. 「ソース・データベース」ページで、「データベース・インスタンス」リストでデータベース名を選択して「次へ」をクリックします。
6. 次に示すように、SQLを使用して、すべてのプラガブル・データベース(PDB)がオープンしていることを確認します。

   ```
   SQL> SELECT name, open_mode FROM v$pdbs;
   ```

   いずれかのPDBのステータスがOPEN以外の場合は、SQLを使用してオープンにします。
7. 「テンプレート・プロパティ」ページで、「名前」フィールドにテンプレート名を入力します。データベース名を使用することをお薦めします。

   デフォルトでは、テンプレート・ファイルはディレクトリ$ORACLE_HOME/assistants/dbca/templatesに生成されます。「説明」フィールドにファイルの説明を入力して、「テンプレート・データファイル」フィールドでテンプレート・ファイルの場所を変更できます。

   入力が完了したら、「次へ」をクリックします。
8. 「データベース関連ファイルの位置」ページで、現行のディレクトリ構造にデータベースをリストアできるように「ファイル位置を保持」を選択して「終了」をクリックします。

DBCAは、データベース構造ファイル(template_name.dbc)およびデータベースの事前構成済イメージ・ファイル(template_name.dfb)の2つのファイルを生成します。

Oracle Clusterwareのインストールの完了

Oracle Clusterwareのインストールを完了します。

関連項目

- Oracle Grid Infrastructureのインストレーション・ガイド

クラスタの検証

クラスタ検証ユーティリティ(CVU)を使用して、クラスタの構成を検証します。

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

事前構成済データベース・イメージのコピー

事前構成済データベース・イメージをコピーします。前の項でDBCAを使用して作成したデータベース構造ファイル(*.dbc)およ

びデータベースの事前構成済イメージ・ファイル(*.dfb)も、DBCAを実行するクラスタのノード上の一時的な場所にコピーします。

関連項目

- DBCAを使用したシングル・インスタンス・データベースのイメージの作成

Oracle Database 12cソフトウェアおよびOracle RACのインストール

1. Oracle Universal Installerを実行して、Oracle DatabaseおよびOracle RACをインストールします。
2. Oracle Universal Installerのハードウェアのクラスタ・インストールの指定ページでクラスタ・インストール・モードを選択し、Oracle RACデータベースに含めるノードを選択します。
3. Oracle Universal Installerの「データベース構成タイプ」ページで、「拡張」インストール・タイプを選択します。

   Oracle Databaseソフトウェアのインストール後、Oracle Universal Installerはインストール後の構成ツール(Net Configuration Assistant (NETCA)、DBCAなど)を実行します。
4. DBCAのテンプレートを選択するページで、前の項で一時的な位置にコピーしたテンプレートを使用します。テンプレートの位置を選択するには、「参照」オプションを使用します。

   デプロイするオプションを選択します。Oracle RACデータベース、Oracle RAC One Nodeデータベースまたはシングル・インスタンスOracle Databaseから選択できます。
5. Oracle RACデータベースを作成すると、DBCAによって「パスワード管理」ページが表示され、このページで、SYSDBAとSYSOPERの権限を持つデータベース・ユーザーのパスワードを変更する必要があります。DBCAを終了すると、変換処理が完了します。

DBCAを使用したクラスタ上のシングル・インスタンスのOracle RAC One Nodeへの変換

DBCAでシングル・インスタンスOracle DatabaseをOracle RAC One Nodeに変換するには、次の手順を実行します。

1. $ORACLE_HOME/binディレクトリに移動します。
2. 次のように、DBCAを起動します。

   ```
   $ dbca
   ```
3. 「ようこそ」ウィンドウから、Oracle RAC One Nodeデータベースを選択します。
4. 前の項でデプロイするために選択したテンプレートを使用します。

DBCAを使用したクラスタ上のシングル・インスタンスのOracle RACへの変換

シングル・インスタンス・データベースがクラスタ・ノード上に存在する場合は、次の3つのシナリオが考えられます。

- シナリオ1: シングル・インスタンス・データベースのOracleホームがクラスタ・ノードにインストールされ、Oracle RACが有効になっている場合。
- シナリオ2: シングル・インスタンス・データベースのOracleホームはクラスタ・ノードにインストールされているものの、このOracleホームのOracle RAC機能が無効になっている場合。
- シナリオ3: シングル・インスタンス・データベースのOracleホームが、クラスタのローカル・ノードのみにインストールされている場合。これは、Oracle Database 12cをインストールするときに、Oracle Universal Installerの「ハードウェア

のクラスタ・インストール・モードの指定」ページで「ローカル・インストール」オプションを選択した場合に発生します。

関連項目

- RAC対応のOracleホームからクラスタ上のシングル・インスタンス・データベースが実行されている場合
- RAC非対応のOracle ホームからクラスタ上のシングル・インスタンス・データベースが実行されている場合
- DBCAを使用したOracle DatabaseのインストールのOracle RACへの変換
- Oracle Database 12cソフトウェアおよびOracle RACのインストール

RAC対応のOracleホームからクラスタ上のシングル・インスタンス・データベースが実行されている場合

Oracle RACオプションが有効なOracleホームから実行されているクラスタ・ノードでシングル・インスタンス・データベースを変換するには、次の手順を実行します。

1. DBCAを使用して、シングル・インスタンス・データベースの事前構成済イメージを作成します。手動で変換を実行するには、シングル・インスタンス・データベースを停止します。
2. ノードをクラスタに追加します。すべてのノードがOracle ClusterwareおよびOracle RACで使用される共有記憶域にアクセスできることを確認します。
3. 既存のOracleホームから、このホームを新しいノードに拡張します。
4. 新しく追加したノードから、NETCAを使用して追加のノードにリスナーを構成します。既存のノードで使用したポート番号およびプロトコルと同じポート番号およびプロトコルを選択します。NETCAでノード・リスト・ページに既存のノードが表示される場合は、リスナーがすでに構成されているため、ノードを選択しないでください。
5. 次のいずれかの手順でデータベースを変換します。
    - DBCAを使用した自動変換の手順
    - 手動変換の手順

関連項目

- DBCAを使用したシングル・インスタンス・データベースのイメージの作成
- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle ClusterwareがインストールされたノードへのOracle RACの追加

DBCAを使用した自動変換の手順

前の項の説明に従って、DBCAを使用してシングル・インスタンス・データベースの事前構成済イメージを作成する場合は、次のステップを実行して、Oracle RACデータベースへの変換を完了します。

1. 元のノードからDBCAを起動します。クラスタ・データベースの一部として含めるノードの名前を選択します。テンプレートの選択ページで、作成した事前構成済テンプレートを選択します。データベース名を入力し、DBCAのプロンプトに従って残りの項目を入力します。
2. Oracle Databaseデータ・ファイルの共有記憶域の場所を指定します。

Oracle RACデータベースを作成すると、DBCAによって「パスワード管理」ページが表示され、このページで、SYSDBAとSYSOPERの権限を持つデータベース・ユーザーのパスワードを変更する必要があります。DBCAを終了すると、変換処理が完了します。

関連項目

- DBCAを使用したシングル・インスタンス・データベースのイメージの作成
- 手動変換の手順

手動変換の手順

前の項の説明に従って、DBCAを使用してシングル・インスタンス・データベースの事前構成済イメージを作成しなかった場合は、次のステップを実行して、変換を完了します。

1. 追加した各ノード上にOptimal Flexible Architectureディレクトリ構造を作成します。
2. SQL文のCREATE CONTROLFILEをREUSEキーワード付きで実行して制御ファイルを再作成し、Oracle RAC構成に必要なMAXINSTANCESやMAXLOGFILESなどを指定します。MAXINSTANCESのデフォルト値は、32に指定することをお薦めします。
3. データベース・インスタンスを停止します。
4. シングル・インスタンス・データベースでSPFILEを使用していた場合は、次のSQL文を使用して、SPFILEから一時的なパラメータ・ファイル(PFILE)を作成します。

   ```
   CREATE PFILE='pfile_name' from spfile='spfile_name'
   ```

5. CLUSTER_DATABASEパラメータをTRUEに設定し、sid.parameter=value構文を使用して、INSTANCE_NUMBERパラメータをインスタンスごとに一意の値に設定します。

   シングル・インスタンス・データベースのメモリ使用量を最適化していた場合は、システム・グローバル領域(SGA)のサイズを調整して、Oracle RACへの変換時にスワップおよびページングが発生しないようにします。この調整が必要な理由は、Oracle RACでは、グローバル・キャッシュ・サービス(GCS)用に、各バッファに約350バイトずつ必要になるためです。たとえば、バッファが10,000ある場合、Oracle RACは約350×10,000バイトの追加メモリーを必要とします。したがって、DB_CACHE_SIZEパラメータとDB_nK_CACHE_SIZEパラメータをこれに応じて変更し、SGAのサイズを調整します。
6. ステップ4で作成したPFILEを使用して、データベース・インスタンスを起動します。
7. シングル・インスタンス・データベースで自動UNDO管理を使用していた場合は、CREATE UNDO TABLESPACE SQL文を使用して、追加インスタンスごとにUNDO表領域を作成します。
8. 2つ以上のREDOログを持つREDOスレッドを追加インスタンスごとに作成します。SQL文のALTER DATABASEを使用して、新しいREDOスレッドを使用可能にします。その後で、データベース・インスタンスを停止します。
9. Oracleパスワード・ファイルを、元のノードまたは作業中のノードから追加ノード(クラスタ・データベースのインスタンスが存在するノード)の対応する位置にコピーします。追加インスタンスごとに、各パスワード・ファイルのORACLE_SID名を適切に置換します。
10. REMOTE_LISTENERパラメータに単一クライアント・アクセス名(SCAN)およびポートを設定します。
11. データベースとインスタンスのネット・サービス・エントリ、インスタンスごとのLOCAL_LISTENERのアドレス・エントリ、およびtnsnames.oraファイルのREMOTE_LISTENERのアドレス・エントリを構成し、tnsnames.oraファイルをすべてのノードにコピーします。
12. PFILEからSPFILEを作成します。
13. 次のエントリを含む$ORACLE_HOME/dbs/initsid.oraファイルを作成します(ここで、spfile_path_nameはSPFILEの完全パス名です)。

    ```
    spfile='spfile_path_name'
    ```

14. ローカル・ノードで、SQL*Plusを使用してcatclust.sqlを実行します。このスクリプトによって、Oracle RACデータベースに必要なディクショナリ・ビューが作成されます。たとえば:

```
SQL> start ?/rdbms/admin/catclust.sql
```

15. SRVCTLを使用して、Oracle RACまたはOracle RAC One Nodeデータベースの構成とそのインスタンスのノードへのマッピングを追加します。

    a. Oracle RACデータベースの構成を追加するには、次のコマンドを使用します。

```
$ srvctl add database -dbname db_name -oraclehome Oracle_home -spfile spfile_path_name
$ srvctl add instance -dbname db_name -instance inst1_name -node node1_name
$ srvctl add instance -dbname db_name -instance inst2_name -node node2_name
...
```

    b. Oracle RAC One Nodeデータベースの構成を追加するには、次のコマンドを使用します。

```
$ srvctl add database -dbname db_name -dbtype RACONENODE -oraclehome Oracle_home
  -spfile spfile_path_name
```

16. SRVCTLを使用してOracle RACまたはOracle RAC One Nodeデータベースを起動します。

```
srvctl start database -d db_name
```

SRVCTLでデータベースを起動すると、変換処理は完了です。次のSQL文を実行すると、Oracle RACデータベースのすべてのインスタンスのステータスを確認できます。

```
SQL> SELECT * FROM v$active_instances;
```

関連項目

- DBCAを使用したシングル・インスタンス・データベースのイメージの作成
- 『Oracle Real Application Clustersインストレーション・ガイド』

RAC非対応のOracle ホームからクラスタ上のシングル・インスタンス・データベースが実行されている場合

Oracle RACオプションが無効なOracleホームから実行されるクラスタで、シングル・インスタンス・データベースを作成できます。Oracle RAC非対応のクラスタでOracleホームを作成する場合は、Oracle Databaseソフトウェアのインストール時にOracle Universal Installerの「ノードの選択」ページで「ローカル」および非クラスタを選択できます。単一ノードのクラスタ(およびOracle RAC)のインストールを実行して、後でOracle RACオプションを無効にすることもできます。

次の手順に従って、このタイプのシングル・インスタンス・データベースをOracle RACまたはOracle RAC One Nodeデータベースに変換します。

1. 前の項の説明に従って、DBCAを使用してシングル・インスタンス・データベースの事前構成済イメージを作成します。手動で変換を実行するには、シングル・インスタンス・データベースを停止します。
2. ディレクトリを、Oracleホームのrdbmsディレクトリにあるlibサブディレクトリに変更します。
3. 次のコマンドを実行して、oracleバイナリに再度リンクします。

```
make -f ins_rdbms.mk rac_on
make -f ins_rdbms.mk ioracle
```

4. ノードをクラスタに追加します。すべてのノードがOracle ClusterwareおよびOracle RACで使用される共有記憶域

にアクセスできることを確認します。

関連項目

- DBCAを使用したシングル・インスタンス・データベースのイメージの作成
- Oracle Clusterware管理およびデプロイメント・ガイド

rconfigおよびOracle Enterprise Managerを使用して変換するための準備

rconfigまたはOracle Enterprise Managerを使用すると、シングル・インスタンス・データベースのインストールをOracle RACデータベースに簡単に変換できます。

rconfigは、コマンドライン・ユーティリティです。Oracle Enterprise Manager Cloud Controlのデータベース管理オプション(「クラスタ・データベースへの変換」)は、GUIベースの変換ツールです。次の項では、これらの変換ツールを使用する方法について説明します。

- Oracle RACデータベースへの変換の前提条件
- rconfigを使用したOracle RACへの変換時の構成の変更
- rconfigまたはOracle Enterprise Managerを使用したデータベースのOracle RACへの変換
- Oracle Enterprise Managerを使用したデータベースのOracle RACへの変換

ノート:

変換を実行する前には、既存のデータベースのバックアップを作成しておきます。大規模な変更を行う前にも、データベースのバックアップを作成してください。

Oracle RACデータベースへの変換の前提条件

データベースを変換する前に、Oracle Real Application Clusters (Oracle RAC)環境がこれらの前提条件を満たしている必要があります。

シングル・インスタンスのデータベースをOracle RACデータベースに変換する前に、Oracle RACデータベース・ノードを作成する各クラスタ・ノードで次の条件が満たされていることを確認します。

- Oracle Clusterware 19cがインストールおよび構成済で実行されている。
- Oracle RAC 19cソフトウェアがインストールされている。
- Oracleソフトウェアで、Oracle RACオプションが有効になっている。
- Oracle Cluster File SystemまたはOracle ASMのいずれかの共有記憶域がすべてのノードから使用可能でアクセスできる。
- oracleアカウント、またはOracleソフトウェアのインストールに使用されたユーザー・アカウントにユーザー等価関係が存在する。
- Oracle Enterprise Managerを使用する場合は、各ノードのOracle Management Agentが構成され、実行

中であり、クラスタおよびホスト情報とともに構成されている。

- 既存のデータベースをバックアップしている。

ノート:

Oracle RAC データベースでは、クラスタ化された Oracle ASM インスタンスを使用する必要があります。

rconfigを使用したOracle RACへの変換時の構成の変更

rconfigユーティリティを使用してシングル・インスタンス・データベースをOracle RACに変換すると、これらの変更が発生します。

- 変換時に、rconfigによってターゲットのOracle RACデータベースがアーカイブ・ログ・モードに設定され、データベースのアーカイブが有効になります。アーカイブ・ログ領域を使用しない場合は、変換の完了後にアーカイブ・ログを無効にできます。
- 共有記憶域タイプの値として「CFS」と入力し、シングル・インスタンス・データベース記憶域に対してクラスタ・ファイル・システムを使用している場合は、rconfigによって、データベース記憶域に対してOracle Managed Filesが使用されるように環境が変換され、データ・ファイルが共有記憶域の場所の下にあるサブディレクトリに配置されます。
- 変換時に、rconfigによって、データベース・ファイルが指定した共有の場所に移動され、Oracle Managed Filesを使用して構成されます。

  変換されたデータベースでOracle Managed Filesを使用しないようにするには、シングル・インスタンス・データベースのファイルを共有ファイル・システムに配置して、rconfigでこれらのファイルが移動されないように指定する必要があります。

rconfigまたはOracle Enterprise Managerを使用したデータベースのOracle RACへの変換

このリストでは、シングル・インスタンスのOracleデータベースをOracle RACデータベースに変換するシナリオを示します。

- シングル・インスタンスのOracle Database 19cデータベースを、このシングル・インスタンス・データベースと同じOracleホームから実行し、同じデータ・ファイルを使用するOracle RAC 19cデータベースに変換します。

  このシナリオでは、Oracle RACデータベース・ホームからrconfigユーティリティを実行するか、またはOracle Enterprise Manager Cloud Controlのシングル・インスタンスのデータベース・ターゲットでRACへの変換オプションを使用します。
- Oracle Database 19cより前のリリースのOracle Databaseを使用するシングル・インスタンス・データベースを、このシングル・インスタンス・データベースと同じOracleホームから実行し、同じデータ・ファイルを使用するOracle RAC 19cデータベースに変換します。

  このシナリオでは、Oracle Universal InstallerおよびDatabase Upgrade Assistant (DBUA)を使用して、シングル・インスタンス・データベースをOracle Database 19cに更新します。その後で、前述のシナリオで説明したようにrconfigまたはOracle Enterprise ManagerのRACへの変換オプションを使用します。
- シングル・インスタンスのOracle Database 19cを、このシングル・インスタンス・データベースとは異なるOracleホームから実行し、同じデータ・ファイルを使用するOracle RAC 19cデータベースに変換します。

このシナリオでは、ターゲットのデータベース・ホームでrconfigユーティリティを実行するか、またはOracle Enterprise Manager Cloud Controlのシングル・インスタンスのデータベース・ターゲットでRACへの変換オプションを使用します。プロンプトに従って、ファイル記憶域の場所を指定します。

ノート:

ターゲット・データベース・ホームおよびソース・データベース・ホームの両方に Oracle ホーム・ユーザーを指定する場合、ターゲット・データベース・ホームの Oracle ホーム・ユーザーと、ソース・データベース・ホームの Oracle ホーム・ユーザーは同じである必要があります。

- シングル・インスタンス・データベースが実行されているホストがOracle RACデータベースのノードではない環境で、シングル・インスタンスのOracle Database 19cを、異なるOracleホームから実行するOracle RAC 19cデータベースに変換します。

  このシナリオでは、シングル・インスタンスのデータベースのクローン・イメージを作成し、そのクローン・イメージをOracle RACデータベースが使用するノードであるホストに移動します。その後で、前述のシナリオで説明したようにrconfigまたはOracle Enterprise ManagerのRACへの変換オプションを使用します。

Oracle Enterprise Managerを使用したデータベースのOracle RACへの変換

Oracle Enterprise Manager Cloud Controlを使用して、シングル・インスタンス・データベースをOracle RACデータベースに変換できます。

この機能を使用するには、次のステップを実行します。

1. Oracle Enterprise Manager Cloud Controlにログインします。ホーム・ページで、「ターゲット」タブをクリックします。
2. 「ターゲット」ページで、「データベース」タブをクリックし、Oracle RACに変換するデータベースの「名前」列にあるリンクをクリックします。
3. 「データベース・ホーム」ページで、「可用性」メニューから「クラスタ・データベースへの変換」を選択します。
4. SYSDBA権限を持つデータベース・ユーザーSYSとして、変換するデータベースにログインし、「次へ」をクリックします。
5. 「クラスタ・データベースへの変換: クラスタ資格証明」ページで、oracleユーザーのユーザー名とパスワード、および変換するターゲット・データベースのパスワードを指定します。ターゲット・データベースでOracle ASMを使用している場合は、SYSASMユーザーとパスワードも指定して、「次へ」をクリックします。
6. 「ホスト」ページで、インストールしたOracle RACデータベースのクラスタ・メンバーにするクラスタ内のホスト・ノードを選択します。選択が完了したら、「次へ」をクリックします。
7. 「データベースへの変換: オプション」ページで、既存のリスナーとポート番号を使用するか、またはクラスタに新しいリスナーとポート番号を指定するかどうかを選択します。また、クラスタのクラスタ・データベース・インスタンスの接頭辞も指定します。

   情報の入力が終了したら、「次へ」をクリックします。または、情報の入力方法の決定についての情報が必要な場合は、「ヘルプ」をクリックします。
8. 「クラスタ・データベースへの変換: 共有記憶域」ページで、既存の共有記憶域領域を使用するオプションを選択するか、またはデータベース・ファイルを新しい共有記憶域の場所にコピーするオプションを選択します。また、既存の高速リ

カバリ領域を使用するか、またはOracle Databaseによって管理されたファイルを使用して、リカバリ・ファイルを新しい高速リカバリ領域にコピーするかどうかを決定します。

Oracle ASMを使用する場合、データ・ファイルとリカバリ・ファイルを別の障害グループに配置することをお薦めします。障害グループは、2つのディスク間で共有されているコントローラなどの共有ハードウェアまたは同じスピンドル上にある2つのディスクによって定義されます。2つのディスクで障害が発生したハードウェアを共有しており、両方のディスクが使用できなくなった場合、これらのディスクは同じ障害グループに属しています。Oracle ASMを使用しない場合、データ・ファイルとリカバリ・ファイルを別の場所(別個のOracle ASM障害グループなど)に配置して、ハードウェアの障害によって可用性が低下しないようにすることをお薦めします。

情報の入力が終了したら、「次へ」をクリックします。または、情報の入力方法の決定についての情報が必要な場合は、「ヘルプ」をクリックします。

9. 「クラスタ・データベースへの変換: 確認」ページで、選択したオプションを確認します。変換に進むには、「ジョブの発行」をクリックします。選択したオプションを変更するには、「戻る」をクリックします。変換を取り消す場合は、「取消」をクリックします。
10. 「確認」ページで、「ジョブの表示」をクリックし、変換の状態を確認します。

関連項目

- Oracle Databaseアップグレード・ガイド

rconfigを使用したデータベースのOracle RACへの変換

コマンドライン・ユーティリティrconfigを使用すると、ConvertToRAC.xmlファイルで提供する値に応じて、シングル・インスタンス・データベースをOracle RACデータベースに変換することも、Oracle RAC One Nodeデータベースに変換することもできます。この機能を使用するには、次のステップを実行します。

1. oracleユーザーで、$ORACLE_HOME/assistants/rconfig/sampleXMLsディレクトリに移動し、viなどのテキスト・エディタを使用してConvertToRAC.xmlファイルを開きます。
2. ConvertToRAC.xmlファイルを確認し、システムに必要なパラメータを変更します。XMLサンプル・ファイルには、ファイルの構成方法を説明するコメントが含まれています。XMLファイルにパスワードを入力しないでください。かわりに、rconfigユーティリティでパスワードの入力を要求するようにします。

警告:

変換が正常に完了することを確認するには、変換オプション Convert verify="ONLY"を設定して、テスト変換を実行します。

パラメータの変更が終了したら、file_name.xmlという形式の名前を付けてファイルを保存します。選択した名前をノートにとっておきます。

3. ディレクトリ$ORACLE_HOME/binに移動し、次のコマンドを使用してrconfigを実行します(ここで、input.xmlはステップ2で構成したXML入力ファイルの名前です)。

```
rconfig input.xml
```

たとえば、convert.xmlというXML入力ファイルを作成した場合は、次のコマンドを使用します。

```
$ ./rconfig convert.xml
```

rconfigユーティリティによって、必要なパスワードの入力を求めるプロンプトが表示されます。

ノート:

ConvertToRAC.xml ファイルの Convert verify オプションには、3 つのオプションがあります。

- Convert verify="YES": rconfig は変換を開始する前に、シングル・インスタンスから Oracle RAC に変換するための前提条件が満たされていることを確認するチェックを行います。
- Convert verify="NO": rconfig は前提条件のチェックを行わずに、変換を開始します。
- Convert verify="ONLY": rconfig は前提条件のチェックのみを行います。前提条件のチェックが完了しても変換は開始されません。

変換の実行に失敗した場合、次の手順を使用してリカバリを実行し、変換を再試行します。

1. DBCAのデータベースの削除オプションを使用して、データベースの削除を試行します。
2. ソース・データベースをリストアします。
3. 変換ログを確認し、rconfigでレポートされた問題で変換の失敗の原因になった可能性があるものを修正します。rconfigのログ・ファイルは、$ORACLE_BASE/cfgtoollogsのrconfigディレクトリの下に作成されます。
4. 変換を再試行します。

関連項目

- Oracle Databaseアップグレード・ガイド

ConvertToRAC用のrconfig XML入力ファイルの例

これら2つのXML ConvertToRAC入力ファイルのrconfigユーティリティの例を確認します。

ノート:

XML ファイルにパスワードを含めないでください。かわりに、rconfig ユーティリティでパスワードの入力を要求するようにします。

例15-1 ポリシー管理データベースのrconfig ConvertToRAC XMLファイルの例

この例では、Oracle ASMを使用する単一インスタンス・データベースをOracle ASM記憶域で(サーバー・プールを使用して)ポリシー管理のOracle RACデータベースに変換するXML入力ファイルを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<n:RConfig xmlns:n="http://www.example.com/rconfig"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.example.com/rconfig">
    <n:ConvertToRAC>
<!-- Verify does a precheck to ensure all pre-requisites are met, before the
 conversion is attempted. Allowable values are: YES|NO|ONLY -->
        <n:Convert verify="YES">
```

```
<!--Specify current OracleHome of non-rac database for SourceDBHome -->
              <n:SourceDBHome>/oracle/product/12.1.0/db_1</n:SourceDBHome>
<!--Specify OracleHome where the rac database should be configured. It can be same
 as SourceDBHome -->
              <n:TargetDBHome>/oracle/product/12.1.0/db_1</n:TargetDBHome>
<!--Specify SID of non-rac database and credential. User with sysdba role is
 required to perform conversion -->
              <n:SourceDBInfo SID="sales">
                <n:Credentials>
                  <n:User>sys</n:User>
                  <n:Role>sysdba</n:Role>
                </n:Credentials>
              </n:SourceDBInfo>
<!--Specify the list of existing or new server pools which are used by the
 Policy Managed Cluster Database. -->
              <n:ServerPoolList>
                <n:ExistingServerPool name="custom"/>
                <n:NewServerPool name="newpool" cardinality="2"/>
              </n:ServerPoolList>
<!--Specify RacOneNode along with servicename to convert database to RACOne
Node -->
              <!--n:RacOneNode  servicename="salesrac1service"/-->
<!--InstancePrefix is not required for Policy Managed database. If specified, it
 will be ignored. Instance names are generated automatically based on db_unique_
name for Policy Managed dababase.-->
<!-- Listener details are no longer needed starting 11.2. Database is registered
 with default listener and SCAN listener running from Oracle Grid Infrastructure
 home. -->
<!--Specify the type of storage to be used by rac database. Allowable values are
 CFS|ASM. The non-rac database should have same storage type. ASM credentials are
 no needed for conversion. -->
              <n:SharedStorage type="ASM">
<!--Specify Database Area Location to be configured for rac database.If this field
 is left empty, current storage will be used for rac database. For CFS, this field
 will have directory path. -->
                <n:TargetDatabaseArea>+ASMDG</n:TargetDatabaseArea>
<!--Specify Fast Recovery Area to be configured for rac database. If this field is
 left empty, current recovery area of non-rac database will be configured for rac
 database. If current database is not using recovery Area, the resulting rac
 database will not have a recovery area. -->
                <n:TargetFlashRecoveryArea>+ASMDG</n:TargetFlashRecoveryArea>
              </n:SharedStorage>
        </n:Convert>
    </n:ConvertToRAC>
</n:RConfig>
```

例15-2 管理者管理データベースのrconfig ConvertToRAC XMLファイルの例

この例では、Oracle ASMを使用する単一インスタンス・データベースを管理者管理のOracle RACデータベースに変換するXML入力ファイルを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<n:RConfig xmlns:n="http://www.example.com/rconfig"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.example.com/rconfig rconfig.xsd">
    <n:ConvertToRAC>
<!-- Verify does a precheck to ensure all pre-requisites are met, before the
```

```
 conversion is attempted. Allowable values are: YES|NO|ONLY -->
        <n:Convert verify="YES">
<!--Specify current OracleHome of non-rac database for SourceDBHome -->
              <n:SourceDBHome>/oracle/product/12.1.0/db_1</n:SourceDBHome>
<!--Specify OracleHome where the rac database should be configured. It can be same
 as SourceDBHome -->
              <n:TargetDBHome>/oracle/product/12.1.0/db_1</n:TargetDBHome>
<!--Specify SID of non-rac database and credential. User with sysdba role is
 required to perform conversion -->
              <n:SourceDBInfo SID="sales">
                <n:Credentials>
                  <n:User>sys</n:User>
                  <n:Role>sysdba</n:Role>
                </n:Credentials>
              </n:SourceDBInfo>
<!--Specify the list of nodes that should have rac instances running for the Admin
 Managed Cluster Database. LocalNode should be the first node in this nodelist.
 -->
              <n:NodeList>
                <n:Node name="node1"/>
                <n:Node name="node2"/>
              </n:NodeList>
<!--Specify RacOneNode along with servicename to convert database to RACOne
Node -->
<!--n:RacOneNode  servicename="salesrac1service"/-->
<!--Instance Prefix tag is optional starting with 11.2. If left empty, it is
 derived from db_unique_name.-->
              <n:InstancePrefix>sales</n:InstancePrefix>
<!-- Listener details are no longer needed starting 11.2. Database is registered
 with default listener and SCAN listener running from Oracle Grid Infrastructure
 home. -->
<!--Specify the type of storage to be used by rac database. Allowable values are
 CFS|ASM. The non-rac database should have same storage type. ASM credentials
are not needed for conversion. -->
              <n:SharedStorage type="ASM">
<!--Specify Database Area Location to be configured for rac database.If this field
 is left empty, current storage will be used for rac database. For CFS, this
field will have directory path. -->
                <n:TargetDatabaseArea>+ASMDG</n:TargetDatabaseArea>
<!--Specify Fast Recovery Area to be configured for rac database. If this field is
 left empty, current recovery area of non-rac database will be configured for rac
 database. If current database is not using recovery Area, the resulting rac
 database will not have a recovery area. -->
                <n:TargetFlashRecoveryArea>+ASMDG</n:TargetFlashRecoveryArea>
              </n:SharedStorage>
        </n:Convert>
    </n:ConvertToRAC>
</n:RConfig>
```

変換後のステップ

変換が完了したら、Oracle RAC環境の構成に関する推奨事項に従います。

- 前の章で説明したように、ロード・バランシングおよび透過的アプリケーション・フェイルオーバーを使用する際の推奨事項に従います。

- 『*Oracle Database*管理者ガイド』の説明に従って、ディクショナリ管理表領域ではなくローカル管理表領域を使用して、競合を軽減し、順序をOracle RACで管理します。
- 自動セグメント領域管理の使用方法については、『*Oracle Database*管理者ガイド』のガイドラインに従ってください。

Oracle RACでのバッファ・キャッシュおよび共有プールの容量に関する要件は、シングル・インスタンスのOracle Databaseでの要件よりもわずかに大きくなります。このため、バッファ・キャッシュのサイズを約10%、共有プールのサイズを約15%増加する必要があります。

関連項目

- 動的データベース・サービスによるワークロード管理
- ローカル管理表領域の使用について
- ローカル管理表領域のセグメント領域管理の指定

A サーバー制御ユーティリティのリファレンス

サーバー制御ユーティリティ(SRVCTL)を使用して、Oracle Real Application Clusters (Oracle RAC)構成情報を管理します。

ノート:

Oracle Grid Infrastructureの管理操作に固有のSRVCTLコマンドは、『Oracle Clusterware管理およびデプロイメント・ガイド』に記載されています
この付録の内容は次のとおりです。

- SRVCTLの使用方法
- 単一文字ではなくキーワードとしてのコマンド・パラメータの指定
- SRVCTLオブジェクトの値の文字セットおよび大文字小文字の区別
- SRVCTLを使用できるタスクのサマリー
- SRVCTLヘルプの使用方法
- SRVCTLの権限とセキュリティ
- 追加のSRVCTLトピック
- 非推奨のSRVCTLサブプログラムまたはコマンド
- SRVCTLのコマンド・リファレンス

SRVCTLの使用方法

SRVCTLは、デフォルトでクラスタの各ノードにそれぞれインストールされます。SRVCTLを使用するには、ノードのオペレーティング・システムにログインし、大/小文字が区別される構文を使用して、SRVCTLコマンドとそのパラメータを入力します。

- 管理しているデータベースのOracleホームから、現行のOracle Databaseリリースで提供されているSRVCTLのバージョンを使用します。SRVCTLのバージョンは、管理対象のオブジェクト(リスナー、Oracle ASMインスタンス、Oracle Database、Oracle Databaseインスタンスおよびオラクル・データベース・サービス)のバージョンと同じである必要があります。
- SRVCTLでは、同じオブジェクトに対する複数コマンドの同時実行はサポートされていません。したがって、各データベース、サービスまたは他のオブジェクトに対して、1つずつSRVCTLコマンドを実行します。
- カンマ区切りリストをSRVCTLコマンドの一部として指定する場合、リスト内の項目の間に空白をしないでください。たとえば:

  ```
  srvctl add database -serverpool "serverpool1,serverpool3"
  ```

  Windows環境でカンマ区切りリストを指定する場合は、リストを二重引用符("")で囲む必要があります。LinuxまたはUNIX環境で、カンマ区切りリストを二重引用符内で囲むことは可能ですが、二重引用符は無視されます。
- SRVCTLコマンドの入力時に、新しい行で入力を続ける場合、オペレーティング・システムの継続文字を使用できます。

Linuxでは、バックスラッシュ(¥)記号です。

- 出力が生成されないSRVCTLコマンドは、正常なコマンドです。完了(成功)時にすべてのSRVCTLコマンドがメッセージを返すわけではありません。しかし、SRVCTLコマンドが失敗した場合は、常にエラー・メッセージが返されます。
- SRVCTLは、成功時には0、失敗時には1、警告時には2を返します。start、stop、enable、disableなどの一部のコマンドでは、リクエストによって何も変更されない場合、警告に対して2が返されることがあります。つまり、コマンドのオブジェクトはすでに起動されているか、すでに停止されているか、すでに無効化されているかなどの場合です。警告の場合は、SRVCTLによって、すでに行われた処理に関するメッセージも出力されます。
- -evalパラメータは、複数のSRVCTLコマンドで使用できます。このパラメータを使用すると、システムに変更を加えることなく、コマンドの実行をシミュレートできます。SRVCTLによって戻される出力には、特定のコマンドを実行した場合の結果が示されます。たとえば、サーバーを再配置した場合に想定される結果を確認するには、次のようにします。

```
$ srvctl relocate server -servers "rac1" -eval -serverpool pool2
Database db1
     will stop on node rac1
     will start on node rac7
     Service mySrv1
          will stop on node rac1, it will not run on any node
     Service myServ2
          will stop on node rac1
          will start on node rac6
Server rac1
     will be moved from pool myPoolX to pool pool2
```

-evalパラメータは、次のコマンドで使用できます。

- srvctl add database
- srvctl add service
- srvctl add srvpool
- srvctl modify database
- srvctl modify service
- srvctl modify srvpool
- srvctl relocate server
- srvctl relocate service
- srvctl remove srvpool
- srvctl start database
- srvctl start service
- srvctl stop database
- srvctl stop service

単一文字ではなくキーワードとしてのコマンド・パラメータの指定

Oracle Database 12cより前のリリースでは、SRVCTLコマンドライン・インタフェースのパラメータとして単一文字が使用されていました。ただし、この方法では、SRVCTLコマンドで使用できる一意のパラメータの数に制限が発生します。Oracle Database 12cで導入されたSRVCTLコマンド・パラメータは、単一文字ではなく完全な単語になっています(-multicastportや-subdomainなど)。

下位互換性をサポートするために、単一文字パラメータと新しいキーワード・パラメータを組み合せて使用できます。新たに導入されたキーワード・パラメータは、単一文字パラメータと併用可能です。

ノート:

Oracle Database 12c 以降では、キーワード・パラメータが優先されるために、単一文字パラメータは非推奨になりました(異なる機能の実装に、コマンドに応じて同じ文字を使用することを回避するためです)。

該当する場合は、-help パラメータの後に-compatible パラメータを追加することによって、等価の単一文字を取得できます。

SRVCTLオブジェクトの値の文字セットおよび大文字小文字の区別

SRVCTLは、様々なタイプの多くのオブジェクトとやり取りします。文字セットと名前の長さの制限、およびオブジェクト名で大/小文字が区別されるかどうかは、オブジェクト・タイプによって異なります。

表A-1 SRVCTLオブジェクト名の文字列制限

オブジェクト・タイプ	文字セット制限	大/小文字の区別	最大長
db_domain	英数字、アンダースコア(_)およびシャープ記号(#)		128 文字
db_unique_name	英数字、アンダースコア(_)、シャープ記号(#)およびドル記号($)。最初の 8 文字はポリシー管理データベースのインスタンス名に使用されるため、これらは一意である必要があります。	いいえ	30 文字(ただし、最初の 8 文字は、同じクラスタの他のデータベースに対して一意である必要があります)
diskgroup_name	ディスク・グループのネーミングには、他のデータベース・オブジェクトのネーミングと同じ制限があります。 関連項目: データベース・オブジェクトのネーミング規則の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。	いいえ(すべての名前は大文字に変換されます)	

オブジェクト・タイプ	文字セット制限	大/小文字の区別	最大長
instance_name	英数字	プラットフォームに依存	15 文字
listener_name			
node_name		いいえ	
scan_name	最初の文字はアルファベットである必要があります	いいえ	
server_pool	英数字、アンダースコア(_)、シャープ記号(#)、ピリオド(.)およびドル記号。ただし、名前をピリオドで始めること、一重引用符(' ')を含むこと、または「Generic」または「Free」という名前にすることはできません。この 2 つの名前は、組込みサーバー・プールで予約されています。		250 文字
service_name			250 文字
volume_name	英数字(ダッシュ(-)で始めることはできません。最初の文字はアルファベットにする必要があります)。	いいえ	11 文字

SRVCTLを使用できるタスクのサマリー

SRVCTLを使用して、データベース、インスタンス、クラスタ・データベース、クラスタ・データベース・インスタンス、Oracle ASMインスタンスおよびディスク・グループ、サービス、リスナーまたは他のクラスタウェア・リソースを管理できます。

- データベース構成タスク

タスク	コマンド
データベース構成情報の追加、変更および削除	srvctl add database srvctl modify database srvctl remove database

タスク	コマンド
Oracle RAC データベース構成に対するインスタンスの追加または削除	srvctl add instance srvctl remove instance
データベース構成に対するサービスの追加または削除	srvctl add service srvctl remove service
データベース構成内のインスタンスおよびサービスの移動と、サービス構成の変更	srvctl relocate database srvctl relocate service srvctl modify instance srvctl modify service
データベース構成内のインスタンスまたはサービスの環境設定および解除	srvctl modify instance srvctl modify service
データベース構成内のクラスタ・データベース全体の環境設定および解除	srvctl setenv database srvctl unsetenv database

- 一般的なデータベース管理タスク

タスク	コマンド
データベースの起動と停止	srvctl start database srvctl stop database
データベース・インスタンスの起動と停止	srvctl start instance srvctl stop instance
データベース・サービスの開始、停止および再配置	srvctl start service srvctl stop service srvctl relocate service
データベース、データベース・インスタンスまたはデータベース・サー	srvctl status database srvctl status instance

タスク	コマンド
ビスのステータスの取得	srvctl status service

- ノード・レベル・タスク

タスク	コマンド
VIP の管理	srvctl add vip srvctl config vip srvctl disable vip srvctl enable vip srvctl getenv vip srvctl modify vip srvctl relocate vip srvctl remove vip srvctl setenv vip srvctl start vip srvctl status vip srvctl stop vip srvctl unsetenv vip
ノード・アプリケーションの管理	srvctl add nodeapps srvctl disable nodeapps srvctl enable nodeapps srvctl getenv nodeapps srvctl modify nodeapps srvctl remove nodeapps srvctl setenv nodeapps srvctl unsetenv nodeapps

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

SRVCTLヘルプの使用方法

この項では、SRVCTLの状況依存ヘルプを使用する方法について説明します。

すべてのSRVCTLコマンドに関するヘルプを表示するには、コマンドラインから次のように入力します。

```
srvctl -help
```

各SRVCTLコマンドのコマンド構文およびパラメータのリストを表示するには、コマンドラインから次のように入力します。

```
srvctl command (or verb) object (or noun) -help
```

SRVCTLで-helpを使用してコマンドのオンライン・ヘルプを要求すると、各パラメータの完全な単語が出力されます。該当する場合は、-helpパラメータの後に-compatibleパラメータを追加することによって、等価の単一文字を取得できます。たとえば:

```
$ srvctl config database -help -compatible
```

前述のコマンドを実行すると、srvctl config databaseコマンドの使用方法が出力され、すべてのパラメータが完全な単語として一覧表示され、その後に、該当する場合は等価の単一文字がカッコで囲まれて表示されます。

SRVCTLのバージョン・ナンバーを表示するには、次のように入力します。

```
$ srvctl -version
```

SRVCTLの権限とセキュリティ

SRVCTLを使用してデータベース構成を変更するには、管理するホームのソフトウェア所有者としてオペレーティング・システムにログインします。

たとえば、様々なユーザーがOracle DatabaseおよびOracle Grid Infrastructureをインストールした場合、データベースを管理するにはデータベース・ソフトウェア所有者(ora_dbなど)としてログインし、Oracle ASMインスタンスを管理するにはOracle Grid Infrastructureソフトウェア所有者(ora_asmなど)としてログインします。

OSDBAオペレーティング・システム・グループのメンバーであるユーザーは、データベースを起動および停止できます。Oracle ASMインスタンスを停止および起動するには、ユーザーがOSASMオペレーティング・システム・グループのメンバーであることが必要です。

リスナー、Oracle Notification Services、サービスなどのオブジェクトを作成または登録するには、Oracleホームのソフトウェア所有者としてオペレーティング・システムにログインする必要があります。そのOracleホームに作成または登録したオブジェクトは、Oracleホームの所有者のユーザー・アカウントで実行されます。データベースは、実行元になるデータベース・ホームのデータベース・インストール所有者として実行されます。

オブジェクト上でsrvctl add操作を実行するには、そのオブジェクトが実行されているホームのOracleアカウント所有者としてログインする必要があります。

一部のSRVCTLコマンドでコマンドを実行するには、LinuxシステムおよびUNIXシステムではrootでログインし、Windowsシ

ステムでは管理者権限を持つユーザーとしてログインする必要があります。この付録のコマンド例では、それらのコマンドの前にrootプロンプト(#)を付けてあります。

追加のSRVCTLトピック

- SRVCTLはリスナー、インスタンス、ディスク・グループ、ネットワークなど、Oracleから提供されるリソースの管理に使用し、CRSCTLはOracle Clusterwareおよびそのリソースの管理に使用します。

ノート:

Oracle が提供するリソース(ora という名前で始まるリソース)は、CRSCTL では直接操作しないでください。直接操作することはクラスタ構成に悪影響を与える場合があります。

- [Ctrl]キーを押しながら[C]キーを押すと、SRVCTLコマンドの実行をキャンセルできますが、その結果、構成データが破損される場合があります。

  この方法ではSRVCTLを中断しないでください。

非推奨のSRVCTLサブプログラムまたはコマンド

いくつかのSRVCTLコマンドおよびパラメータは、このリリースでは非推奨になりました。

すべてのSRVCTLコマンドの単一文字パラメータ

単一文字パラメータは、Oracle Database 12cでは非推奨になりました。

かわりに各パラメータの完全なキーワードを使用してください。引き続き単一文字パラメータが使用される従来のツールやスクリプトをサポートするために、現在のバージョンのSRVCTLでは、単一文字パラメータと完全なキーワード・パラメータの両方がサポートされています。

この付録のコマンド・リファレンスに、各SRVCTLコマンドのキーワードが示されています。表A-2に、非推奨になった単一文字パラメータを示します。

表A-2 SRVCTLコマンドで非推奨になった単一文字パラメータ

単一文字	詳細名	値	説明	関連コマンド
A	address	{VIP_name | IP}/netmask/[if1[|if2...]]	ノード・アプリケーションの VIP アドレス指定	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
a	all		その種のすべてのリソース	srvctl config database 共通

単一文字	詳細名	値	説明	関連コマンド
a	diskgroup	diskgroup_list	Oracle ASM ディスク・グループのカンマ区切りリスト	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
a	detail		詳細な構成情報の出力	共通
a	available	available_list	使用可能インスタンスのカンマ区切りリスト	サービスおよびサーバー・プールのコマンド
a	abort		失敗したオンライン再配置の停止	データベースの再配置
a	viponly		VIP 構成の表示	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
B	rlbgoal	{NONE| SERVICE_TIME| THROUGHPUT}	サービスのランタイム・ロード・バランシングの目標	サービスおよびサーバー・プールのコマンド
c	currentnode	current_node	サービスの再配置元となるノードの名前	サービスおよびサーバー・プールのコマンド
c	cardinality	{UNIFORM| SINGLETON}	サービスをサーバー・プール内のすべてのアクティブ・サーバーで実行するか(UNIFORM)、1 台のサーバーでのみ実行するか(SINGLETON)	サービスおよびサーバー・プールのコマンド
c	dbtype	type	データベースのタイプ: Oracle RAC One Node、Oracle RAC またはシングル・インスタンス	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
d	db または database	db_unique_name	データベースの一意の名前	共通
d	device	volume_device	ボリューム・デバイスのパス	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマ

単一文字	詳細名	値	説明	関連コマンド
				ンド
d	domain		GNS から供給されるサブドメインの表示	OC4J、ホーム、CVU および GNS のコマンド
e	emport	em_port_number	Oracle Enterprise Manager のローカル・リスニング・ポート	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
e	failovertype	{NONE|SESSION BASIC|TRANSACTION}	サービスのフェイルオーバー・タイプ	サービスおよびサーバー・プールのコマンド
e	server	server_list	Oracle RAC One Node データベースの候補サーバーのリスト	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
f	force		強制削除	共通
g	diskgroup	diskgroup_name	ディスク・グループ名	ファイル・システム、ディスクグループのコマンド
g	gsdonly		GSD 構成の表示	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
g	serverpool	server_pool_name server_pool_list	サーバー・プール名 データベース・サーバー・プール名のカンマ区切りリスト	サービスおよびサーバー・プールのコマンド データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
h	help			共通
i	重要度	number	サーバー・プールの重要度を	サービスおよびサーバー・プール

単一文字	詳細名	値	説明	関連コマンド
			表す番号	のコマンド
i	instance	instance_name instance_list	管理者管理 Oracle RAC One Node データベースのインスタンス名の接頭辞 インスタンス名のカンマ区切りリスト	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
I	ip	ip_address	GNS がリスニングする VIP アドレス	OC4J、ホーム、CVU および GNS のコマンド
i	oldinst	instance_name	元のインスタンス名	サービスおよびサーバー・プールのコマンド
i	scannumber	scan_ordinal _number	SCAN 用の IP アドレスの序数	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
i	vip	vip_name or "vip_name_list"	VIP 名	ノード・アプリケーション、GNS、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
j	acfspath	acfs_path_list	データベースの依存性を設定する Oracle ACFS パスのカンマ区切りリスト	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
j	clbgoal	{SHORT|LONG}	サービスの接続時ロード・バランシングの目標	サービスおよびサーバー・プールのコマンド
k	netnum	network_number	ネットワーク番号	サービスおよびサーバー・プールのコマンド ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド

単一文字	詳細名	値	説明	関連コマンド
				OC4J、ホーム、CVU および GNS のコマンド
l	list		GNS のすべてのレコードのリスト	OC4J、ホーム、CVU および GNS のコマンド
l	listener	listener_name	リスナーの名前	ASM のコマンド
l	loglevel	log_level	GNS が実行するロギング・レベル(0 から 6)の指定	OC4J、ホーム、CVU および GNS のコマンド
l	min	number	サーバー・プールの最小サイズ	サービスおよびサーバー・プールのコマンド
l	onslocalport	port_number	ローカル・クライアント接続用の Oracle Notification Service リスニング・ポート	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
l	role	service_role	二重引用符(””)で囲んだサーバー・ロールのカンマ区切りリスト(各ロールは PRIMARY、PHYSICAL_STANDBY、LOGICAL_STANDBY または SNAPSHOT_STANDBY のいずれか)	サービスおよびサーバー・プールのコマンド
m	domain	domain_name	データベースのドメイン	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
m	failovermethod	{NONE|BASIC}	サービスのフェイルオーバー・メソッド	サービスおよびサーバー・プールのコマンド
m	multicastpost		GNS デーモンがマルチキャスト・リクエストをリスニングしているポート	OC4J、ホーム、CVU および GNS のコマンド

単一文字	詳細名	値	説明	関連コマンド
m	path	mountpoint_path	マウントポイント・パス	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
n	name		特定のアドレスを使用したGNS による名前の通知	OC4J、ホーム、CVU および GNS のコマンド
n	node	node_name	特定のノードの名前	共通
n	nodes	node_list	ノード名のカンマ区切りリスト	ファイル・システムのコマンド
n	dbname	database_name	データベース名(DB_NAME)、-db パラメータで指定する一意の名前と異なる場合	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
n	scanname	scan_name	完全修飾 SCAN 名(ドメインを含む)	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
n	servers	server_list	候補としてのサーバー名のカンマ区切りリスト	サービスおよびサーバー・プールのコマンド
n	targetnode	node_name	サービスの再配置先となるノードの名前	サービスおよびサーバー・プールのコマンド
o	oraclehome	oracle_home	$ORACLE_HOME パス	データベースのコマンド
p	endpoints	[TCP:]port_number[/IPC:key][/NMP:pipe_name][/TCPS:s_port][/SDP: port]	SCAN リスナー・エンドポイント	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
p	port		GNS デーモンが DNS サーバーとの通信に使用するポート	OC4J、ホーム、CVU および GNS のコマンド

単一文字	詳細名	値	説明	関連コマンド
p	rmiport	port_number	OC4J RMI ポート番号	OC4J、ホーム、CVU および GNS のコマンド
P	tafpolicy	{NONE|BASIC}	TAF ポリシーの指定	サービスおよびサーバー・プールのコマンド
p	spfile	spfile_location	サーバー・パラメータ・ファイルのパス	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
q	notification	{TRUE|FALSE}	FAN が OCI 接続に対して有効かどうか	サービスのコマンド
q	query		GNS への、名前に属しているレコードの問合せ	OC4J、ホーム、CVU および GNS のコマンド
r	preferred	preferred_list	優先インスタンスのカンマ区切りリスト	サービスおよびサーバー・プールのコマンド
r	onsremoteport	port_number	リモート・ホストからの接続用の Oracle Notification Service リスニング・ポート	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
r	relocate		VIP の再配置	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
r	revert		管理者管理 Oracle RAC One Node データベースの候補サーバー・リストからの、失敗したオンライン再配置リクエストのターゲット・ノードの削除	データベースの再配置
r	role	role_type	スタンバイ・データベースのロール: PRIMARY、PHYSICAL_STANDBY、LOGICAL_STANDBY または	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマ

単一文字	詳細名	値	説明	関連コマンド
			SNAPSHOT_STANDBY	ンド
s	onsonly		Oracle Notification Service デーモン構成の表示	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
s	skip		ポートの確認のスキップ	リスナー、SCAN および SCAN リスナー。
s	statfile	file_name	前に実行した srvctl stop home コマンドによって作成された state_file のファイル・パス	OC4J、ホーム、CVU および GNS のコマンド
s	status		GNS のステータスの表示	OC4J、ホーム、CVU および GNS のコマンド
S	subnet	subnet/net_mask/[if1[|if2...]]	ネットワークのネットワーク・アドレス指定	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
s	service	service_name service_name_list	サービスの名前 サービス名のカンマ区切りリスト	サービスおよびサーバー・プールのコマンド
s	startoption	start_options	データベースの起動オプション (mount、open、read only)	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
t	checkinterval	time_interval	チェック間隔(分)	OC4J、ホーム、CVU および GNS のコマンド
t	edition	edition_name	サービスの初期セッション・エディション	サービスおよびサーバー・プールのコマンド

単一文字	詳細名	値	説明	関連コマンド
t	envs	"name_list"	環境変数のリスト	共通
t	namevals	"name= value,..."	環境変数の名前および値	共通
T	nameval	"name=value"	単一環境変数の名前と値	共通
t	update	instance_name	新しいインスタンス名	サービスおよびサーバー・プールのコマンド
t	remoteservers	host_name[:port_number][,host_name[:port_number]...]	このクラスタの外部にあるOracle Notification Service デーモン用のリモート・ホスト名とポート番号のペアのリスト	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
t	stopoption	stop_options	データベースの停止オプション(NORMAL、TRANSACTIONAL、IMMEDITATE または ABORT)	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
t	toversion	target_version	ダウングレード先のバージョン	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
u	max	number	サーバー・プールの最大サイズ	サービスおよびサーバー・プールのコマンド
u	nettype	network_type	ネットワーク・サーバー・タイプ(STATIC、DHCP または MIXED)	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
u	newinst		サービス構成への新しいインスタンスの追加	サービスのコマンド
u	update		SCAN VIP の数に合わせた SCAN リスナーの更新	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコ

単一文字	詳細名	値	説明	関連コマンド
				マンド
u	user	oracle_user	ファイル・システムのマウントおよびアンマウントが許可されたOracle ユーザーまたはその他のユーザー	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
v	verbose		詳細出力	共通
v	volume	volume_name	ボリュームの名前	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
V	versions			共通
w	failoverdelay	number	フェイルオーバーの遅延	サービスおよびサーバー・プールのコマンド
w	nettype	network_type	ネットワーク・サーバー・タイプ(STATIC、DHCP または MIXED)	ノード・アプリケーション、VIP、ネットワーク、リスナー、SCAN VIP および SCAN リスナーのコマンド
w	timeout	timeout	オンライン再配置のタイムアウト(分)	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド
x	dtp	{TRUE | FALSE}	分散トランザクション処理を有効化するかどうか	サービスおよびサーバー・プールのコマンド
x	node	node_name	ノード名(このパラメータは非クラスタ・データベースでのみ使用します)	共通
y	noprompt		確認プロンプトの抑止	共通
y	policy	{AUTOMATIC | MANUAL}	リソースの管理ポリシー	データベース、インスタンス、Oracle ASM、ディスク・グルー

単一文字	詳細名	値	説明	関連コマンド
				プ、ファイル・システム、サービスおよびサーバー・プールのコマンド
z	failoverretry	number	フェイルオーバー再試行回数	サービスおよびサーバー・プールのコマンド
z	rmdepondisk		ディスク・グループへのデータベースの依存性の削除	データベース、インスタンス、Oracle ASM、ディスク・グループおよびファイル・システムのコマンド

その他のSRVCTLコマンドおよびパラメータ

次のコマンド・パラメータは、このリリースでは非推奨になりました。

表A-3 SRVCTLで非推奨になったコマンドおよびパラメータ

コマンド	非推奨のパラメータ
srvctl modify asm	-node node_name
srvctl modify instance	-z かわりに、-node オプションを使用し、その値を""に設定してください。
srvctl modify gns	[-ip ip_address] [-advertise host_name -address address] [-delete host_name -address address] [-createalias name -alias alias] [-deletealias alias] かわりに、srvctl update gns コマンドを使用してください。
srvctl * oc4j	名詞の oc4j は非推奨になり、qosmserver に置き換えられました。SRVCTL は、サポートが終了するまで名詞の oc4j を受け入れます。
srvctl add service	PRECONNECT オプションの-tafpolicy パラメータは非推奨です。
srvctl modify service	-failovermethod {NONE | BASIC}は、非推奨です。 PRECONNECT オプションの-tafpolicy パラメータは非推奨です。

SRVCTLのコマンド・リファレンス

Oracle RAC環境で使用するSRVCTLコマンドの包括的なリスト。

SRVCTLのコマンド、オブジェクト名およびパラメータでは、大/小文字が区別されます。データベース、インスタンス、リスナーおよびサービスの名前は、大/小文字が区別されず、そのまま保持されます。LISTENERとlistenerのように、大/小文字のみが異なるリスナー名は作成できません。SRVCTLでは次のコマンド構文を使用します。

```
srvctl command object [parameters]
```

このSRVCTL構文の各要素の意味は次のとおりです。

- commandは、start、stop、removeなどの動詞です。
- object (名詞とも呼ばれる)は、SRVCTLがコマンドを実行するターゲットまたはオブジェクト(データベースやインスタンスなど)です。オブジェクトの短縮形も使用できます。
- parametersは、コマンドの追加パラメータを使用できるようにすぐ前のコマンドの組合せの使用範囲を拡大します。たとえば、-instancesパラメータは、優先インスタンス名のカンマ区切りリストが後に続くことを示し、-instanceパラメータでは、名前のリストではなく1つの値のみが許可されます。カンマ区切りリストの項目の間に空白を使用しないでください。

ノート:

Windows でカンマ区切りリストを指定する場合は、リストを二重引用符(“”)で囲む必要があります。

表A-4に、SRVCTLコマンドのobjectの部分に使用できるキーワードを示します。各オブジェクト・キーワードとして、完全な名前または短縮形のいずれかを使用できます。「目的」列に、オブジェクトとそのオブジェクトに実行できるアクションを記述します。

表A-4 オブジェクト・キーワードおよび短縮形

オブジェクト	キーワード	用途
データベース	database	データベースに対する追加、変更、環境変数管理、構成表示、有効化、無効化、起動、停止、ステータス取得の他、データベース構成情報のアップグレード、ダウングレードおよび削除。
インスタンス	instanceinst	データベース・インスタンスに対する追加、変更、有効化、無効化、起動、停止、ステータス取得および削除。
リスナー	listenerlsnr	リスナーに対する追加、変更、環境変数管理、構成表示、有効化、無効化、起動、停止、ステータス取得および削除
ネットワーク	network	デフォルト以外のネットワークの追加、変更、構成表示および削除 ノート: また、ノード・アプリケーション・オブジェクトと、config および modify コマンドは、デフォルトのネットワークを管理します。

オブジェクト	キーワード	用途
ノード・アプリケーション	nodeapps	ノード・アプリケーションに対する追加、変更、環境変数管理、構成表示、有効化、無効化、起動、停止、ステータス取得および削除
Oracle Notification Service	ons	Oracle Restart の Oracle Notification Service インスタンスのみに対する追加、構成、有効化、起動、ステータス取得、停止、無効化および削除
単一クライアント・アクセス名 (SCAN)	scan	SCAN VIP に対する追加、構成表示、変更、有効化、無効化、起動、停止、再配置、ステータス取得および削除
SCAN リスナー	scan_listener	SCAN リスナーに対する追加、構成表示、変更、有効化、無効化、起動、停止、再配置、ステータス取得および削除
サービス	service	サービスに対する追加、変更、構成表示、有効化、無効化、起動、停止、ステータス取得、再配置および削除
仮想 IP	vip	VIP に対する追加、環境変数管理、構成表示、有効化、無効化、起動、停止、ステータス取得および削除

ノート:

Oracle Grid Infrastructure管理操作に固有のSRVCTLコマンドについては、CWADD SRVCTLコマンド・リファレンスを参照してください

srvctl add database

データベース構成をOracle Clusterwareに追加します。

構文

```
srvctl add database -db db_unique_name [-eval]
     -oraclehome oracle_home [-node node_list] [-domain domain_name]
     [-spfile spfile] [-pwfile password_file_path]
     [-dbtype  {RACONENODE | RAC | SINGLE} [-server "server_list"]]
     [-instance instance_name] [-timeout timeout]]
     [-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY"]
     [-startoption start_options] [-stopoption stop_options] [-dbname db_name]
     [-acfspath "acfs_path_list"] [-policy {AUTOMATIC | MANUAL | NORESTART}]
     [-serverpool "server_pool_list" [-pqpool "pq_pool_list"]]
     [-diskgroup "disk_group_list"] [-css_critical {yes | no}] [-cpucount cpu_count]
     [-memorytarget memory_target] [-maxmemory max_memory] [-cpucap cpu_cap] [-defaultnetnum
network_number] [-verbose]
```

パラメータ

表A-5 srvctl add databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮定的に評価できます。 ノート: このパラメータは、ポリシー管理データベースでのみ使用できます。
-oraclehome oracle_home	Oracle Database のホーム・ディレクトリのパスを指定します。
-node node_list	単一のノード名、または非クラスタ(シングル・インスタンス) Oracle データベースを登録するノード名のリストをカンマ区切りで指定します。Oracle Database 19c リリース更新(19.7)以降、複数のクラスタ・ノード上に Standard Edition 高可用性データベースを登録できます。 ノート: ポリシー管理のシングル・インスタンス・データベースを作成する際、このパラメータは Oracle Clusterware でのみ使用でき、-serverpool パラメータと併用して使用できます。
-domain db_domain	データベースのドメインを指定します。 ノート: データベースに対して DB_DOMAIN 初期化パラメータを設定している場合は、このパラメータを使用する必要があります。
-spfile spfile	データベース・サーバー・パラメータ・ファイルのパス名を指定します。
-pwfile password_file_path	パスワード・ファイルの場所へのフル・パスを入力します。
-dbtype {RACONENODE | RAC | SINGLE}	追加するデータベースのタイプを指定します(Oracle RAC One Node、Oracle RAC、またはシングル・インスタンス)。-node node_name パラメータを指定しない場合のデフォルトは RAC で、-type パラメータのデフォルトは SINGLE です。
-server server_list	Oracle RAC One Node データベースの候補サーバーのリスト。

パラメータ	説明
	ノート: このパラメータは、管理者管理 Oracle RAC One Node データベースでのみ使用できます。Oracle RAC One Node データベースがポリシー管理である場合、このパラメータは使用できません。
-instance instance_name	Oracle RAC One Node データベースのインスタンス名の接頭辞を指定します。このパラメータのデフォルト値は、データベースのグローバルな一意の名前の最初の 12 文字です。 ノート: このパラメータは、管理者管理 Oracle RAC One Node データベースでのみ使用できます。Oracle RAC One Node データベースがポリシー管理である場合、このパラメータは使用できません。
-timeout timeout	Oracle RAC One Node データベースのオンライン・データベース再配置タイムアウトを分単位で指定します。デフォルト値は 30 です。
-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY}	Oracle Data Guard 構成でのデータベースのロールを指定します。デフォルトは PRIMARY です。
-startoption start_options	データベースの起動オプション(OPEN、MOUNT、NOMOUNT など)デフォルト値は OPEN です。 ノート: ● 起動オプションに複数の語を指定する場合(read only、read write など)、語をスペースで区切り、二重引用符(″″)で囲みます。たとえば″read only″とします。 ● Oracle Data Guard 構成でスイッチオーバーを実行する場合、プライマリ・データベースになるスタンバイ・データベースの-startoption は、スイッチオーバーの後では常に OPEN に設定されます。
-stoption stop_options	データベースの停止オプションを指定します(NORMAL、TRANSACTIONAL、IMMEDIATE、ABORT など)。
-dbname db_name	データベースの名前を指定します(-db パラメータで指定した一

パラメータ	説明
	意の名前と異なる場合)。
-acfspath "acfs_path_list"	データベースの依存性が設定された、二重引用符("")で囲まれた単一の Oracle ACFS パスまたは Oracle ACFS パスのカンマ区切りリスト このパラメータは、データベースが ORACLE_HOME ファイル・システムとは異なるファイル・システムで ORACLE_BASE を使用する場合など、ORACLE_HOME 以外の Oracle ACFS ファイル・システムへの依存性を作成する場合に使用します。
-policy {AUTOMATIC | MANUAL | NORESTART}	データベースの管理ポリシーを指定します。 ● AUTOMATIC(デフォルト): データベースは、データベース・ホスト・コンピュータの再起動時に前回の実行状態(起動または停止)へ自動的に戻ります。 ● MANUAL: データベース・ホスト・コンピュータの再起動時にデータベースは自動的には再起動されません。MANUAL に設定しても、Oracle Clusterware は、実行中のデータベースを監視し、障害発生時にデータベースを再起動します。 ● NORESTART: MANUAL 設定と同様に、データベース・ホスト・コンピュータの再起動時にデータベースは自動的には再起動されません。ただし、NORESTART 設定では、障害が発生しても、データベースを再起動することはありません。
-serverpool "server_pool_name" [-pqpool "pq_pool_name"]	データベースの配置の制御に使用するサーバー・プールの名前を指定します。このパラメータを指定しない場合、デフォルトで Generic サーバー・プールが使用されます。 また、必要に応じて、データベースで使用されるパラレル問合せサーバー・プールの名前を指定することもできます。 ノート: ● このパラメータは、Oracle Clusterware でのみ使用できます。このパラメータと-node パラメータは併用できますが、サーバー・プールは MAX_SIZE=1 とし、1 台のみのサーバー(-node で指定)を構成しておく必要があ

パラメータ	説明
	ります。 ● サーバー・プールを追加した後、srvctl add service コマンドを使用してサービスをサーバー・プールに割り当てることができます。
-diskgroup "disk_group_list"	Oracle ASM ディスク・グループのカンマ区切りリストを指定します(データベースで Oracle ASM 記憶域を使用している場合)。
-css_critical {YES | NO}	このパラメータを YES に設定することにより、サービスに重みを追加できます。クラスタ内のノードに障害が発生した場合、Oracle Clusterware は最も重みの小さいノードを削除して、クリティカルなサービスを使用可能な状態に保ちます。 ノート: このパラメータは、管理者管理ノードでのみ使用できます。ノードがポリシー管理ノードになった場合、その時点でこのパラメータは適用されなくなります。
-cpucount cpu_count	CPU の数を指定します。デフォルトの値は 0。
-memorytarget memory_target	データベースに割り当てるターゲット・メモリー(MB)を指定します。デフォルトは 0 です。
-maxmemory max_memory	リソースに割り当てる最大メモリー(MB)を指定します。-memorytarget を指定して-maxmemory を指定しない場合、-maxmemory はデフォルト値の 0 になります。-memorytarget が-maxmemory 以下であれば、-maxmemory と-memorytarget は両方とも検証されます。
-cpucap cpu_cap	データベースに必要なワークロード CPU の最大使用率を 1 から 100 までのパーセンテージで指定します。デフォルトは 0 です。
-defaultnetnum network_number	サービスの追加時にネットワーク番号を指定しない場合にデフォルトで設定されるサービスのネットワーク番号(整数)を指定します。この番号は、ネットワークを追加したときに指定した-netnum パラメータの値と一致している必要があります。

例

次のコマンド例は、ポリシー管理Oracle RACデータベースを追加します。

```
$ srvctl add database -db crm -oraclehome /u01/oracle/product/12c/mydb
    -domain example.com -spfile +diskgroup1/crm/spfilecrm.ora
    -role PHYSICAL_STANDBY -startoption MOUNT -dbtype RAC -dbname crm_psd
    -policy MANUAL -serverpool "svrpool1,svrpool2" -diskgroup "dgrp1,dgrp2"
```

次のコマンド例は、管理者管理データベースを追加します。

```
$ srvctl add database -db crm -oraclehome /u01/oracle/product/12c/mydb
-domain example.com
```

srvctl config database

Oracle RACデータベースの構成、またはOracle Clusterwareに登録されたすべての構成済データベースを表示します。

構文

```
srvctl config database [-db db_unique_name] [-all] [-verbose]
```

パラメータ

表A-6 srvctl config databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前。このパラメータを指定しない場合は、すべてのデータベース・リソースの構成が表示されます。
-all	詳細な構成情報の出力。
-verbose	冗長出力を表示します。

例

このコマンドによって、次のような出力が返されます。

```
$ srvctl config database -d main4
Database unique name: main
Database name:
Oracle home: /ade/mjkeenan_main4/oracle
Oracle user: mjkeenan
Spfile:
Password file:
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups:
Mount point paths:
Services: test
Type: RAC
```

```
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: main41,main42
Configured nodes: mjkeenan_main4_0,mjkeenan_main4_1
CSS critical: no
CPU count: 0
Memory target : 0
Maximum memory: 0
CPU cap: 0
Database is administrator managed
```

srvctl convert database

Oracle RAC One Nodeデータベースへ、またはOracle RAC One Nodeデータベースからデータベースを変換します。

構文

次の構文モデルのいずれかとともに、このコマンドを使用します。

```
srvctl convert database -db db_unique_name -dbtype RACONENODE
   [-instance instance_name] [-timeout timeout]
srvctl convert database -db db_unique_name -dbtype RAC [-node node_name]
```

パラメータ

表A-7 srvctl convert databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。 ノート: 非クラスタ・データベースを指定すると、このコマンドから、rconfig を使用して非クラスタ・データベースを Oracle RAC または Oracle RAC One Node に変換するように求めるエラーが戻されます。
-dbtype RACONENODE | RAC	変換するデータベースのタイプ(Oracle RAC One Node または Oracle RAC)を指定します。 ノート: 進行中または失敗したオンライン・データベース再配置がある場合は、先にこのオンライン・データベース再配置を完了または強制終了してから、このコマンドを再実行するように求めるエラーが戻されます。
-instance instance_name	オプションで、Oracle RAC One Node データベースのインスタンス名の接頭辞を指定できます。このパラメータのデフォルト値は、データベースのグローバルな一意の名前の最初の 12 文字です。

パラメータ	説明
	ノート: ● このパラメータは、Oracle RAC データベースから Oracle RAC One Node データベースに変換する場合にのみ使用できます。 ● 変換後のインスタンスをオンラインにするには、srvctl stop/start database コマンドを使用してデータベースを再起動する必要があります。
-timeout timeout	オプションで、Oracle RAC One Node データベースのオンライン・データベース再配置タイムアウトを分単位で指定できます。デフォルト値は 30 です。
-node node_name	オプションで、管理者管理 Oracle RAC データベースのノードの名前を指定できます。デフォルトは、最初の候補です。 ノート: ノード名を指定しない場合、またはデータベースが実行されてないノード名を指定した場合、このコマンドから、正しいノードを指定するように求めるエラーが戻されます。

例

次に、このコマンドの例を示します。

```
$ srvctl convert database -db myDB -dbtype RACONENODE -instance myDB3
```

srvctl disable database

実行中のデータベースを無効化します。

データベースがクラスタ・データベースの場合、そのインスタンスも無効化されます。

構文

```
srvctl disable database -db db_unique_name [-node node_name]
```

パラメータ

表A-8 srvctl disable databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	無効化するデータベースの名前を指定します。

パラメータ	説明
-node node_name	必要に応じて、データベースを無効化するノードを指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、データベースmydb1を無効化します。

```
$ srvctl disable database -db mydb1
```

srvctl downgrade database

データベースとそのサービスの構成を、現行バージョンから特定の下位バージョンにダウングレードします。

構文

```
srvctl downgrade database -db db_unique_name -oraclehome Oracle_home
  -targetversion to_version
```

パラメータ

表A-9 srvctl downgrade databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	ダウングレードするデータベースの一意の名前を指定します。
-oraclehome Oracle_home	Oracle ホームへのパスを指定します。
-targetversion to_version	ダウングレード先のデータベース・バージョンを指定します。

srvctl enable database

クラスタ・データベースとそのインスタンスを有効化します。

構文

```
srvctl enable database -db db_unique_name [-node node_name]
```

パラメータ

表A-10 srvctl enable databaseコマンドのパラメータ

パラメータ	説明

パラメータ	説明
-db db_unique_name	有効化するデータベースの一意の名前を指定します。
-node node_name	必要に応じて、有効化するデータベース・リソースが存在するノードの名前を指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、mydb1という名前のデータベースを有効化します。

```
$ srvctl enable database -db mydb1
```

srvctl getenv database

データベースに関連付けられている環境変数の値を表示します。

構文

```
srvctl getenv database -db db_unique_name [-envs "name_list"]
```

パラメータ

表A-11 srvctl getenv databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	環境変数の値を表示するデータベースの一意の名前を指定します。
-envs "name_list"	必要に応じて、値を二重引用符("")で囲んだ特定の環境変数の名前のカンマ区切りリストを指定できます。 このパラメータを使用しない場合、SRVCTL によって、データベースに関連付けられているすべての環境変数の値が表示されます。

例

次の例では、crmという名前のデータベースの環境構成を表示します。

```
$ srvctl getenv database -db crm
```

srvctl modify database

データベースの構成を変更します。

構文

```
srvctl modify database -db db_unique_name [-dbname db_name]
     [-instance instance_name] [-oraclehome oracle_home] [-user user_name]
     [-server "server_list"] [-timeout timeout] [-domain db_domain]
     [-spfile spfile] [-pwfile password_file_path]
     [-role {PRIMARY|PHYSICAL_STANDBY|LOGICAL_STANDBY|SNAPSHOT_STANDBY}]
     [-startoption start_options] [-stopoption stop_options]
     [-startconcurrency start_concurrency] [-stopconcurrency stop_concurrency]
     [-policy {AUTOMATIC | MANUAL | NORESTART | USERONLY}]
     [-serverpool "server_pool_name"] [-node node_list]
     [-pqpool pq_server_pool] [{-diskgroup "diskgroup_list" | -nodiskgroup}]
     [-acfspath "acfs_path_list"] [-css_critical {yes | no}]
     [-cpucount cpu_count [-overridepools overridepool_list]]
     [-memorytarget memory_target] [-maxmemory max_memory]
     [-defaultnetnum network_number] [-disabledreason {DECOMMISSIONED}]
     [-force] [-eval] [-verbose]
```

パラメータ

表A-12 srvctl modify databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前。
-dbname db_name	データベースの名前(-db パラメータで指定する一意の名前と異なる場合)
-instance instance_name	インスタンス名の接頭辞。このパラメータは、管理者管理 Oracle RAC One Node データベースで必要になります。
-oraclehome oracle_home	Oracle Database のホーム・ディレクトリのパス。
-user user_name	Oracle ホーム・ディレクトリを所有するユーザーの名前。 ノート: -user パラメータを指定する場合は、このコマンドを権限付きモードで実行する必要があります。
-server server_list	Oracle RAC One Node データベースの候補サーバーのリスト。 ノート: このパラメータは、管理者管理 Oracle RAC One Node データベースでのみ使用できます。Oracle RAC One Node データベースがポリシー管理である場合、このパラメータ

パラメータ	説明
	は使用できません。
-timeout timeout	Oracle RAC One Node データベースのオンライン・データベース再配置タイムアウト(分単位)。デフォルト値は 30 です。
-domain db_domain	データベースのドメイン。 ノート: データベースに対して DB_DOMAIN 初期化パラメータを設定している場合は、このパラメータを使用する必要があります。
-spfile spfile	データベース・サーバー・パラメータ・ファイルのパス名。
-pwfile password_file_path	パスワード・ファイルの場所へのフル・パスを入力します。
-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY}	Oracle Data Guard 構成でのデータベースのロール。デフォルトは PRIMARY です。
-startoption start_options	データベースの起動オプション(OPEN、MOUNT、NOMOUNT など)デフォルト値は OPEN です。 ノート: ● 起動オプションに複数の語を指定する場合(read only、read write など)、語をスペースで区切り、二重引用符(""")で囲みます。たとえば"read only"とします。 ● Oracle Data Guard 構成でスイッチオーバーを実行する場合、プライマリ・データベースになるスタンバイ・データベースの-startoption は、スイッチオーバーの後では常に OPEN に設定されます。
-stopoption stop_options	データベースの停止オプション(NORMAL、TRANSACTIONAL、IMMEDIATE、ABORT など)
-startconcurrency start_concurrency	同時に起動するインスタンスの数、またはこのオプションを無効にする場合は 0 (ゼロ)。
-stopconcurrency stop_concurrency	同時に停止するインスタンスの数、またはこのオプションを無効

パラメータ	説明
	にする場合は 0 (ゼロ)。
-policy {AUTOMATIC | MANUAL | NORESTART | USERONLY}	データベースの管理ポリシー。 ● AUTOMATIC(デフォルト): データベースは、データベース・ホスト・コンピュータの再起動時に前回の実行状態(起動または停止)へ自動的に戻ります。 ● MANUAL: データベース・ホスト・コンピュータの再起動時にデータベースは自動的には再起動されません。MANUAL に設定しても、Oracle Clusterware は、実行中のデータベースを監視し、障害発生時にデータベースを再起動します。 ● NORESTART: MANUAL 設定と同様に、データベース・ホスト・コンピュータの再起動時にデータベースは自動的には再起動されません。ただし、NORESTART 設定では、障害が発生しても、データベースを再起動することはありません。 ● USERONLY: データベースは、他の理由(自動起動、依存性による起動、ノード障害など)の結果としてではなく、ユーザー・コマンドによってのみ再起動できます。
-serverpool "server_pool_name"	データベースの配置の制御に使用するサーバー・プールの名前を指定します。このパラメータを指定しない場合、デフォルトで Generic サーバー・プールが使用されます。 ノート: ● このパラメータは、Oracle Clusterware でのみ使用できます。このパラメータと-node パラメータは併用できますが、サーバー・プールは MAX_SIZE=1 とし、1 台のみのサーバー(-node で指定)を構成しておく必要があります。 ● サーバー・プールを追加した後、srvctl add service コマンドを使用してサービスをサーバー・プールに割り当てることができます。
-node node_list	単一のノード名、または非クラスタ(シングル・インスタンス)

パラメータ	説明
	Oracle データベースの構成を変更するノード名のリストをカンマ区切りで指定します。Oracle Database 19c リリース更新(19.7)以降、複数のクラスタ・ノード上の Standard Edition 高可用性データベース構成を変更できます。 ノート: Oracle RAC One Node データベースを変更する際、このパラメータは-serverpool パラメータでのみ使用できます。
-pqpool "pq_pool_list"	パラレル問合せサーバー・プール名のカンマ区切りリスト ノート: このパラメータは、Oracle Clusterware およびポリシー管理データベースでのみ使用できます。
-diskgroup "disk_group_list"	Oracle ASM ディスク・グループのカンマ区切りリスト(データベースで Oracle ASM 記憶域を使用している場合)
-acfspath "acfs_path_list"	データベースの依存性が設定された、二重引用符("")で囲まれた単一の Oracle ACFS パスまたは Oracle ACFS パスのカンマ区切りリスト このパラメータは、データベースが ORACLE_HOME ファイル・システムとは異なるファイル・システムで ORACLE_BASE を使用する場合など、ORACLE_HOME 以外の Oracle ACFS ファイル・システムへの依存性を作成する場合に使用します。
-css_critical {YES | NO}	このパラメータを YES に設定することにより、サービスに重みを追加できます。クラスタ内のノードに障害が発生した場合、Oracle Clusterware は最も重みの小さいノードを削除して、クリティカルなサービスを使用可能な状態に保ちます。 ノート: このパラメータは、管理者管理ノードでのみ使用できます。ノードがポリシー管理ノードになった場合、その時点でこのパラメータは適用されなくなります。
-cpucount cpu_count [-overridepools overridepool_list]	CPU の数を指定します。デフォルト値は 0 です。-overridepools オプションを使用して、特定のサーバー・プールの CPU 数を指定します。
-memorytarget memory_target	データベースに割り当てるターゲット・メモリー(MB)を指定しま

パラメータ	説明
	す。デフォルトは 0 です。
-maxmemory max_memory	リソースに割り当てる最大メモリー(MB)を指定します。-memorytarget を指定して-maxmemory を指定しない場合、-maxmemory はデフォルト値の 0 になります。-memorytarget が-maxmemory 以下であれば、-maxmemory と-memorytarget は両方とも検証されます。
-defaultnetnum network_number	サービスの追加時にネットワーク番号を指定しない場合にデフォルトで設定されるサービスのネットワーク番号を指定します。
-disabledreason {DECOMMISSIONED}	データベースを廃止(つまり、そのデータベースは再度起動できず、使用されていない)としてマークします。これは、将来の日付に削除されるデータベースを対象としています。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮定的に評価できます。 ノート: このパラメータは、ポリシー管理データベースでのみ使用できます。

使用上のノート

- srvctl modify databaseコマンドは、管理者管理データベースをポリシー管理データベースに変換できます。管理者管理データベースの実行については、サーバー・リストが提供されている場合は、そのデータベースが実行されているノードがそのリストに含まれている必要があります。インスタンス名の接頭辞は、srvctl add databaseコマンドの実行後に変更することはできません。
- Oracle RAC One Nodeデータベースでは、管理ポリシーをAUTOMATIC (-policyパラメータを使用)から変更できません。実行しようとすると、エラー・メッセージが表示されます。非クラスタ・データベースが実行されるノードを変更する場合に使用する-nodeパラメータについても同様です。
- ポリシー管理Oracle RAC One Nodeデータベースでは、-serverpoolパラメータを使用して、サーバー・プール間でOracle RAC One Nodeデータベースを移動できますが、指定できるサーバー・プールは1つのみです。サーバー・プールのリストを指定すると、エラーが戻されます。

例

次の例は、データベースのロールをロジカル・スタンバイに変更します。

```
$ srvctl modify database -db crm -role logical_standby
```

次の例では、Oracle ASMディスク・グループSYSFILES、LOGSおよびOLTPを使用するようにracTestデータベースに指示します。

```
$ srvctl modify database -db racTest -diskgroup "SYSFILES,LOGS,OLTP"
```

関連項目

- Oracle Data Guardの構成
- データベースの起動
- データベースの停止

srvctl predict database

特定のデータベースの障害の結果を予測します。

構文

```
srvctl predict database -db db_unique_name [-verbose]
```

使用上のノート

- 確認するデータベースの一意の名前を指定します。
- 必要に応じて、-verboseパラメータを使用して、詳細な出力を表示できます。

srvctl relocate database

ノード間でのOracle RAC One Nodeデータベースの再配置を開始します。

また、このコマンドは、再配置失敗後にクリーン・アップし、Oracle RAC One Nodeデータベースの再配置にのみ使用できます。

構文

このコマンドは、次のいずれかの構文モデルで使用します。

Oracle RAC One Nodeデータベースのオンライン再配置を開始するには:

```
srvctl relocate database -db db_unique_name [-node target_node] [-timeout timeout]
   [-stopoption NORMAL] [-drain_timeout drain_timeout] [-verbose]
```

Oracle RAC One Nodeデータベースの失敗したオンライン再配置を中断するには:

```
srvctl relocate database -db db_unique_name -abort [-revert]
   [-drain_timeout drain_timeout] [-verbose]
```

パラメータ

表A-13 srvctl relocate databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	再配置するデータベースの一意の名前を指定します。
-node target_node	必要に応じて、Oracle RAC One Node データベースを再配置するターゲット・ノードを指定できます。 ノート: 管理者管理 Oracle RAC One Node データベースを再配置する場合は、このパラメータを使用する必要がありま

パラメータ	説明
	す。
-timeout timeout	必要に応じて、Oracle RAC One Node データベースのオンライン・データベース再配置タイムアウトを分単位で指定できます。デフォルト値は 30 です。
-stopoption NORMAL	このパラメータは、実行中のインスタンスのデフォルトの停止オプション(プライマリ・データベースの SHUTDOWN TRANSACTIONAL LOCAL のデフォルト、スタンバイ・データベースの SHUTDOWN IMMEDIATE など)をオーバーライドする場合に使用します。-stopoption で使用できる唯一の値は NORMAL です。
-abort	このパラメータは、失敗したオンライン・データベース再配置を強制終了する場合に使用します。
-revert	このパラメータは、管理者管理 Oracle RAC One Node データベースの候補サーバー・リストから、失敗したオンライン再配置リクエストのターゲット・ノードを削除する場合に使用します。
-drain_timeout timeout	リソースの排出が完了するまでの許容時間を秒数で指定します。有効な値は、空の文字列("")、0、または正の整数です。デフォルト値は空の文字列です(このパラメータが設定されていないことを表します)。0 に設定されている場合は、即座に排出が発生します。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。サービスに設定すると、この値は、コマンドライン値が設定されていない場合に使用されます。
-verbose	このパラメータは、詳細出力を表示する場合に使用します。

使用上のノート

- 再配置するOracle RAC One Nodeデータベースが実行されていない場合、コマンドはエラーを戻します。
- 別のオンライン・データベース再配置がこのOracle RAC One Nodeデータベースに対してアクティブの場合、コマンドはエラーを戻します。
- このOracle RAC One Nodeデータベースのオンライン・データベース再配置が失敗し、ターゲット・ノードがいずれかの

再配置と同じでない場合、失敗したオンライン・データベース再配置を強制終了し、新しい再配置を開始するように求めるエラーがコマンドから戻されます。

- このOracle RAC One Nodeデータベースのオンライン・データベース再配置が失敗し、ターゲット・ノードが同じ(またはターゲットを指定していない)場合、このコマンドはデータベース再配置を試行します。

例

次の例は、rac1という管理者管理Oracle RAC One Nodeデータベースをnode7というサーバーに再配置します。

```
$ srvctl relocate database -db rac1 -node node7
```

srvctl remove database

データベース構成を削除します。

このコマンドの実行後には、パスワード・ファイルがデフォルトの場所にあることを確認します(SYSユーザーのパスワードで、SYSユーザーとしてデータベースに接続する場合)。

構文

```
srvctl remove database -db db_unique_name [-force] [-noprompt] [-verbose]
```

パラメータ

表A-14 srvctl remove databaseコマンドのパラメータ

パラメータ	説明
-database db_unique_name	データベースの一意の名前。
-force	強制的にデータベースを削除して、すべての依存性を無視します。
-noprompt	プロンプトを非表示にします。
-verbose	冗長出力を表示します。

例

crmという名前のデータベースを削除するには:

```
$ srvctl remove database -db crm
```

srvctl setenv database

クラスタ・データベース環境構成を管理します。

構文

このコマンドは、次のいずれかの構文モデルで使用します。

```
srvctl setenv database -db db_unique_name -envs "name=val[,...]"

srvctl setenv database -db db_unique_name -env "name=val"
```

パラメータ

表A-15 srvctl setenv databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	環境変数を設定するデータベースの一意の名前を指定します。
-envs "name=val[,...]"	設定する環境変数の名前/値ペアのカンマ区切りリストを、二重引用符("")で囲んで指定します。
-env "name=val"	カンマやその他の特殊文字を含んだ値を二重引用符("")で囲んで設定する単一の環境変数を指定します。

使用上のノート

ここで、コマンドに関する追加情報を追加します。

例

次の例では、クラスタ・データベースの言語環境変数を設定します。

```
$ srvctl setenv database -db crm -env LANG=en
```

srvctl start database

データベースとその有効化されたインスタンスおよびデータベース・インスタンスが存在するノードのすべてのリスナーを起動します。

起動しないリスナーを無効化できます。

構文

```
srvctl start database -db db_unique_name [-eval] [-startoption start_options]
  [-startconcurrency number_of_instances] [-node node_name]
```

パラメータ

表A-16 srvctl start databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	起動するデータベースの一意の名前を指定します。

パラメータ	説明
-eval	必要に応じて、このパラメータを使用して、コマンドがシステムに及ぼす影響を仮定的に評価できます。
-startoption start_options	必要に応じて、起動コマンドのオプションを設定できます(OPEN、MOUNT、NOMOUNT など)。 ノート: ● このコマンド・パラメータは、すべてのデータベース起動オプションをサポートします。 ● 起動オプションに複数の語を指定する場合(read only、read write など)、語をスペースで区切り、二重引用符(″″)で囲みます。たとえば″read only″とします。 関連項目: 起動オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください
-startconcurrency number_of_instances	必要に応じて、同時に起動するデータベース・インスタンスの数を指定するか、または空の起動同時実行値の場合は 0 を指定できます。このパラメータを srvctl start database コマンドとともに使用すると、srvctl add | modify database コマンドを使用して構成された-startconcurrency 値がオーバーライドされます。 ノート: -startconcurrency パラメータの値が合計インスタンス数より大きい場合、このパラメータには効果がなく、0 と同じです。
-node node_name	必要に応じて、データベースを起動するノードの名前を指定できます。 ノート: ● このコマンドは、Oracle RAC One Node データベースおよび Standard Edition 高可用性データベースにのみ適用されます。 ● 指定するノードは、管理者管理の Oracle RAC

パラメータ	説明
	One Node データベースまたは Standard Edition 高可用性データベースの候補リストに含まれている必要があります。ノードは、ポリシー管理の Oracle RAC One Node データベースのサーバー・プールに存在する必要があります。 ● データベースが指定したノード以外ですでに実行されている場合、このコマンドはエラーを戻します。 ● ノードを指定しない場合、Oracle Clusterware はそのポリシー(分散、リソース数、候補ノードの順序)に従って Oracle RAC One Node データベースまたは Standard Edition 高可用性データベースを起動するノードを選択します。 ● 起動しようとしている Oracle RAC One Node データベースのアクティブなオンライン・データベース再配置がある場合、両方のインスタンスはすでに実行されており、コマンドによりエラー・メッセージが戻されます。オンライン・データベース再配置中にのみ、Oracle RAC One Node データベースの 2 つのインスタンスが存在します。 Oracle RAC One Node データベースのオンライン・データベース再配置が失敗したときに、ノードが指定されていない場合、このコマンドは両方のデータベース・インスタンスを起動しようとします。 Oracle RAC One Node データベースのオンライン・データベース再配置が失敗したときに、ノードが指定されていた場合、このコマンドは失敗された再配置を停止し、そのノードでインスタンスを起動しようとします。

例

次の例では、crmデータベースを起動し、起動オプションを読取り専用に設定します。

```
$ srvctl start database -db crm -startoption "read only"
```

srvctl status database

このコマンドは、データベースの現在の状態を表示します。

構文

```
srvctl status database {-db db_unique_name {[-serverpool serverpool_name]
   | [-sid] [-home]} | -serverpool serverpool_name | -thisversion | -thishome}
   [-force] [-detail] [-verbose]
```

パラメータ

表A-17 srvctl status databaseのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。
-serverpool serverpool_name	オプションで、SRVCTL で情報を表示するノードが含まれているサーバー・プールを指定できます。
-sid	このパラメータは、このノードで実行中の Oracle インスタンスの SID を表示する場合に使用します。
-home	このパラメータは、指定したデータベースの Oracle ホームを表示する場合に使用します。
-thisversion	このパラメータは、SRVCTL と同じ Oracle 製品バージョンのデータベースのステータスを表示する場合に使用します。
-thishome	このパラメータは、この Oracle ホームで構成されたデータベースのステータスを表示する場合に使用します。
-force	無効化されたアプリケーションを含めます
-detail	このパラメータは、詳細なデータベース・ステータス情報を表示する場合に使用します。
-verbose	STATE_DETAILS 属性と INTERNAL_STATE 属性を表示します。これには、STABLE、STARTING、STOPPING、および CLEANING が含まれています。 INTERNAL_STATE が STABLE の場合、SRVCTL は追加の情報を表示しなくなります。INTERNAL_STATE が STARTING の場合、SRVCTL は次を表示します。 Instance instance_name is being started INTERNAL_STATE が CLEANING の場合、SRVCTL は次を表示します。

パラメータ	説明
	Instance instance_name is being cleaned up INTERNAL_STATE が STOPPING の場合、SRVCTL は次を表示します。 Instance instance_name is being stopped

使用上のノート

このコマンドの出力には、データベースの各実行中インスタンスのOracle ASMまたはOracle ASM IOServerインスタンスに関する情報が含まれます。

例

このコマンドでは、次のような出力が表示されます。

```
$ srvctl status database -db db00 -detail
Instance db00_1 is connected to ASM instance +ASM3
Instance db00_2 is connected to ASM I/O server instance +IOS1
```

srvctl stop database

データベース、そのインスタンスおよびそのサービスを停止します。

構文

```
srvctl stop database -db db_unique_name [-stopoption stop_options]
  [-stopconcurrency number_of_instances] [-drain_timeout timeout] [-eval]
  [-force] [-verbose]
```

パラメータ

表A-18 srvctl stop databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	停止するデータベースの一意の名前を指定します。
-stopoption stop_options	必要に応じて、停止コマンドのオプションを指定できます(NORMAL、TRANSACTIONAL LOCAL、IMMEDIATE、ABORT など)。
-stopconcurrency number_of_instances	必要に応じて、同時に停止するデータベース・インスタンスの数を指定するか、または空の停止同時実行値の場合は 0 を指定できます。このパラメータを srvctl stop database コマンドとともに使用すると、srvctl add | modify database コマンドを使用して構成された-

パラメータ	説明
	stopconcurrency 値がオーバーライドされます。 ノート: -stopconcurrency パラメータの値が合計インスタンス数より大きい場合、このパラメータには効果がなく、0 と同じです。
-drain_timeout timeout	必要に応じて、リソース・ドレイニング処理を完了するために許可される時間(秒)を指定できます。デフォルトでは、このパラメータは設定されていません。0 または任意の正の整数を指定できます。空の文字列にすると、パラメータの設定が解除されます。0 を指定した場合、エージェントはサービス・ドレイニングに関連する処理を即時に実行します。 ドレイン・タイムアウトは、サービスがセッションのドレイニングの完了を待機する最大時間であり、これを越えると終了する(srvctl stop service または srvctl stop instance の場合)か、データベースの停止処理に入ります(srvctl stop database)。セッション・ドレイニングが 10 秒で完了し、(CLI またはリソース属性の)ドレイン・タイムアウト値が 100 秒の場合、SRVCTL は 10 秒間待機します。残りの 90 秒は待機しません。
-eval	必要に応じて、このパラメータを使用して、コマンドがシステムに及ぼす影響を仮定的に評価できます。
-force	必要に応じて、このパラメータを使用して、データベース、そのインスタンス、そのサービス、およびそれらのサービスに依存するリソースを停止できます。
-verbose	必要に応じて、このパラメータを使用して、詳細な出力を表示できます。

例

次のコマンド例では、データベースを停止して、詳細な出力を含めます。

```
$ srvctl stop database -db db1 -drain_timeout 50 -verbose
Draining in progress on services svc1,svc2.
Drain complete on services svc1.
Draining in progress on services svc2.
Draining in progress on services svc2.
Drain complete on services svc2.
```

srvctl unsetenv database

クラスタ・データベース環境構成の設定を解除します。

構文

```
srvctl unsetenv database -db db_unique_name -envs "name_list"
```

パラメータ

表A-19 srvctl unsetenv databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	環境変数の設定を解除するデータベースの一意の名前を指定します。
-envs "name_list"	二重引用符("")で囲まれた環境変数名のカンマ区切りリストを指定します。

例

次の例では、2つのクラスタ・データベース環境変数の設定を解除します。

```
$ srvctl unsetenv database -db crm -envs "CLASSPATH,LANG"
```

srvctl update database

指定したデータベースを更新して、新しいリスナー・エンドポイントが使用されるようにします。

構文

```
srvctl update database -db db_unique_name
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 更新するデータベースの一意の名前を指定します。

srvctl upgrade database

このコマンドの実行元であるデータベース・ホームのバージョンに、データベースの構成とそのすべてのサービスをアップグレードします。

構文

```
srvctl upgrade database -db db_unique_name -oraclehome Oracle_home
```

パラメータ

表A-20 srvctl upgrade databaseコマンドのパラメータ

パラメータ	説明
-db db_unique_name	アップグレードするデータベースの一意の名前を指定します。
-oraclehome Oracle_home	アップグレードした ORACLE_HOME へのパスを指定します。

srvctl disable diskgroup

指定したいくつかのノード上の特定のディスク・グループを無効化します。

構文

```
srvctl disable diskgroup -diskgroup diskgroup_name [-node "node_list"]
```

パラメータ

表A-21 srvctl disable diskgroupコマンドのパラメータ

パラメータ	説明
-diskgroup diskgroup_name	無効化する Oracle ASM ディスク・グループの名前を指定します。
-node "node_list"	必要に応じて、ディスク・グループを無効化するノード名を二重引用符("")で囲んだカンマ区切りリストを指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、クラスタ内の2つのノード(mynode1およびmynode2)でOracle ASMディスク・グループ(dgroup1)を無効化します。

```
$ srvctl disable diskgroup -diskgroup dgroup1 -node "mynode1,mynode2"
```

srvctl enable diskgroup

指定したいくつかのノード上の特定のディスク・グループを有効化します。

構文

```
srvctl enable diskgroup -diskgroup diskgroup_name [-node "node_list"]
```

パラメータ

表A-22 srvctl enable diskgroupコマンドのパラメータ

パラメータ	説明

パラメータ	説明
-diskgroup diskgroup_name	有効化する Oracle ASM ディスク・グループの名前を指定します。
-node "node_list"	必要に応じて、ディスク・グループを有効化するノード名を二重引用符(""")で囲んだカンマ区切りリストを指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、ノード(mynode1およびmynode2)でOracle ASMディスク・グループ(diskgroup1)を有効化します。

```
$ srvctl enable diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2"
```

srvctl predict diskgroup

Oracle ASMディスク・グループ障害の結果を予測します。

構文

```
srvctl predict diskgroup -diskgroup diskgroup_name [-verbose]
```

使用上のノート

障害を評価するOracle ASMディスク・グループの名前を指定します。必要に応じて、-verboseパラメータを使用して、詳細な出力を表示できます。

srvctl remove diskgroup

Oracle ClusterwareまたはOracle Restartから、特定のOracle ASMディスク・グループ・リソースを削除します。

構文

```
srvctl remove diskgroup -diskgroup diskgroup_name [-force]
```

使用上のノート

削除するOracle ASMディスク・グループの名前を指定します。必要に応じて、-forceパラメータを使用して、依存性を無視し、ディスク・グループを強制的に削除できます。

例

次の例では、DG1 Oracle ASMディスク・グループを強制的に削除します。

```
$ srvctl remove diskgroup -diskgroup DG1 -force
```

srvctl start diskgroup

指定したいくつかのノード上の特定のOracle ASMディスク・グループ・リソースを起動します。

構文

```
srvctl start diskgroup -diskgroup diskgroup_name [-node "node_list"]
```

パラメータ

表A-23 srvctl start diskgroupコマンドのパラメータ

パラメータ	説明
-diskgroup diskgroup_name	起動する Oracle ASM ディスク・グループの名前を指定します。
-node "node_list"	必要に応じて、ディスク・グループ・リソースを起動するノード名を二重引用符(""")で囲んだカンマ区切りリストを指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、ノード(mynode1およびmynode2)でOracle ASMディスク・グループ(diskgroup1)を起動します。

```
$ srvctl start diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2"
```

srvctl status diskgroup

指定したいくつかのノード上にある特定のディスク・グループのステータスを表示します。

構文

```
srvctl status diskgroup -diskgroup diskgroup_name [-node "node_list"]
  [-detail] [-verbose]
```

パラメータ

表A-24 srvctl status diskgroupコマンドのパラメータ

パラメータ	説明
-diskgroup diskgroup_name	ステータスを表示する Oracle ASM ディスク・グループの名前を指定します。
-node "node_list"	必要に応じて、Oracle ASM ディスク・グループのステータス

パラメータ	説明
	を確認するノード名のカンマ区切りリストを指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-detail	必要に応じて、このパラメータを使用して、Oracle ASM ディスク・グループの詳細なステータス情報を表示できます。
-verbose	必要に応じて、このパラメータを使用して冗長出力を表示できます。

例

次の例では、dgrp1 Oracle ASMディスク・グループのステータスを表示します。

```
$ srvctl status diskgroup -diskgroup dgrp1 -node "mynode1,mynode2" -detail
```

srvctl stop diskgroup

指定したいくつかのノード上の特定のOracle ASMディスク・グループ・リソースを停止します。

構文

```
srvctl stop diskgroup -diskgroup diskgroup_name [-node "node_list"] [-force]
```

パラメータ

表A-25 srvctl stop diskgroupコマンドのパラメータ

パラメータ	説明
-diskgroup diskgroup_name	停止する Oracle ASM ディスク・グループの名前を指定します。
-node "node_list"	必要に応じて、Oracle ASM ディスク・グループ・リソースを停止するノード名を二重引用符("")で囲んだカンマ区切りリストを指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-force	必要に応じて、このパラメータを使用して、強制ディスマウントを実行できます。このパラメータでは、停止するディスク・グループに依存するデータベースは停止されませんが、これによりデー

パラメータ	説明
	タベースに障害が発生する可能性があります。

例

次のコマンドは、2つのノード(mynode1およびmynode2)でOracle ASMディスク・グループ(diskgroup1)を停止します。

```
$ srvctl stop diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2" -force
```

srvctl start home

指定したOracleホームのすべてのOracle Restart管理リソースおよびOracle Clusterware管理リソースを起動します。

構文

```
srvctl start home -oraclehome Oracle_home -statefile state_file -node node_name
```

パラメータ

表A-26 srvctl start homeコマンドのパラメータ

パラメータ	説明
-oraclehome Oracle_home	Oracle Restart または Oracle Clusterware 管理リソースを起動する Oracle ホームのパスを指定します。
-statefile state_file	SRVCTL で状態ファイルを書き込むディレクトリへのパスを指定します。
-node node_name	Oracle ホームが存在するノードの名前を指定します。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次のコマンドにより、Oracleホームが起動されます。

```
$ srvctl start home -oraclehome /u01/app/oracle/product/12.2.0/db_1
  -statefile ~/state.txt -node node1
```

srvctl status home

指定したOracleホームのすべてのOracle Restart管理リソースおよびOracle Clusterware管理リソースのステータスを表示します。

構文

```
srvctl status home -oraclehome Oracle_home -statefile state_file -node node_name
```

パラメータ

表A-27 srvctl status homeコマンドのパラメータ

パラメータ	説明
-oraclehome Oracle_home	Oracle Restart または Oracle Clusterware 管理リソースを起動する Oracle ホームのパスを指定します。
-statefile state_file	このコマンドが生成した状態情報を保持するテキスト・ファイルが含まれているディレクトリへのパスを指定します。
-node node_name	Oracle ホームが存在するノードの名前を指定します。 ノート: このパラメータは、Oracle Clusterware では必須ですが、それ以外では使用できません。

例

次の例では、特定のOracleホームのステータスを取得します。

```
$ srvctl status home -oraclehome /u01/app/oracle/product/12.1/dbhome_1 -statefile
 ~/state.txt -node stvm12
```

前述のコマンドでは次のような出力が戻されます。

```
Database cdb1 is running on node stvm12
```

srvctl stop home

指定したOracleホームから実行されるすべてのOracle Restart管理リソースまたはOracle Clusterware管理リソースを停止します。

構文

```
srvctl stop home -oraclehome Oracle_home -statefile state_file -node node_name
  [-stopoption stop_options] [-force]
```

パラメータ

表A-28 srvctl stop homeコマンドのパラメータ

パラメータ	説明
-oraclehome Oracle_home	Oracle Restart または Oracle Clusterware 管理リソースを起動する Oracle ホームのディレクトリ・パスを指定します。 ノート: 指定する Oracle ホームのパスは、SRVCTL を起動

パラメータ	説明
	する Oracle ホームと同じバージョンである必要があります。
-statefile state_file	SRVCTL で状態ファイルに書き込むディレクトリへのパスを指定します。
-node node_name	Oracle ホームが存在するノードの名前を指定します。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-stopoption stop_options	必要に応じて、データベースの停止オプションを指定できます (NORMAL、TRANSACTIONAL、IMMEDIATE、ABORT など) 関連項目: 停止オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。
-force	必要に応じて、このパラメータを使用して、エラーがレポートされる場合にもリソースを停止できます。

例

次の例では、Oracleホームを停止します。

```
$ srvctl stop home -oraclehome /u01/app/oracle/product/12.1.0/db_1 -statefile
 ~/state.txt
```

srvctl add instance

クラスタ・データベース構成にインスタンスの構成を追加します。

このコマンドは管理者管理データベースにのみ使用できます。ポリシー管理データベースの場合は、srvctl modify srvpoolコマンドを使用してインスタンスを追加し、データベースで使用するサーバー・プールの最大サイズまたは最小サイズ(あるいはその両方)を増やします。

構文

```
srvctl add instance -db db_unique_name -instance instance_name
     -node node_name [-force]
```

パラメータ

表A-29 srvctl add instanceコマンドのパラメータ

パラメータ	説明

パラメータ	説明
-db db_unique_name	インスタンスを追加するデータベースの一意の名前
-instance instance_name	追加するインスタンスの名前
-node node_name	インスタンスを作成するノードの名前
-force	必要に応じて、一部のリソースが停止されても追加操作を強制できます。

使用上のノート

- このコマンドは、Oracle ClusterwareおよびOracle RACでのみ使用できます。
- このコマンドによって、CARDINALITYリソース属性の値が増加します。
- Oracle RAC One Nodeデータベースでこのコマンドを使用しようとすると、コマンドからデータベースをOracle RACに変換する必要があるという内容のエラーが戻されます。

例

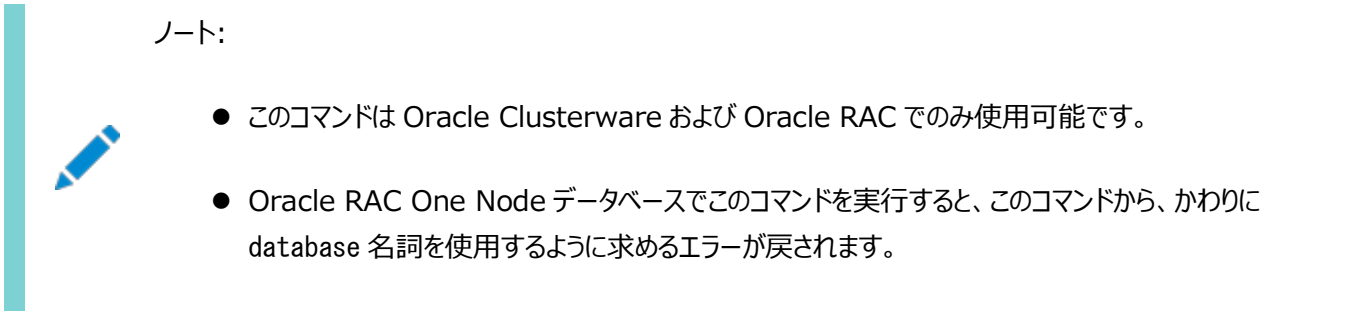
次に、このコマンドの例を示します。

```
$ srvctl add instance -db crm -instance crm01 -node gm01
$ srvctl add instance -db crm -instance crm02 -node gm02
$ srvctl add instance -db crm -instance crm03 -node gm03
```

srvctl disable instance

データベース・インスタンスを無効化します。

このコマンドで無効化するデータベース・インスタンスが、有効化されている最後のデータベース・インスタンスである場合は、この操作によってデータベースも無効化されます。



ノート:

- このコマンドは Oracle Clusterware および Oracle RAC でのみ使用可能です。
- Oracle RAC One Node データベースでこのコマンドを実行すると、このコマンドから、かわりに database 名詞を使用するように求めるエラーが戻されます。

構文

```
srvctl disable instance -db db_unique_name -instance "instance_name_list"
```

パラメータ

表A-30 srvctl disable instanceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	インスタンスを無効化するデータベースの一意の名前を指定します。
-instance "instance_name_list"	無効化する 1 つのインスタンス名、または二重引用符("")で囲んだインスタンス名のカンマ区切りリストを指定します。

例

次の例では、crmデータベースの2つのインスタンス(crm1およびcrm2)を無効化します。

```
$ srvctl disable instance -db crm -instance "crm1,crm3"
```

srvctl enable instance

Oracle RACデータベースのインスタンスを有効化します。
このコマンドを使用してすべてのインスタンスを有効化する場合、データベースも有効化されます。

ノート:

- このコマンドは、Oracle Clusterware および Oracle RAC でのみ使用できます。
- Oracle RAC One Node データベースでこのコマンドを実行すると、このコマンドから、かわりに database 名詞を使用するように求めるエラーが戻されます。

構文

```
srvctl enable instance -db db_unique_name -instance "instance_name_list"
```

パラメータ

表A-31 srvctl enable instanceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	インスタンスを有効化するデータベースの一意の名前を指定します。
-instance "instance_name_list"	有効化するインスタンス名を二重引用符("")で囲んだカンマ区切りリストを指定します。

例

次の例では、crmデータベースの2つのインスタンスを有効化します。

```
$ srvctl enable instance -db crm -instance "crm1,crm2"
```

srvctl modify instance

管理者管理データベースでは、データベース・インスタンスの構成を現在のノードから別のノードへ変更します。ポリシー管理データベースでは、指定されたノードでデータベースを実行するときに使用するインスタンス名を定義します。

構文

```
srvctl modify instance -db db_unique_name -instance instance_name
     -node node_name
```

パラメータ

表A-32 srvctl modify instanceコマンドのパラメータ

パラメータ	説明
-database db_unique_name	データベースの一意の名前を指定します。
-instance instance_name	データベース・インスタンス名を指定します。 ノート: ● ポリシー管理データベース・インスタンスを変更する場合は、インスタンス名にアンダースコア(_)を含める必要があります(pmdb1_1 など)。 ● これまでに起動したことのないインスタンス名を指定するときに、その名前の形式が prefix_number ではない場合は、インスタンス番号、UNDO、および REDO を SPFILE で割り当てることが必要になる場合があります。
-node node_name	インスタンスを実行するノードの名前。このパラメータの値を""に設定できるのは、ポリシー管理データベースの場合のみです。

使用上のノート

このコマンドは、実行インスタンスの名前変更または再配置には使用できません。

例

次の例では、データベース・インスタンスamdb1が、指定されたノードmynodeで実行されるように、管理者管理データベースamdbの構成を変更します。

```
$ srvctl modify instance -db amdb -instance amdb1 -node mynode
```

次の例は、mynodeで実行する場合に、ポリシー管理データベースpmdbでインスタンス名pmdb1を使用するようにします。

```
$ srvctl modify instance -db pmdb -instance pmdb1_1 -node mynode
```

次の例は、前述の例で確立されたディレクティブを削除します。

```
$ srvctl modify instance -db pmdb -instance pmdb1_1 -node ""
```

srvctl remove instance

管理者が管理するデータベースのインスタンスの構成を削除します。

ポリシー管理のデータベースの構成を削除するには、srvctl modify srvpoolコマンドを使用してサーバー・プールのサイズを縮小する必要があります。

構文

```
srvctl remove instance -db db_unique_name -instance instance_name
  [-noprompt] [-force]
```

パラメータ

表A-33 srvctl remove instanceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	管理者管理データベースの一意の名前を指定します。
-instance instance_name	削除するインスタンスの名前を指定します。
-noprompt	このパラメータは、プロンプトを抑止する場合に使用します。
-force	このパラメータは、インスタンスが実行中ではないことの確認をスキップし、実行中であっても削除する場合に使用します。また、インスタンスに実行中のサービスがないことのチェックもスキップし、インスタンスが削除される前にこれらのサービスを停止します。

使用上のノート

- このコマンドは、Oracle ClusterwareおよびOracle RACでのみ使用できます。
- -forceパラメータを使用した場合は、インスタンスで実行されているすべてのサービスが停止されます。インスタンスを削除する前に、削除されるインスタンスを優先インスタンスまたは使用可能インスタンスとして使用しないようにサービスを再構成することをお薦めします。
- Oracle RAC One Nodeデータベースでこのコマンドを使用しようとすると、コマンドから、そのデータベースを削除しないかぎり、インスタンスを削除できないことを示すエラーが戻されます。

例

次の例では、crmデータベースからcrm01データベース・インスタンスを削除します。

```
$ srvctl remove instance -db crm -instance crm01
```

srvctl start instance

クラスタ・データベースのインスタンスを起動します。

srvctl start instanceコマンドは、データベース・インスタンスおよびデータベース・インスタンスを含むノード上のすべてのリスナーを起動するために使用します

構文

srvctl start instanceコマンドは、次の構文モデルの1つで使用します。

```
srvctl start instance -db db_unique_name -node node_name
   [-instance "instance_name"] [-startoption start_options]

srvctl start instance -db db_unique_name -instance "inst_name_list"
    [-startoption start_options]
```

パラメータ

表A-34 srvctl start instance Parameters

パラメータ	説明
-db db_unique_name	データベースの一意の名前。
-node node_name	単一ノードの名前。 ノート: このパラメータは、ポリシー管理データベースに使用します。
-instance { "instance_name" | "inst_name_list" }	シングル・インスタンスの名前またはインスタンス名のカンマ区切りリスト ノート: このパラメータは、管理者管理データベースに使用します。
-startoption start_options	起動コマンドのオプション(OPEN、MOUNT、NOMOUNT など) ノート: 起動オプションに複数の語を指定する場合(read only、read write など)、語をスペースで区切り、二重引用符("")で囲みます。たとえば"read only"とします。

使用上のノート

- このコマンドはOracle ClusterwareおよびOracle RACでのみ使用可能です。
- Oracle RAC One Nodeデータベースでこのコマンドを実行すると、このコマンドから、かわりにdatabase名詞を使用するように求めるエラーが戻されます。

関連項目

- SQL*Plusユーザーズ・ガイドおよびリファレンス

srvctl status instance

インスタンスのステータスを表示します。

ノート:

このコマンドは Oracle Clusterware および Oracle RAC でのみ使用可能です。

srvctl stop instance

srvctl stop instanceコマンドは、インスタンスを停止し、指定したインスタンスで実行中のサービスを停止します。

構文

次の構文モデルのいずれかとともに、このコマンドを使用します。

1つ以上のノードですべてのインスタンスを停止するには:

```
srvctl stop instance -node "node_list" [-stopoption stop_options]
    [-drain_timeout timeout] [-force] [-failover] [-verbose]
```

特定のノードで実行されているデータベースのインスタンスを停止するには:

```
srvctl stop instance -db db_unique_name -node "node_list"
    [-stopoption stop_options] [-drain_timeout timeout] [-force] [-failover] [-verbose]
```

データベースの名前で1つ以上のインスタンスを停止するには:

```
srvctl stop instance -db db_unique_name -instance "instance_name_list"
    [-stopoption stop_options] [-drain_timeout timeout] [-force] [-failover] [-verbose]
```

パラメータ

表A-35 srvctl stop instanceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	停止するインスタンスのデータベースの一意の名前を指定します。
-node "node_list"	二重引用符("")で囲まれたノード名のカンマ区切りリストを指定します。 -db の値を指定せずに-node を指定すると、ノードで実行されているすべてのインスタンスは、どのデータベースに関連付けられているかにかかわらず停止します。
-instance "instance_name_list"	二重引用符("")で囲まれたインスタンス名のカンマ区切りリス

パラメータ	説明
	トを指定します。
-stopoption stop_options	停止コマンドのオプションを指定します(NORMAL、TRANSACTIONAL LOCAL、IMMEDIATE、ABORT など)。
-drain_timeout timeout	リソース・ドレイニング処理を完了するために許可される時間(秒)。デフォルトでは、このパラメータは設定されていません。0 または任意の正の整数を指定できます。空の文字列にすると、パラメータの設定が解除されます。0 を指定した場合、エージェントはサービス・ドレイニングに関連する処理を即時に実行します。 ドレイン・タイムアウトは、サービスがセッションのドレイニングの完了を待機する最大時間であり、これを越えると終了する(srvctl stop service または srvctl stop instance の場合)か、データベースの停止処理に入ります(srvctl stop database)。セッション・ドレイニングが 10 秒で完了し、(CLI またはリソース属性の)ドレイン・タイムアウト値が 100 秒の場合、SRVCTL は 10 秒間待機します。残りの 90 秒は待機しません。
-force	このパラメータは、srvctl stop instance コマンドがエラーで失敗したときに、インスタンスと実行中のサービスを強制的に停止する場合に使用します。
-failover	-failover を指定すると、インスタンス停止時に、サービスは使用可能インスタンスにフェイルオーバーします。
-verbose	冗長出力を表示します。

使用上のノート

このコマンドをOracle RAC One Nodeデータベースで実行すると、このコマンドは、かわりにsrvctl stop databaseコマンドを使用するように求めるエラーを返します。

例

次のコマンド例では、ノードserver1で実行されているdb1データベースのインスタンスを停止し、詳細な出力が含まれます。

```
$ srvctl stop instance -db db1 -node server1 -drain_timeout 50 -verbose
Draining in progress on services svc1
Draining in progress on services svc1
Drain complete on services svc1
```

関連項目

- データベースの停止

srvctl update instance

srvctl update instanceコマンドは、データベース・インスタンスのオープン・モードまたはターゲットOracle ASMインスタンスを変更します。

構文

```
srvctl update instance -db db_unique_name [-instance "instance_name_list"
    | -node "node_list"] [-startoption start_options] [-targetinstance instance_name]
```

パラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前。
-instance "instance_name_list" | -node "node_list"	更新するインスタンス名またはノード名のカンマ区切りリスト。ノード名のリストを指定すると、SRVCTL は指定されたノードで実行中のインスタンスを更新します。
-startoption start_options	データベースの起動オプション(OPEN、MOUNT、"READ ONLY"など)。
-targetinstance instance_name	ターゲットの Oracle ASM インスタンスまたは Oracle ASM IOServer インスタンス。間に空白を入れずに二重引用符(""")を使用して、デフォルトのターゲット・インスタンスを指定します。

例

次に、このコマンドの例を示します。

```
$ srvctl update instance -db db00 -instance db00_3 -targetinstance +ASM2
```

srvctl add listener

クラスタ内のすべてのノードにリスナーを追加します。

構文

次の構文モデルのいずれかとともに、このコマンドを使用します。

Oracle Databaseリスナーを作成するには:

```
srvctl add listener [-listener listener_name] [-netnum network_number] [-oraclehome Oracle_home]
```

```
  [-user user_name] [-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][/IPC:key][/NMP:pipe_name]
  [/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]" [-group group_name]] [-invitednodes
"node_list"]
  [-invitedsubnets "subnet_list"] [-skip]
```

Oracle ASMリスナーを作成するには:

```
srvctl add listener [-listener listener_name] -asmlistener [-subnet subnet]
  [-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][/IPC:key][/NMP:pipe_name]
  [/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]" [-group group_name]] [-invitednodes
"node_list"]
  [-invitedsubnets "subnet_list"] [-skip]
```

SCANリスナーを作成するには、srvctl add scan_listenerコマンドを使用します。

パラメータ

表A-36 srvctl add listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	リスナー名を指定します。このパラメータは省略可能です。 このパラメータを指定しない場合、リスナーの名前はデフォルトで LISTENER (データベース・リスナーの場合)または LISTENER_ASM (Oracle ASM リスナーの場合)になります。
-netnum network_number	VIP を取得するネットワーク番号(オプション)。指定しなかった場合は、nodeapps VIP が取得されるネットワークと同じデフォルトのネットワークから VIP が取得されます。 ノート: Oracle Database リスナーを追加する場合は、このパラメータを使用してください。
-oraclehome oracle_home	クラスタ・データベースの Oracle ホームを指定します。このパラメータを含めなかった場合は、SRVCTL によってデフォルトで Grid ホームが使用されます。 ノート: Oracle Database リスナーを追加する場合は、このパラメータを使用してください。
-user user_name	リスナーを実行するユーザーを、より低い権限のユーザーに設定するには、このコマンドを使用します。セキュリティを向上させるためにこのパラメータを使用することをお薦めします。 ノート: ● このコマンドを実行し、-user パラメータを使用するには、root としてログインする必要があります。 ● Oracle Database リスナーを追加する場合は、このパラメータを使用します。

パラメータ	説明
	● -user パラメータを使用する場合は、次のことを確認します。 このパラメータを使用する前に、Oracle Base ディレクトリ内のリスナー・ログ・ディレクトリと、Grid_home/network/admin/user_name ディレクトリの両方が各ノードに存在する必要があります。また、user_name は、ディレクトリでの読取り、書込みおよび実行の権限を持っている必要があります。 Oracle_Base/diag/tnslsnr/host_name/lower_case_listener_name ディレクトリが存在し、user_name がこのディレクトリに対して読取り、書込みおよび実行の権限を持っています。 ● リスナーの管理に LSNRCTL を使用するには、まず、TNS_ADMIN を Grid_home/network/admin/user_name に設定しておく必要があります。
-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][/IPC:key][/NMP:pipe_name][/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]"]	リスナーのプロトコル仕様。port_list を使用して、TCP ポートまたはリスナー・エンドポイントのカンマ区切りリストを指定します。 Oracle Database リスナーに-endpoints パラメータを指定しない場合、SRVCTL は、1521 から 1540 の間で空きポートを探します。 TCPS ポート、SDP ポート、および EXADIRECT ポートのエンドポイントを指定することもできます。 ノート: この属性はオンライン・リソース属性変更を使用して変更できます。
-group group_name	必要に応じて、-endpoints に-group パラメータを使用して、ソース・エンドポイントのグループを指定することもできます。このパラメータは、Exadata および Exalogic システムで EXADIRECT プロトコルに使用されます。
-invitednodes "node_list"	リスナーへの登録を許可するノード名のカンマ区切りリストを指定します。
-invitedsubnets "subnet_list"	リスナーへの登録を許可するサブネットのカンマ区切りリストを指定します。
-skip	ポートの確認をスキップすることを示します。
-asmlistener	リスナー・タイプとして Oracle ASM リスナーを指定します。-listener パラメータを指定しない場合、Oracle ASM リスナーの名前はデフォルトで LISTENER_ASM になります。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-subnet subnet	Oracle ASM リスナーに使用するサブネットを指定します。

パラメータ	説明
	ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

使用上のノート

-userパラメータを指定する場合、LinuxおよびUNIXプラットフォームで、rootユーザーとしてこのコマンドを実行する必要があります。

例

次のコマンドは、ポート1341、1342および1345でリスニングするlistener112というリスナーを追加し、クラスタ内の各ノードのOracleホーム・ディレクトリから実行されます。

```
$ srvctl add listener -listener listener112 -endpoints "1341,1342,1345"
-oraclehome /u01/app/oracle/product/12.2.0/db1
```

リスナーがGridホームではなくOracle RACホームで構成されている場合は、listener.oraファイルが、$ORACLE_HOME/bin/orabasehomeユーティリティによって返される場所の下のサブディレクトリnetwork/admin (/u02/racbase/homes/OraDB20Home1/network/adminなど)に作成されます。

srvctl config listener

Oracle Clusterwareに登録されている特定のリスナーの構成情報を表示します。

構文

```
srvctl config listener [-listener listener_name | -asmlistener] [-all]
```

パラメータ

表A-37 srvctl config listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name | -asmlistener	特定のリスナーの名前またはリスナーのタイプ(Oracle ASM)。 このパラメータを指定しない場合は、SRVCTL によってデフォルトのデータベース・リスナーの構成が表示されます。
-all	詳細な構成情報の出力。

例

このコマンドによって、次のような出力が返されます。

```
Name: LISTENER
Subnet: 10.100.200.195
Type: type
Owner: scott
```

```
Home: Grid_home
End points: TCP:1521
```

srvctl disable listener

リスナー・リソースを無効化します。

構文

```
srvctl disable listener [-listener listener_name] [-node node_name]
```

パラメータ

表A-38 srvctl disable listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	必要に応じて、特定のリスナー・リソースの名前を指定できます。このパラメータを指定しない場合、リスナー名はデフォルトで LISTENER になります。
-node node_name	必要に応じて、無効化するリスナー・リソースが実行されているクラスタ・ノードの名前を指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、ノードnode5でlistener_crmという名前のリスナー・リソースを無効化します。

```
$ srvctl disable listener -listener listener_crm -node node5
```

srvctl enable listener

リスナー・リソースを有効化します。

構文

```
srvctl enable listener [-listener listener_name] [-node node_name]
```

パラメータ

表A-39 srvctl enable listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	必要に応じて、リスナー・リソースの名前を指定できます。このパラメータを使用しない場合、リスナー名はデフォルトで

パラメータ	説明
	LISTENER になります。
-node node_name	必要に応じて、リスナーを有効化するクラスタ・ノードの名前を指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。

例

次の例では、node5という名前のノードでlistener_crmという名前のリスナーを有効化します。

```
$ srvctl enable listener -listener listener_crm -node node5
```

srvctl getenv listener

指定したリスナーの環境変数を表示します。

構文

```
srvctl getenv listener [-listener listener_name] [-envs "name_list"]
```

パラメータ

表A-40 srvctl getenv listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	必要に応じて、環境変数を取得するリスナーの名前を指定できます。 このパラメータを使用しない場合、リスナー名はデフォルトで LISTENER になります。
-envs "name_list"	必要に応じて、環境変数の名前を二重引用符(“”)で囲んだカンマ区切りリストを指定できます。 このパラメータを使用しなかった場合、SRVCTL はリスナーに関連付けられているすべての環境変数の値を表示します。

例

次の例は、デフォルト・リストに指定されたすべての環境変数を一覧表示します。

```
$ srvctl getenv listener
```

srvctl modify listener

リスナーのいくつかの要素を変更します

デフォルト・リスナーか、Oracle RestartまたはOracle Clusterwareに登録されている特定のリスナーについて、リスナーの実行元Oracleホーム・ディレクトリとその所有者であるオペレーティング・システム・ユーザーの名前、リスナー・エンドポイント、またはリスナーがリスニングするパブリック・サブネットを変更します。

リスナーの名前を変更する場合、srvctl remove listenerおよびsrvctl add listenerコマンドを使用します。

構文

```
srvctl modify listener [-listener listener_name] [-oraclehome oracle_home]
 [-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][/IPC:key][/NMP:pipe_name]
 [/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]"] [-group <group>]
 [-user user_name] [-netnum network_number]
```

パラメータ

表A-41 srvctl modify listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	必要に応じて、変更するリスナーの名前を入力できます。 このパラメータを使用しない場合、デフォルトの名前である LISTENER が使用されます。
-oraclehome oracle_home	このパラメータを使用することを選択した場合は、SRVCTL は指定した Oracle ホームから実行されるようにリスナーを移動します。 ノート: このパラメータを使用する場合は、新しい ORACLE_HOME 所有者に対応するリソース所有権を SRVCTL が更新できるように、権限を持つユーザーでコマンドを実行してください。
-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][/IPC:key][/NMP:pipe_name][/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]"	必要に応じて、このパラメータを使用して、リスナーのプロトコル仕様を変更できます。プロトコルの文字列を二重引用符("")で囲む必要があります。 port_list は、ポート番号のカンマ区切りリストです。 TCPS ポート、SDP ポートおよび EXADIRECT ポートのエンドポイントを変更することもできます。 ノート: この属性はオンライン・リソース属性変更を使用して変更できます。

パラメータ	説明
-group group_name	必要に応じて、-endpoints に-group パラメータを使用して、ソース・エンドポイントのグループを指定することもできます。このパラメータは、Exadata および Exalogic システムで EXADIRECT プロトコルに使用されます。
-user user_name	必要に応じて、指定された Oracle リスナーを所有するオペレーティング・システム・ユーザーの名前を指定できます ノート: ● このパラメータは、Oracle Clusterware でのみ使用できます。 ● このコマンドを実行し、-user パラメータを使用するには、root としてログインする必要があります。 ● -user パラメータを使用する場合は、次のことを確認します。 このパラメータを使用する前に、ORACLE_BASE 内のリスナー・ログ・ディレクトリと、Grid_home/network/admin/user_name ディレクトリの両方を各ノードに配置する必要があります。また、user_name は、ディレクトリでの読取り、書込みおよび実行の権限を持っている必要があります。 $ORACLE_BASE/diag/tnslsnr/host_name/lower_case_listener_name ディレクトリが存在し、user_name がこのディレクトリに対して読取り、書込みおよび実行の権限を持っています。 ● リスナーの管理に LSNRCTL を使用するには、まず、TNS_ADMIN を Grid_home/network/admin/user_name に設定しておく必要があります。
-netnum network_number	必要に応じて、このパラメータを使用して、リスナーがリスニングするパブリック・サブネットを変更できます。 ノート: 常に 1 つ以上のリスナーをデフォルトのネットワークに持つことをお薦めします。このパラメータを使用して、デフォルトのネットワークをリスニングする唯一のリスナーのネットワークを変更しないでください。

例

次の例は、デフォルト・リスナーのTCPポートを変更します。

```
$ srvctl modify listener -endpoints "TCP:1521,1522"
```

srvctl predict listener

リスナー障害の結果を予測します。

構文

```
srvctl predict listener listener_name [-verbose]
```

使用上のノート

障害の結果を予測するリスナーの名前を指定します。必要に応じて、-verboseパラメータを使用して、詳細な出力を表示できます。

srvctl remove listener

Oracle ClusterwareまたはOracle Restartから特定のリスナーまたはすべてのリスナーの構成を削除します。

構文

```
srvctl remove listener [-listener listener_name | -all] [-force]
```

使用上のノート

- 必要に応じて、削除するリスナーの名前を指定したり、-allパラメータを使用してすべてのリスナーを削除できます。リスナー名を指定しない場合、リスナー名はデフォルトでLISTENER (データベース・リスナーの場合)またはLISTENER_ASM (Oracle ASMリスナーの場合)になります。
- 必要に応じて、-forceパラメータを使用して、このリスナーに依存する他のリソース(データベースなど)があるかどうかの確認をスキップし、リスナーを削除できます。

例

次の例では、lsnr01という名前のリスナーの構成を削除します。

```
$ srvctl remove listener -listener lsnr01
```

srvctl setenv listener

リスナー環境構成を管理します。

構文

このコマンドは、次のいずれかの構文モデルで使用します。

```
srvctl setenv listener [-listener listener_name] -envs "name=val[,...]"
srvctl setenv listener [-listener listener_name] -env "name=val"
```

パラメータ

表A-42 srvctl setenv listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	必要に応じて、リスナーの名前を指定できます。 このパラメータを使用しない場合、リスナー名はデフォルトで LISTENER になります。
-envs "name=val[,...]"	環境変数の名前/値ペアのカンマ区切りリストを、二重引用符("")で囲んで指定します。
-env "name=val"	このパラメータを使用して、単一の環境変数を二重引用符("")で囲み、カンマまたは他の特殊文字を含んだ値に設定できるようにします。

例

次の例は、デフォルト・リスナーの言語環境構成を設定します。

```
$ srvctl setenv listener -env "LANG=en"
```

srvctl start listener

特定のノードでデフォルト・リスナーを起動するか、Oracle Clusterwareに登録されているすべてのノードまたは特定のノードで指定のリスナーを起動します。

構文

```
srvctl start listener [-node node_name] [-listener listener_name]
```

パラメータ

表A-43 srvctl start listenerコマンドのパラメータ

パラメータ	説明
-node node_name	リスナーを起動する特定のノード名を指定します。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-listener listener_name	特定のリスナー名を指定します。リスナーの名前を取得するには、srvctl config listener コマンドを使用します。 このパラメータに値を割り当てていない場合、SRVCTL はクラスタ内の既知のリスナーをすべて起動します。

例

次のコマンドは、server3というノード上のOracle Clusterwareによって管理されるすべてのリスナーを起動します。

```
$ srvctl start listener -node server3
```

srvctl status listener

リスナー・リソースのステータスを表示します。

構文

```
srvctl status listener [-listener listener_name] [-node node_name] [-verbose]
```

パラメータ

表A-44 srvctl status listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	オプションで、リスナーの名前を指定できます。 このパラメータを使用しない場合、リスナー名はデフォルトでLISTENER になります。
-node node_name	必要に応じて、クラスタ・ノードの名前を指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-verbose	必要に応じて、このパラメータを使用して冗長出力を表示できます。

例

次の例では、ノードnode2のデフォルト・リスナーのステータスを表示します。

```
$ srvctl status listener -node node2
```

srvctl stop listener

すべてのノードまたは指定したノードのデフォルト・リスナーまたは特定のリスナーを停止します。

このコマンドは、非クラスタ・データベースのリスナーを非クラスタ・データベース・ホームから停止する場合にも使用できます。ただし、非クラスタ・データベース・ホームから実行する場合は、SRVCTLに-nodeパラメータは指定できません。

構文

```
srvctl stop listener [-listener listener_name] [-node node_name] [-force]
```

パラメータ

表A-45 srvctl stop listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	停止するリスナーの名前を指定します。 このパラメータに値を割り当てていない場合、SRVCTL はクラスタ内の既知のリスナーをすべて停止します。
-node node_name	オプションで、特定のリスナーを実行している単一ノードの名前を指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-force	強制的にリスナーを停止します。

例

次のコマンドでは、ノードmynode1のリスナーをすべて停止します。

```
$ srvctl stop listener -node mynode1
```

srvctl unsetenv listener

リスナーの環境構成の設定を解除します。

構文

```
srvctl unsetenv listener [-listener listener_name] -envs "name_list"
```

パラメータ

表A-46 srvctl unsetenv listenerコマンドのパラメータ

パラメータ	説明
-listener listener_name	必要に応じて、環境構成の設定を解除するリスナーの名前を指定できます。 このパラメータを使用しない場合、リスナー名はデフォルトで LISTENER になります。
-envs "name_list"	設定を解除する環境変数名を二重引用符("")で囲んだカンマ区切りリストを指定します。

例

次の例は、デフォルト・リスナーの環境変数TNS_ADMINの設定を解除します。

```
$ srvctl unsetenv listener -envs "TNS_ADMIN"
```

srvctl update listener

新しいエンドポイントをリスニングするようリスナーを更新します。

構文

```
srvctl update listener
```

使用上のノート

- このコマンドには、-helpを除き、その他のパラメータは指定できません。
- このコマンドは、Oracle Clusterwareでのみ使用できます。

srvctl add network

静的ネットワークまたは動的ネットワークを追加します。

サーバーが複数のネットワークに接続している場合、このコマンドを使用して、Oracle RAC用の追加のネットワーク・インタフェースを構成でき、これによって複数のパブリック・ネットワーク上にVIPを作成できるようになります。

構文

```
srvctl add network [-netnum net_number] -subnet subnet/netmask[/if1[|if2...]]
   [-nettype {STATIC | DHCP | AUTOCONFIG | MIXED}] [-pingtarget "ping_target_list"]
   [-skip] [-verbose]
```

パラメータ

表A-47 srvctl add networkコマンドのパラメータ

パラメータ	説明
-netnum net_number	ネットワーク番号。デフォルトは 1 です。
-subnet subnet/netmask [/if1[|if2|...]]	サブネットを定義します。インタフェース名を指定しない場合、ネットワークは指定されたサブネットの任意のインタフェースを使用します。 IPv6 の場合、netmask は接頭辞の長さ(64 など)です。
-nettype {STATIC|DHCP|AUTOCONFIG|MIXED}	ネットワーク・タイプ(STATIC、DHCP、AUTOCONFIG または MIXED)を指定します。 ネットワーク・タイプとして STATIC を指定した場合は、srvctl add vip コマンドを使用して仮想 IP アドレスを指定する必要があります。

パラメータ	説明
	ネットワーク・タイプとして DHCP を指定した場合は、VIP エージェントによって DHCP サーバーから IP アドレスが取得されます。 ネットワーク・タイプとして AUTOCONFIG を指定した場合は、VIP エージェントによってネットワークのステートレス IPv6 アドレスが生成されます。IPv6 ネットワークにのみ AUTOCONFIG を使用できます。サブネット/ネットマスクの指定が IPv6 アドレスに対応していない場合は、SRVCTL によってエラーが戻されます。 ネットワーク・タイプに MIXED を指定する場合、VIP リソースは、静的 IP アドレスと、DHCP サーバーから動的に取得する IP アドレス(IPv4 の場合)またはステートレス自動構成を使用して動的に取得する IP アドレス(IPv6 の場合)との両方を使用します。
-pingtarget "ping_target_list"	ping する IP アドレスまたはホスト名のカンマ区切りリスト。
-skip	このパラメータはサブネットのチェックをスキップする場合に使用します。
-verbose	詳細な出力。

使用上のノート

- このコマンドを実行するには、LinuxシステムおよびUNIXシステムではrootユーザーとしてログインし、Windowsでは管理者権限を持つユーザーとしてログインする必要があります。
- このコマンドはOracle Clusterwareでのみ使用可能です。
- DHCP割当てのネットワークがサポートされるのはデフォルトのネットワークのみで、後続のネットワークではサポートされません。
- また、LISTENER_NETWORKSデータベース初期化パラメータを使用して、クライアントが適切なネットワークにリダイレクトするように制御できます。

例

次に、このコマンドの例を示します。

```
# srvctl add network -netnum 3 -subnet 192.168.3.0/255.255.255.0
```

srvctl config network

クラスタのネットワーク構成を表示します。

構文

```
srvctl config network [-netnum network_number]
```

使用上のノート

- 構成情報を表示するネットワークを指定します。

- このコマンドはOracle Clusterwareでのみ使用可能です。

例

次に、このコマンドの例を示します。

```
$ srvctl config network -netnum 2
```

srvctl modify network

ネットワークのサブネット、ネットワーク・タイプまたはIPアドレス・タイプを変更します。

構文

```
srvctl modify network [-netnum network_number] [-subnet subnet/netmask
  [/if1[|if2|...]]] [-nettype network_type | -iptype {ipv4 | ipv6 | both}]
  [-pingtarget "ping_target_list"] [-verbose]
```

パラメータ

表A-48 srvctl modify networkコマンドのパラメータ

パラメータ	説明
-netnum network_number	必要に応じて、変更するネットワーク番号を指定できます。デフォルトは 1 です。
-subnet subnet/netmask [/if1[|if2|...]]	必要に応じて、パブリック・ネットワークのサブネット番号を指定できます。ネットマスクとインタフェースを指定すると、変更対象のネットワークのネットマスクとインタフェースが変更されます。IPv6 サブネットを指定する場合は、**ネットマスク**のかわりに接頭辞の長さ(64 など)を入力します。インタフェース名を指定しない場合、VIP は指定されたサブネットの任意のインタフェースを使用します。 -nettype パラメータを使用してネットワーク・タイプを変更する場合は、-subnet パラメータを使用して既存の IPv4 または IPv6 ネットワークを指定する必要があります。また、-subnet パラメータに指定するサブネットおよびネットマスクは、変更するネットワークのサブネットおよびネットマスクを変更しません。
-nettype network_type	必要に応じて、このパラメータを使用して、ネットワーク・タイプを static、dhcp、autoconfig または mixed に変更できます。
-iptype {ipv4 | ipv6 | both}	ネットワーク・タイプを変更するかわりに、IP アドレスのタイプを ipv4、ipv6 または both に変更できます。
-pingtarget "ping_target_list"	必要に応じて、ping する IP アドレスまたはホスト名を二重引

パラメータ	説明
	用符(″″)で囲んだカンマ区切りリストを指定できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- このコマンドを実行するには、LinuxシステムおよびUNIXシステムではrootでログインし、Windowsでは管理者権限を持つユーザーとしてログインする必要があります。
- ネットワークのIPアドレス・タイプはIPv4からIPv6に、またはIPv6からIPv4に変更できます。
- ネットワーク・タイプとしてstaticを指定した場合は、srvctl add vipコマンドを使用して仮想IPアドレスを指定する必要があります。
- ネットワーク・タイプとしてdhcpを指定した場合は、VIPエージェントによってDHCPサーバーからIPアドレスが取得されます。
- ネットワーク・タイプとしてautoconfigを指定した場合は、VIPエージェントによってネットワークのステートレスIPv6アドレスが生成されます。このパラメータは、IPv6ネットワークに対してのみ使用できます。サブネット/ネットマスクの指定がIPv6アドレスに対応していない場合は、SRVCTLによってエラーが戻されます。
- ネットワークをstaticからmixedに変更する場合は、動的に取得されるアドレスに名前を登録できるように、まずGNSを構成する必要があります。
- ネットワーク・タイプとしてmixedを指定した場合は、VIPリソースで静的IPアドレスとDHCPまたはautoconfigを通じて動的に取得されたIPアドレスの両方が使用されます。
- ネットワーク・タイプとしてmixed_autoconfigを指定した場合は、VIPリソースで静的IP構成が保持され、DHCPサーバーからIPアドレスが取得されるか(IPv4ネットワークを指定した場合)、ステートレスな自動構成IPアドレスが生成されます(IPv6ネットワークを指定した場合)。

例

次の例は、サブネット数、ネットマスクとインタフェース・リストを変更します。

```
# srvctl modify network -subnet 192.168.2.0/255.255.255.0/eth0
```

次の例は、2つ目のネットワークをDHCPに変更します。

```
# srvctl modify network -netnum 2 -nettype dhcp
```

次の例は、IPv6サブネットおよびネットマスクをデフォルト・ネットワークに追加します。

```
# srvctl modify network -subnet 2606:b400:400:18c0::/64
```

次の例は、ネットワークからIPv4構成を削除します。

```
# srvctl modify network -iptype ipv6
```

関連項目

- Oracle Clusterware管理およびデプロイメント・ガイド

srvctl predict network

ネットワーク障害の結果を予測します。

構文

```
srvctl predict network [-netnum network_number] [-verbose]
```

使用上のノート

必要に応じて、障害を評価するネットワークを指定できます。デフォルト値は1です。また、-verboseパラメータを使用して、詳細な出力を表示することもできます。

例

次の例では、ネットワーク番号2で障害の結果を予測します。

```
$ srvctl predict network -netnum 2
```

srvctl remove network

ネットワーク構成を削除します。

構文

```
srvctl remove network {-netnum network_number | -all} [-force] [-verbose]
```

パラメータ

表A-49 srvctl remove networkコマンドのパラメータ

パラメータ	説明
-netnum network_number | -all	削除するネットワーク番号を指定します。または、-all パラメータを使用して、すべてのネットワークを削除することを指定できます。
-force	必要に応じて、このパラメータを使用して、依存性に関係なく、指定したネットワークを削除できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。

- このコマンドを実行するには、完全な管理権限が必要です。LinuxシステムおよびUNIXシステムではrootでログインし、Windowsシステムでは管理者権限を持つユーザーとしてログインする必要があります。

例

次の例では、ネットワークを削除します。

```
# srvctl remove network -netnum 3
```

srvctl add nodeapps

指定したノードにノード・アプリケーション構成を追加します。

構文

このコマンドと、次の構文モデルの1つを一緒に使用して、特定のノードとVIP、または特定のサブネットとネットマスクを指定します。

```
srvctl add nodeapps
    {-node node_name -address {vip_name | ip_address}/netmask[/if1[|if2|..]] [-skip]}
    [-emport em_port] [-onslocalport ons_local_port]
    [-onsremoteport ons_remote_port] [-onshostport hostname_port_list]
    [-remoteservers hostname_port_list [-verbose]

srvctl add nodeapps -subnet subnet/netmask[/if1[|if2|...]] [-emport em_port]
    [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]
    [-onshostport hostname_port_list] [-remoteservers hostname_port_list]
    [-verbose]
```

パラメータ

表A-50 srvctl add nodeappsコマンドのパラメータ

パラメータ	説明
-node node_name	ノード・アプリケーションを作成するノードの名前。ノード名はオプションであり、ローカル・ノードでコマンドを実行する場合は指定する必要がありません。
-address {vip_name | ip_address}/netmask[/if1[|if2|..]]}	指定したノードに従来の VIP ノード・アプリケーションを作成します。 ノート: アップグレード構成および新規の DHCP 以外の構成にこのパラメータを使用する必要があります。
-skip	VIP アドレスの到達可能性の確認をスキップするには、このパラメータを指定します。
-subnet subnet/netmask [/if1[|if2 |...]]	DHCP サブネットを作成します。インタフェース名を指定しない場合、VIP は指定されたサブネットの任意のインタフェースを使用します。
-emport em_port	Oracle Enterprise Manager がリスニングするローカル・ポート。デフォルト・ポートは

パラメータ	説明
	2016 です。
-onslocalport ons_local_port	そのノードの Oracle Notification Service デーモンのリスナー・ポート。 この値を指定しない場合、Oracle Notification Service デーモンのリスナー・ポートはデフォルトで 6100 が使用されます。 ノート: ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。
-onsremoteport ons_remote_port	リモート Oracle Notification Service デーモン接続用のポート番号。 ポート番号を指定しない場合、デフォルト値の 6200 が Oracle Notification Service リモート・ポートとして使用されます。 ノート: ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。
-onshostport host_port_list	Oracle Notification Service ネットワークには含まれていても Oracle Clusterware クラスタには含まれていないリモート・ホストの host[:port]ペアのリスト。 ノート: リモート・ホストに port を指定しなかった場合は、ons_remote_port が使用されます。
-remoteservers host_port_list	クラスタに含まれていないサーバーで使用される Oracle Notification Service デーモン用の host[:port]ペアのリスト。
-verbose	詳細出力

使用上のノート

- このコマンドを実行するには、LinuxシステムおよびUNIXシステムではrootでログインし、Windowsでは管理者権限を持つユーザーとしてログインする必要があります。
- このコマンドはOracle Clusterwareでのみ使用可能です。

例

次に、このコマンドの例を示します。

```
# srvctl add nodeapps -node crmnode1 -address 1.2.3.4/255.255.255.0
```

srvctl config nodeapps

クラスタ内の各ノードのVIP構成を表示します。

ノート:

このコマンドは Oracle Clusterware でのみ使用可能です。

構文

```
srvctl config nodeapps [-viponly] [-onsonly]
```

使用上のノート

-viponlyを使用して、VIPアドレス構成を表示します。-onsonlyを使用して、Oracle Notification Service構成を表示します。

例

次に、このコマンドの例を示します。

```
$ srvctl config nodeapps -viponly -onsonly
```

srvctl disable nodeapps

クラスタ内のすべてのノードのノード・アプリケーションを無効化します。

構文

```
srvctl disable nodeapps [-gsdonly] [-adminhelper] [-verbose]
```

パラメータ

表A-51 srvctl disable nodeappsコマンドのパラメータ

パラメータ	説明
-gsdonly	必要に応じて、このパラメータを使用して、グローバル・サービス・デーモン(GSD)のみを無効化できます。
-adminhelper	必要に応じて、このパラメータを使用して、Administratorヘルパーのみを無効化できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このパラメータは、Oracle Clusterwareでのみ使用できます。

例

次の例では、GSDを無効化します。

```
$ srvctl disable nodeapps -gsdonly -verbose
```

srvctl enable nodeapps

クラスタ内のすべてのノードのノード・アプリケーションを有効化します。

構文

```
srvctl enable nodeapps [-gsdonly] [-adminhelper] [-verbose]
```

パラメータ

表A-52 srvctl enable nodeappsコマンドのパラメータ

パラメータ	説明
-gsdonly	必要に応じて、このパラメータを使用して、グローバル・サービス・デーモン(GSD)のみを有効化できます。
-adminhelper	必要に応じて、このパラメータを使用して、Administrator ヘルパーのみを有効化できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、GSDを有効化します。

```
$ srvctl enable nodeapps -gsdonly -verbose
```

srvctl getenv nodeapps

ノード・アプリケーション構成の環境変数を表示します。

構文

```
srvctl getenv nodeapps [-viponly] [-onsonly] [-envs "name_list"]
```

パラメータ

表A-53 srvctl getenv nodeappsコマンドのパラメータ

パラメータ	説明
-viponly	必要に応じて、このパラメータを使用して、VIP アドレス構成を

パラメータ	説明
	表示できます。
-onsonly	必要に応じて、このパラメータを使用して、Oracle Notification Service 構成を表示できます。
-envs "name_list"	オプションで、環境変数の名前のカンマ区切りリストを、二重引用符("")で囲んで指定できます。 このパラメータを使用しない場合は、ノード・アプリケーションに関連付けられているすべての環境変数の値が表示されます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例は、ノード・アプリケーションのすべての環境変数をリストします。

```
$ srvctl getenv nodeapps -viponly
```

srvctl modify nodeapps

ノード・アプリケーションの構成の変更

構文

このコマンドと、次の構文モデルの1つを一緒に使用して、特定のノードとVIP、または特定のサブネットとネットマスクを指定します。

```
srvctl modify nodeapps {[-node node_name -address {vip_name|vip_address}/
  netmask[/if1[|if2|...]] [-skip]} [-nettype network_type] [-emport em_port]
    [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]
    [-remoteservers host:[port][,...]] [-verbose]
    [-clientdata file] [-pingtarget "ping_target_list"]

srvctl modify nodeapps [-subnet subnet/netmask[/if1[|if2|...]]]
    [-nettype network_type] [-emport em_port]
    [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]
    [-remoteservers host:[port][,host:port,...]] [-verbose]
    [-clientdata file] [-pingtarget "ping_target_list"]
```

パラメータ

表A-54 srvctl modify nodeappsコマンドのパラメータ

パラメータ	説明

パラメータ	説明
-node node_name	変更するノード・アプリケーションが存在するノードの名前を指定します。
-address {vip_name|vip_address}/netmask[/if1[|if2|...]]	ノード・レベル仮想 IP 名またはアドレスを指定します。名前または IP で指定されたアドレスは、デフォルトびネットワークのサブネット番号に一致する必要があります。 ノート: アップグレード構成および新規の DHCP 以外の構成にこのパラメータを使用する必要があります
-skip	必要に応じて、このパラメータを使用して、VIP アドレスの到達可能性の確認をスキップできます。
-subnet subnet/netmask[/if1[|if2|...]]	ノード名およびアドレスを指定するかわりに、パブリック・ネットワークのサブネット番号を指定できます。ネットマスクとインタフェースを指定すると、デフォルト・ネットワークのネットマスクとインタフェースが変更されます。また、netmask オプションの値を指定する場合は、各ネットワークの最初のノードにのみ指定する必要があります。
-nettype network_type	必要に応じて、ネットワーク・サーバー・タイプを static、dhcp または mixed に変更できます。
-emport em_port	必要に応じて、Oracle Enterprise Manager がリスニングするローカル・ポートを変更できます。 ノート: この属性は、オンライン・リソース属性変更を使用して変更することもできます。
-onslocalport ons_local_port	必要に応じて、Oracle Notification Service デーモンがローカル・クライアント接続をリスニングするポートを変更できます。 ノート: ● ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。 ● リソースがオンラインの間は、リソースを再起動せずに、ローカル・ポートを変更できます。

パラメータ	説明
-onsremoteport ons_remote_port	必要に応じて、Oracle Notification Service デーモンがリモート・ホストからの接続をリスニングするポートを変更できます。 ノート: ● ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。 ● リソースがオンラインの間は、リソースを再起動せずに、リモート・ポートを変更できます。
-remoteservers host:[port][,...]	必要に応じて、Oracle Notification Service ネットワークには含まれていてもクラスタには含まれていないリモート・ホストの host:[port ペアのカンマ区切りリストを変更できます。リモート・ホストの port を指定しなかった場合は、ons_remote_port に指定した値が使用されます。
-clientdata file	必要に応じて、インポートするウォレットを含むファイル、または Oracle Notification Service 通信を保護するための SSL に使用されるウォレットを削除する空の文字列を指定できます。
-pingtarget "ping_target_list"	必要に応じて、ping する IP またはホスト名を二重引用符("")で囲んだカンマ区切りリストを指定できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、アプリケーションVIPに100.200.300.40、ネットワーク・インタフェースeth0のサブネットマスクに255.255.255.0を使用するようにmynode1のnodeappsリソースを変更します。

```
$ srvctl modify nodeapps -node mynode1 -addr 100.200.300.40/255.255.255.0/eth0
```

srvctl remove nodeapps

ノード・アプリケーション構成を削除します。

構文

```
srvctl remove nodeapps [-force] [-noprompt] [-verbose]
```

パラメータ

表A-55 srvctl remove nodeappsコマンドのパラメータ

パラメータ	説明
-force	必要に応じて、このパラメータを使用して、依存性に関係なくノード・アプリケーション構成を強制的に削除できます。
-noprompt	必要に応じて、このパラメータを使用してプロンプトを抑止できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- このコマンドを実行するには、完全な管理権限が必要です。LinuxシステムおよびUNIXシステムではrootでログインし、Windowsシステムでは管理者権限を持つユーザーとしてログインする必要があります。

srvctl setenv nodeapps

ノード・アプリケーション構成の環境変数を設定します。

構文

```
srvctl setenv nodeapps {-envs "name=val[,...]" | -env "name=val"}
   [-viponly] [-gsdonly] [-onsonly] [-verbose]
```

パラメータ

表A-56 srvctl setenv nodeappsコマンドのパラメータ

パラメータ	説明
-envs "name=val[,...]"	このパラメータは、環境変数の名前/値ペアを二重引用符("")で囲んだカンマ区切りリストを指定する場合に使用します。
-env "name=val"	または、このパラメータは、カンマやその他の特殊文字を含んだ値を二重引用符("")で囲んで単一の環境変数に設定できるようにする場合にも使用できます。

パラメータ	説明
-viponly	必要に応じて、このパラメータを使用して、VIP 構成のみを変更できます。
-gsdonly	必要に応じて、このパラメータを使用して、GSD 構成のみを変更できます。
-onsonly	必要に応じて、このパラメータを使用して、ONS デーモン構成のみを変更できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、すべてのノード・アプリケーションのCLASSPATH環境変数を設定します。

```
$ srvctl setenv nodeapps -env "CLASSPATH=/usr/local/jdk/jre/rt.jar" -verbose
```

srvctl start nodeapps

クラスタの1つのノードまたはすべてのノードでノード・レベル・アプリケーションを起動します。

構文

```
srvctl start nodeapps [-node node_name] [-gsdonly] [-adminhelper] [-verbose]
```

パラメータ

表A-57 srvctl start nodeappsコマンドのパラメータ

パラメータ	説明
-node node_name	必要に応じて、ノード・レベル・アプリケーションを起動するノードを指定できます。 このパラメータを使用しない場合は、クラスタ内にあるすべてのアクティブなノードでノード・アプリケーションが起動されます。
-gsdonly	必要に応じて、このパラメータを使用して、すべてのノード・アプリケーションではなく GSD のみを起動できます。

パラメータ	説明
-adminhelper	必要に応じて、このパラメータを使用して、すべてのノード・アプリケーションではなくAdministratorヘルパーのみを起動できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

srvctl status nodeapps

ノード・アプリケーションのステータスを表示します。

構文

```
srvctl status nodeapps [-node node_name]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 必要に応じて、ノード・アプリケーションのステータスを表示するノードを指定できます。

srvctl stop nodeapps

クラスタのノードでノード・レベル・アプリケーションを停止します。

構文

```
srvctl stop nodeapps [-node node_name] [-gsdonly] [-adminhelper] [-force]
   [-relocate] [-verbose]
```

パラメータ

表A-58 srvctl stop nodeappsコマンドのパラメータ

パラメータ	説明
-node node_name	必要に応じて、このパラメータを使用して、ノード・アプリケーションを停止するノードを指定できます。 このパラメータを使用しない場合は、クラスタ内でアクティブなすべてのノードでノード・アプリケーションが停止されます。
-gsdonly	必要に応じて、このパラメータを使用して、すべてのノード・アプ

パラメータ	説明
	リケーションではなく GSD のみを停止できます。
-adminhelper	必要に応じて、このパラメータを使用して、すべてのノード・アプリケーションではなく Administrator ヘルパーのみを停止できます。
-force	必要に応じて、このパラメータを使用して、依存性に関係なく、ノード・アプリケーションを停止できます。
-relocate	必要に応じて、このパラメータを使用して、VIP および依存している可能性があるサービスを再配置できます。 ノート: このパラメータを使用する場合は、-node node_name パラメータも指定する必要があります。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

srvctl unsetenv nodeapps

ノード・アプリケーションの環境構成の設定を解除します。

構文

```
srvctl unsetenv nodeapps -envs "name_list" [-viponly] [-gsdonly] [-onsonly]
   [-verbose]
```

パラメータ

表A-59 srvctl unsetenv nodeappsコマンドのパラメータ

パラメータ	説明
-envs "name_list"	設定を解除する環境変数の名前を二重引用符(“”)で囲んだカンマ区切りリストを指定します。
-viponly	必要に応じて、このパラメータを使用して、VIP 構成の設定のみを解除できます。
-gsdonly	必要に応じて、このパラメータを使用して、GSD 構成の設定

パラメータ	説明
	のみを解除できます。
-onsonly	必要に応じて、このパラメータを使用して、ONS デーモン構成の設定のみを解除できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

例

次の例では、指定したノード・アプリケーションの環境構成の設定を解除します。

```
$ srvctl unsetenv nodeapps -envs "test_var1,test_var2"
```

srvctl add ons

Oracle Notification ServiceデーモンをOracle Restart構成に追加します。

構文

```
srvctl add ons [-emport em_port] [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]
   [-remoteservers host[:port][,host[:port]...]]
   [-clientcluster cluster_name] [-clientdata filename]
```

パラメータ

表A-60 srvctl add onsコマンドのパラメータ

パラメータ	説明
-emport em_port	Oracle Enterprise Manager のローカル・リスニング・ポートデフォルトのポート番号は 2016 です。
-onslocalport ons_local_port	必要に応じて、Oracle Notification Service デーモンがローカル・クライアント接続をリスニングするポートを指定できます。 ノート: ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。
-onsremoteport ons_remote_port	必要に応じて、Oracle Notification Service デーモンがリモート・ホストからの接続をリスニングするポートを指定できます。 ノート: ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。
-remoteservers host[:port][host[:port]...]	必要に応じて、Oracle Notification Service ネットワークには含まれていても Oracle Clusterware クラスタには含まれていないリモート・ホストの host:port ペアのカンマ区

パラメータ	説明
	切りリストを指定できます。 ノート: リモート・ホストに port を指定しなかった場合は、ons_remote_port が使用されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。
-clientdata filename	資格証明データを書き込むファイルへのパスを指定します。

使用上のノート

このコマンドは、Oracle Restartでのみ使用できます。

例

次に、このコマンドの例を示します。

```
$ srvctl add ons -onslocalprt 6200
```

srvctl config ons

Oracle Notification Serviceデーモンの構成情報を表示します。

構文

```
srvctl config ons [-all] [-clientcluster cluster_name]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- すべてのONSデーモンまたは特定のクライアント・クラスタのONSデーモンの構成を表示できます。

srvctl disable ons

Oracle RestartインストールのOracle Notification Service (ONS)デーモンを無効にします。

構文

```
srvctl disable ons [-clientcluster cluster_name] [-verbose]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- すべてのONSデーモンまたは特定のクライアント・クラスタのONSデーモンを無効にできます。
- 必要に応じて、-verboseパラメータを使用して、詳細な出力を表示できます。

srvctl enable ons

Oracle Notification Serviceデーモンを有効化します。

構文

```
srvctl enable ons [-clientcluster cluster_name] [-verbose]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- すべてのONSデーモンまたは特定のクライアント・クラスタのONSデーモンを有効にできます。
- 必要に応じて、-verboseパラメータを使用して、詳細な出力を表示できます。

srvctl modify ons

Oracle Restartに登録されたOracle Notification Serviceデーモンで使用するポートを変更します。

構文

```
srvctl modify ons [-emport em_port] [-onslocalprt ons_local_port] [-onsremoteport ons_remote_port]
  [-remoteservers host[:port][,host[:port],...]]
  [-clientcluster cluster_name] [-verbose]
```

パラメータ

表A-61 srvctl modify onsコマンドのパラメータ

パラメータ	説明
-emport em_port	必要に応じて、Oracle Enterprise Manager がリスニングするローカル・ポートを指定できます。デフォルト・ポートは 2016 です。
-onslocalprt ons_local_port	必要に応じて、Oracle Notification Service デーモンがローカル・クライアント接続をリスニングするポートを変更できます。 ノート: ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。
-onsremoteport ons_remote_port	必要に応じて、Oracle Notification Service デーモンがリモート・ホストからの接続をリスニングするポートを変更できます。 ノート: ローカル・ポートとリモート・ポートは、それぞれ一意である必要があります。

パラメータ	説明
-remoteservers host[:port][,host[:port],...]	必要に応じて、Oracle Notification Service ネットワークには含まれていても Oracle Clusterware クラスタには含まれていないリモート・ホストの host:port ペアのカンマ区切りリストを指定できます。 ノート: リモート・ホストの port を指定しなかった場合は、ons_remote_port の値が使用されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Restartでのみ使用できます。

例

次に、このコマンドの例を示します。

```
$ srvctl modify ons -onslocalprt 6203
```

srvctl remove ons

Oracle Grid InfrastructureホームからOracle Notification Serviceを削除します。

構文

```
srvctl remove ons [-clientcluster cluster_name] [-force] [-verbose]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- 共有SCAN機能を使用する場合、-clientclusterパラメータを使用して、共有SCANリスナーを実行しているクラスタの名前を指定します。
- 必要に応じて、-forceパラメータを使用して、依存性に関係なく、Oracle Notification Serviceを削除できます。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

srvctl start ons

Oracle Notification Serviceデーモンを起動します。

構文

```
srvctl start ons [-clientcluster cluster_name] [-verbose]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- すべてのONSデーモンまたは特定のクライアント・クラスタのONSデーモンを有効にできます。
- 必要に応じて、-verboseパラメータを使用して、詳細な出力を表示できます。

srvctl status ons

Oracle Notification Serviceデーモンの現在の状態を表示します。

構文

```
srvctl status ons [-clientcluster cluster_name]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- すべてのONSデーモンまたは特定のクライアント・クラスタのONSデーモンのステータスを表示できます。

srvctl stop ons

Oracle Notification Serviceデーモンを停止します。

構文

```
srvctl stop ons [-clientcluster cluster_name] [-force]
```

使用上のノート

- このコマンドは、Oracle Restartでのみ使用できます。
- すべてのONSデーモンまたは特定のクライアント・クラスタのONSデーモンを停止できます。
- 必要に応じて、-forceパラメータを使用して、依存性に関係なくONSデーモンを停止できます。

srvctl add scan

指定のSCANにOracle Clusterwareリソースを追加します。

構文

```
srvctl add scan -scanname scan_name [-netnum network_number]
```

パラメータ

表A-62 srvctl add scanコマンドのパラメータ

パラメータ	説明
-scanname scan_name	ドメイン名を含む完全修飾されたホスト名。ネットワークが動的な場合、完全修飾されたドメイン名を使用する必要はありませんが、これを使用する場合は、ドメインは GNS サブ

パラメータ	説明
	ドメインにする必要があります。 ノート: この属性はオンライン・リソース属性変更を使用して変更できます。
-netnum network_number	SCAN VIP を取得するネットワーク番号(オプション)。このパラメータを指定しない場合、nodeapps VIP の取得元と同じデフォルト・ネットワークから SCAN VIP が取得されます。

使用上のノート

- このコマンドは、SCANが解決されるIPアドレスの数と同じ数のSCAN VIPリソースを作成するか、またはnetwork_numberによって動的ネットワークおよびOracle GNS構成が識別された場合は3つのSCAN VIPリソースを作成します。
- 静的ネットワークの場合、DNSでSCANが解決されるアドレスは、サブネットのアドレス・タイプと一致する必要があります。
- IPv4ネットワークの場合、SCANはIPv4アドレスに解決される必要があります。
- このコマンドはOracle Clusterwareでのみ使用可能です。

例

次に、このコマンドの例を示します。

```
# srvctl add scan -scanname scan.mycluster.example.com
```

srvctl config scan

すべてのSCAN VIP(デフォルト)、またはordinal_numberで識別される特定のSCAN VIPの構成情報を表示します。

構文

```
srvctl config scan [[-netnum network_number] [-scannumber ordinal_number] | -all]
```

パラメータ

表A-63 srvctl config scanコマンドのパラメータ

パラメータ	説明
-netnum network_number	このパラメータを使用して、特定の SCAN VIP の構成を表示します。
-scannumber ordinal_number	このパラメータを使用して、構成を表示する 3 つの SCAN VIP のいずれか(1 から 3 の値を使用)を指定します。
-all	ネットワーク番号または序数を指定するかわりに、このパラメー

パラメータ	説明
	タを使用して、すべての SCAN VIP の構成を表示できます。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

このコマンドによって、次のような出力が返されます。

```
$ srvctl config scan -scannumber 1
SCAN name: mjk12700890090-r, Network: 1
Subnet IPv4: 198.51.100.1/203.0.113.46/eth0, static
Subnet IPv6:
SCAN 1 IPv4 VIP: 198.51.100.195
SCAN VIP is enabled.
SCAN VIP is individually enabled on nodes:
SCAN VIP is individually disabled on nodes:
```

srvctl disable scan

すべてのSCAN VIP(デフォルト)、またはordinal_numberで識別される特定のSCAN VIPを無効化します。

構文

```
srvctl disable scan [-scannumber ordinal_number]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 必要に応じて、-scannumberパラメータを使用して、3つのSCAN VIPのうち無効化する1つを指定できます。パラメータ値は1から3の範囲になります。

例

次の例では、最初のSCAN VIPを無効化します。

```
$ srvctl disable scan -scannumber 1
```

srvctl enable scan

すべてのSCAN VIP (デフォルト)、またはその序数で識別される特定のSCAN VIPを有効化します。

構文

```
srvctl enable scan [-scannumber ordinal_number]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 必要に応じて、-scannumberパラメータを使用して、3つのSCAN VIPのうち有効化する1つを指定できます。パラメー

タ値の範囲は1から3です。

例

次の例では、最初のSCAN VIPを有効化します。

```
$ srvctl enable scan -scannumber 1
```

srvctl modify scan

DNSで指定するscan_nameを調べると戻されるIPアドレス数に一致するように、SCAN VIPの数を変更します。

このコマンドは、IPアドレスの追加、変更または削除に伴ってDNSが変更され、それに合わせてOracle Clusterwareリソース構成を調整する必要がある場合に使用します。

構文

```
srvctl modify scan -scanname scan_name [-netnum network_number]
```

パラメータ

表A-64 srvctl modify scanコマンドのパラメータ

パラメータ	説明
-scanname scan_name	変更対象の SCAN VIP に解決される SCAN 名を識別します。 ノート: この属性はオンライン・リソース属性変更を使用して変更できます。
-netnum network_number	VIP を取得するネットワーク番号(オプション)。指定しなかった場合は、nodeapps VIP が取得されるネットワークと同じデフォルトのネットワークから VIP が取得されます。

例

システムに現在scan_name1という名前のSCANが存在し、DNSで単一のIPアドレスに解決されるとします。DNSでSCAN scan_name1を変更して3つのIPアドレスに解決されるようにした場合、次のコマンドを使用して追加のSCAN VIPリソースを作成します。

```
$ srvctl modify scan -scanname scan_name1
```

srvctl predict scan

SCAN障害の結果を予測します。

構文

```
srvctl predict scan -scannumber ordinal_number [-verbose]
```

使用上のノート

- 障害をシミュレートするSCAN VIPを識別する序数を指定します。このパラメータに指定できる値の範囲は1から3です。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

ここで、コマンドに関する追加情報を追加します。

srvctl relocate scan

現行のホスティング・ノードからクラスタ内の別のノードに特定のSCAN VIPを再配置します。

構文

```
srvctl relocate scan -scannumber ordinal_number [-node node_name]
```

パラメータ

表A-65 srvctl relocate scanコマンドのパラメータ

パラメータ	説明
-scannumber ordinal_number	再配置する SCAN VIP を識別する序数を指定します。このパラメータに指定できる値の範囲は 1 から 3 です。
-node node_name	必要に応じて、SRVCTL が SCAN VIP を再配置する単一ノードの名前を指定できます。 このパラメータを使用しない場合は、SCAN VIP の再配置先ノードが自動的に選択されます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、最初のSCAN VIPをnode1に再配置します。

```
$ srvctl relocate scan -scannumber 1 -node node1
```

srvctl remove scan

すべてのSCAN VIPからOracle Clusterwareリソースを削除します。

構文

```
srvctl remove scan [-netnum network_number] [-force] [-noprompt]
```

パラメータ

表A-66 srvctl remove scanコマンドのパラメータ

パラメータ	説明
-netnum network_number	VIP を取得するネットワーク番号(オプション)。指定しなかった場合は、nodeapps VIP が取得されるネットワークと同じデフォルトのネットワークから VIP が取得されます。
-force	SCAN VIP に依存する SCAN リスナーが実行中であっても、それらの SCAN VIP を削除します。
-noprompt	このパラメータは、すべてのプロンプトを抑止する場合に使用します。

使用上のノート

-forceオプションを使用すると、実行中のSCAN VIPは依存リソースが削除されるまで停止されないため、手動によるクリーンアップが必要になる場合があります。

例

次に、このコマンドの例を示します。

```
$ srvctl remove scan -force
```

srvctl start scan

クラスタのすべてのノードまたは特定のノードで、すべてのSCAN VIP(デフォルト)または特定のSCAN VIPを起動します。

構文

```
srvctl start scan [-scannumber ordinal_number] [-node node_name]
```

パラメータ

表A-67 srvctl start scanコマンドのパラメータ

パラメータ	説明
-scannumber ordinal_number	必要に応じて、起動する SCAN VIP を識別する序数を指定できます。このパラメータに指定できる値の範囲は 1 から 3 です。 このパラメータを使用しない場合は、すべての SCAN VIP が起動されます。
-node node_name	必要に応じて、起動する SCAN VIP が存在する単一ノードの名前を指定できます。

パラメータ	説明
	このパラメータを指定しない場合は、クラスタのすべてのノードで SCAN VIP が起動されます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、crm1ノード上の序数1で識別されるSCAN VIPを起動します。

```
$ srvctl start scan -scannumber 1 -node crm1
```

srvctl status scan

すべてのSCAN VIP(デフォルト)または特定のSCAN VIPのステータスを表示します。

構文

```
srvctl status scan [-scannumber ordinal_number] [-verbose]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 必要に応じて、ステータスを表示する特定のSCAN VIPを識別する序数を指定できます。このパラメータに指定できる値の範囲は1から3です。このパラメータを使用しない場合は、クラスタのすべてのSCAN VIPのステータスが表示されます。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

srvctl stop scan

実行中または起動中のすべてのSCAN VIP (デフォルト)を停止するか、序数で識別される特定のSCAN VIPを停止します。

構文

```
srvctl stop scan [-scannumber ordinal_number] [-force]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 必要に応じて、停止するSCAN VIPを識別する序数を指定できます。このパラメータに指定できる値の範囲は1から3です。このパラメータを使用しない場合は、すべてのSCAN VIPが停止されます。
- 必要に応じて、-forceパラメータを使用して、依存性に関係なく、SCAN VIPを停止できます。

例

次の例では、序数1で識別されるSCAN VIPを停止します。

```
$ srvctl stop scan -scannumber 1
```

srvctl add scan_listener

SCANリスナー用のOracle Clusterwareリソースを追加します。

構文

```
srvctl add scan_listener [-netnum network_number] [-listener lsnr_name_prefix] [-skip]
  [-endpoints "[TCP:]port_list[/IPC:key][/NMP:pipe_name]
  [/{TCPS|SDP|EXADIRECT}port_list]"]
  [-invitednodes "node_list"] [-invitedsubnets "subnet_list"]
  [-clientcluster cluster_name] [-clientdata <filename>]
```

パラメータ

表A-68 srvctl add scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	SCAN VIP を取得するネットワーク番号(オプション)。このパラメータを指定しない場合、nodeapps VIP の取得元と同じデフォルト・ネットワークから SCAN VIP が取得されます。
-listener lsnr_name_prefix	SCAN リスナー名の接頭辞。
-skip	ポートのチェックをスキップします。
-endpoints "[TCP:]port_list[/IPC:key][/NMP:pipe_name][/{TCPS|SDP|EXADIRECT}port_list]"	SCAN リスナーのプロトコル仕様。port_list を使用して、TCP ポートまたは SCAN リスナー・エンドポイントのカンマ区切りリストを指定します。 TCPS ポート、SDP ポート、および EXADIRECT ポートのエンドポイントを指定することもできます。 ノート: この属性はオンライン・リソース属性変更を使用して変更できます。
-invitednodes "node_list"	SCAN リスナーに登録できるクラスタ外部からのホスト名のカンマ区切りリスト。
-invitedsubnets "subnet_list"	SCAN リスナーに登録できるクラスタ外部からのサブネットのカンマ区切りリスト。CIDR 表記またはワイルドカード(192.168.*など)を使用してサブネットを指定できます。
-clientcluster cluster_name	共有する SCAN リスナーを実行しているクラスタの名前。
-clientdata file_name	共有 SCAN リスナーを実行しているクラスタの名前。

使用上のノート

- 作成されるSCANリスナーのリソースの数は、SCAN VIPリソースの数と同じになります。

- このコマンドはOracle Clusterwareでのみ使用可能です。

例

次に、このコマンドの例を示します。

```
# srvctl add scan_listener -listener myscanlistener
```

srvctl config scan_listener

すべてのSCANリスナー(デフォルト)、またはネットワーク番号あるいはordinal_numberで識別される特定のリスナーの構成情報を表示します。

構文

```
srvctl config scan_listener [[-netnum network_number] [-scannumber ordinal_number]
 [-clientcluster cluster_name] | -all]
```

パラメータ

表A-69 srvctl config scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	このパラメータを使用して、特定の SCAN VIP のリスナーの構成を表示します。
-scannumber ordinal_number	このパラメータを使用して、リスナーの構成を表示する 3 つの SCAN VIP のいずれか(1 から 3 の値を使用)を指定します。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。
-all	ネットワーク番号または序数を指定するかわりに、このパラメータを使用して、すべての SCAN VIP のリスナーの構成を表示できます。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

このコマンドによって、次のような出力が返されます。

```
$ srvctl config scan_listener -scannumber 1
SCAN Listener LISTENER_SCAN1 exists. Port: TCP:1529
Registration invited nodes:
Registration invited subnets:
SCAN Listener is enabled.
```

```
SCAN Listener is individually enabled on nodes:
SCAN Listener is individually disabled on nodes:
```

srvctl disable scan_listener

すべてのSCANリスナー(デフォルト)、または序数またはクライアント・クラスタで識別される特定のリスナーを無効化します。

構文

```
srvctl disable scan_listener [-netnum network_number] [-scannumber ordinal_number]
  [-clientcluster cluster_name]
```

パラメータ

表A-70 srvctl disable scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	このパラメータは、特定のネットワーク番号の SCAN リスナーを無効にする場合に使用します。
-scannumber ordinal_number	このパラメータは、1 から 3 の値を使用して、3 つの SCAN VIP のいずれかを無効にする場合に使用します。このパラメータを使用しない場合は、すべての SCAN リスナーが無効化されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

次の例では、1で識別されるSCANリスナーを無効化します。

```
$ srvctl disable scan_listener -scannumber 1
```

srvctl enable scan_listener

すべてのSCANリスナー(デフォルト)、またはその序数で識別される特定のリスナーを有効化します。

構文

```
srvctl enable scan_listener [-netnum network_number] [-scannumber ordinal_number]
  [-clientcluster <cluster_name>]
```

パラメータ

表A-71 srvctl enable scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	このパラメータは、特定の SCAN VIP のリスナーを有効にする場合に使用します。
-scannumber ordinal_number	このパラメータは、1 から 3 の値を使用して、3 つの SCAN VIP のいずれかを有効にする場合に使用します。このパラメータを使用しない場合は、すべての SCAN リスナーが有効になります。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

次の例では、1で識別されるSCANリスナーを有効化します。

```
$ srvctl enable scan_listener -scannumber 1
```

srvctl modify scan_listener

SCAN VIPのリスナーに一致するようにSCANリスナーを変更するか、SCANリスナー・エンドポイントまたはサービス登録の制限を変更します。

構文

```
srvctl modify scan_listener {-update | -endpoints [TCP:]port_list[/IPC:key]
     [/NMP:pipe_name][/{TCPS|SDP|EXADIRECT}port_list]"} [-invitednodes "node_list"]
     [-invitedsubnets "subnet_list"] [-clientcluster cluster_name]
```

パラメータ

表A-72 srvctl modify scan_listenerコマンドのパラメータ

パラメータ	説明
-update	このパラメータを使用して、現行の SCAN VIP の構成に一致するように SCAN リスナーの構成を更新します。このパラメータは、SCAN VIP リソースの数と一致するように、新規リソースを追加するか、既存の SCAN リスナー・リソースを削除します。
-endpoints "[TCP:]port_list[/IPC:key] [/NMP:pipe_name][/{TCPS|SDP|EXADIRECT}port_list]"]	SCAN リスナーのプロトコル仕様。port_list を使用して、TCP ポートまたはリスナー・エンドポイントのカンマ区切

パラメータ	説明
	りリストを指定します。 TCPS ポート、SDP ポート、および EXADIRECT ポートのエンドポイントを指定することもできます。
-invitednodes "node_list"	SCAN リスナーに登録できるクラスタの外側からホスト名のカンマ区切りリストを指定するには、このパラメータを使用します。
-invitedsubnets "subnet_list"	SCAN リスナーに登録できるクラスタの外側からサブネットのカンマ区切りリストを指定するには、このパラメータを使用します。CIDR 表記またはワイルドカード(192.168.*など)を使用してサブネットを指定できます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。

例

システムに現在scan_name1という名前のSCANが存在し、最近そのDNSエントリを1つではなく3つのIPアドレスに解決されるように変更したとします。srvctl modify scanコマンドを実行して追加のSCAN VIPリソースを作成した後に、次のコマンドを使用して、この2つの追加のSCAN VIPにあわせて2つの追加のSCANリスナーのOracle Clusterwareリソースを作成します。

```
$ srvctl modify scan_listener -update
```

srvctl predict scan_listener

SCANリスナー障害の結果を予測します。

構文

```
srvctl predict scan_listener -scannumber ordinal_number [-verbose]
```

使用上のノート

- -scannumberパラメータは、3つのSCANリスナーのうち、障害の結果を予測する1つを指定する場合に使用します。このパラメータに指定できる値の範囲は1から3です。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

srvctl relocate scan_listener

現行のホスティング・ノードからクラスタ内の別のノードに特定のSCANリスナーを再配置します。

構文

```
srvctl relocate scan_listener -scannumber ordinal_number [-node node_name]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 再配置するSCANリスナーを識別する序数を指定します。このパラメータに指定できる値の範囲は1から3です。
- 必要に応じて、SCANリスナーを再配置する単一ノードの名前を指定できます。このパラメータを指定しない場合は、SCANリスナーの再配置先ノードが自動的に選択されます。

例

次の例では、3で識別されるSCANリスナーをクラスタのnode2に再配置します。

```
$ srvctl relocate scan_listener -scannumber 3 -node node2
```

srvctl remove scan_listener

すべてのSCANリスナーからOracle Clusterwareリソースを削除します。

構文

```
srvctl remove scan_listener [-netnum network_number] [-clientcluster cluster_name]
  [-force] [-noprompt]
```

パラメータ

表A-73 srvctl remove scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	SCAN VIP を取得するネットワーク番号(オプション)。このパラメータを指定しない場合、nodeapps VIP の取得元と同じデフォルト・ネットワークから SCAN VIP が取得されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。
-force	実行中の SCAN リスナーを停止せずに削除します。
-noprompt	このパラメータは、すべてのプロンプトを抑止する場合に使用します。

例

次に、このコマンドの例を示します。

```
$ srvctl remove scan_listener -force
```

srvctl start scan_listener

クラスタのすべてのノードまたは特定のノードで、すべてのSCANリスナー(デフォルト)または特定のSCANリスナーを起動します。

構文

```
srvctl start scan_listener [-netnum network_number] [-scannumber ordinal_number]
  [-node node_name] [-clientcluster cluster_name]
```

パラメータ

表A-74 srvctl start scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	このパラメータは、特定のネットワーク番号の SCAN リスナーを起動する場合に使用します。
-scannumber ordinal_number	このパラメータは、1 から 3 の値を使用して、3 つの SCAN VIP のいずれかを起動する場合に使用します。このパラメータを使用しない場合は、すべての SCAN リスナーが起動されます。
-node node_name	SCAN リスナーを起動する単一ノードの名前を指定します。このパラメータを使用しない場合は、クラスタのすべてのノードで SCAN リスナーが起動されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

次の例では、1で識別されるSCANリスナーを起動します。

```
$ srvctl start scan_listener -scannumber 1
```

srvctl status scan_listener

すべてのSCANリスナー(デフォルト)または特定のリスナーのステータスを表示します。

構文

```
srvctl status scan_listener [[-netnum network_number] [-scannumber ordinal_number]
   | [-clientcluster cluster_name] | -all] [-verbose]
```

パラメータ

表A-75 srvctl status scan_listenerコマンドのパラメータ

パラメータ	説明

パラメータ	説明
-netnum network_number	ネットワーク番号。デフォルトのネットワーク番号は 1 です。
-scannumber ordinal_number	特定の SCAN リスナーを識別する序数。このパラメータに指定できる値の範囲は 1 から 3 です。このパラメータを使用しない場合は、クラスタのすべての SCAN リスナーのステータスが表示されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。
-all	すべてのネットワークの SCAN リスナーのステータスを表示します。
-verbose	詳細情報を表示します。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

srvctl stop scan_listener

実行中または起動中のすべてのSCANリスナー(デフォルト)を停止するか、序数で識別される特定のリスナーを停止します。

構文

```
srvctl stop scan_listener [-netnum network_number] [-scannumber ordinal_number]
  [-clientcluster cluster_name] [-force]
```

パラメータ

表A-76 srvctl stop scan_listenerコマンドのパラメータ

パラメータ	説明
-netnum network_number	このパラメータは、特定のネットワーク番号の SCAN リスナーを停止する場合に使用します。
-scannumber ordinal_number	このパラメータは、1 から 3 の値を使用して、3 つの SCAN VIP のいずれかを停止する場合に使用します。このパラメータを使用しない場合は、すべての SCAN リスナーが停止されます。
-clientcluster cluster_name	共有 SCAN リスナーを実行しているクラスタの名前。
-force	依存性に関係なく SCAN リスナーを停止します。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

次の例では、1で識別されるSCANリスナーを停止します。

```
$ srvctl stop scan_listener -scannumber 1
```

srvctl update scan_listener

新しいエンドポイントをリスニングするようSCANリスナーを更新します。

構文

```
srvctl update scan_listener
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- このコマンドには、-helpを除き、その他のパラメータは指定できません。

srvctl relocate server

サーバーをクラスタのサーバー・プールに再配置します。

構文

```
srvctl relocate server -servers "server_name_list" -serverpool pool_name
   [-eval] [-force]
```

パラメータ

表A-77 srvctl relocate serverコマンドのパラメータ

パラメータ	説明
-servers "server_name_list"	別のサーバー・プールに再配置する、単一のサーバー名、または二重引用符(“”)で囲んだ複数のサーバー名のカンマ区切りリストを指定します。
-serverpool pool_name	サーバーの移動先のサーバー・プールの名前を指定します。
-eval	オプションで、このパラメータを使用するとコマンドがシステムに及ぼす影響を仮定的に評価できます。
-force	必要に応じて、このパラメータを使用して、いくつかのリソースを停止することになってもサーバーの再配置を強制できます。

例

次の例では、2台のサーバーを別のサーバー・プールに再配置します。

```
$ srvctl relocate server -servers "server1, server2" -serverpool sp3
```

srvctl status server

特定のサーバーの現在の状態を表示します。

構文

```
srvctl status server -server "server_name_list" [-detail]
```

使用上のノート

- -serverパラメータは、ステータスを確認する単一のサーバー名か、または複数のサーバー名を二重引用符("")で囲んだカンマ区切りリストを指定する場合に使用します。
- 必要に応じて、-detailパラメータを使用して、詳細なステータス情報を出力できます。

srvctl add service

データベースにサービスを追加し、それらのサービスをインスタンスに割り当てます。

構文

次の構文モデルのいずれかとともに、このコマンドを使用します。

サービスをポリシー管理データベースに追加するには:

```
srvctl add service -db db_unique_name -service service_name_list [-eval]
   -serverpool server_pool [-pdb pluggable_database]
   [-cardinality {UNIFORM | SINGLETON}] [-edition edition_name]
   [-netnum network_number] [-role
"[PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]"]
   [-policy {AUTOMATIC | MANUAL}] [-notification {TRUE | FALSE}] [-rfpool pool_name]
   [-clbgoal {SHORT | LONG}] [-failovertype {NONE|SESSION|SELECT|TRANSACTION|AUTO}]
   [-rlbgoal {NONE | SERVICE_TIME | THROUGHPUT}] [-dtp {TRUE | FALSE}]
   [-failovermethod {NONE | BASIC}] [-failoverretry failover_retries]
   [-drain_timeout timeout] [-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}]
   [-failover_restore {NONE|LEVEL1|AUTO}] [-failoverdelay failover_delay]
   [-gsmflags gsm_flags] [-sql_translation_profile sql_translation_profile]
   [-global {TRUE | FALSE}] [-maxlag max_lag_time] [-commit_outcome {TRUE|FALSE}]
   [-retention retention_time] [-replay_init_time replay_initiation_time]
   [-session_state {STATIC | DYNAMIC | AUTO}] [-pqservice pq_service]
   [-pqpool pq_pool] [-css_critical {yes | no}] [-force]
```

サービスを管理者管理データベースに追加するには:

```
srvctl add service -database db_unique_name -service service_name_list
   [-pdb pluggable_database] [-eval]
   [-preferred preferred_list] [-available available_list] [-failback {YES | NO}]
   [-netnum network_number] [-tafpolicy {BASIC | NONE}]
   [-edition edition_name]
   [-role "[PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]"
   [-policy {AUTOMATIC | MANUAL}] [-notification {TRUE | FALSE}]
   [-clbgoal {SHORT | LONG}] [-failovertype {NONE|SESSION|SELECT|TRANSACTION|AUTO}]
```

```
   [-rlbgoal {NONE | SERVICE_TIME | THROUGHPUT}] [-dtp {TRUE | FALSE}]
   [-failovermethod {NONE | BASIC}] [-failoverretry failover_retries]
   [-drain_timeout timeout] [-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}]
   [-failover_restore {NONE|LEVEL1|AUTO}] [-failoverdelay failover_delay]
   [-sql_translation_profile sql_translation_profile]
   [-global {TRUE | FALSE}] [-maxlag max_lag_time] [-commit_outcome {TRUE|FALSE}]
   [-retention retention_time] [-replay_init_time replay_initiation_time]
   [-session_state {STATIC|DYNAMIC|AUTO}] [-force] [-verbose]
```

既存のサービスの優先リストおよび使用可能リストを更新するには:

```
srvctl add service -db db_unique_name -service service_name_list
   -update {-prefered preferred_list | -available available_list} [-force]
   [-verbose]
```

パラメータ

次の表に、srvctl add serviceのすべてのパラメータとその説明を示し、Oracle RACデータベースまたは非クラスタ・データベースにサービスを追加するときにそれらのパラメータを使用できるかどうかを示します。

表A-78 srvctl add serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前。
-service service_name_list	同じサービスを提供する複数のデータベースにわたって接続を拡大しない場合は、service_name.service_domain はクラスタ内で一意である必要があります。サービス名(sales.example.com など)の一部としてサービス・ドメインを指定しない場合、そのサービス名に DB_DOMAIN データベース属性が追加されます。カンマ区切りリストで、1 つのサービスまたは複数のサービスを指定できます。 ノート: -service パラメータの値には、4KB の制限があります。したがって、インスタンスに割り当てられたすべてのサービスの名前の長さは、合計で 4KB を超えないようにする必要があります。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮定的に評価できます。 ノート: このパラメータは、ポリシー管理サービスでのみ使用できます。
-edition edition_name	サービスの初期セッション・エディション。 サービスにエディションを指定すると、そのサービスを指定するそれ以降のすべて接続で、初期セッション・エディションとしてこのエディションが使用されます。ただし、セッション接続で異なるエディション

パラメータ	説明
	を指定した場合は、そのセッション接続で指定したエディションが初期セッション・エディションとして使用されます。 SRVCTL は、指定されたエディション名を検証しません。接続中、接続ユーザーは指定されたエディションの USE 権限を持っている必要があります。そのエディションが存在しないか、接続ユーザーが指定されたエディションの USE 権限を持たない場合は、エラーが発生します。
-sql_translation_profile profile_name	Oracle Database 以外から Oracle Database にアプリケーションを移行した後でサービスを追加する場合は、このパラメータを使用して、そのサービスに対する SQL 翻訳プロファイルを指定します。 このパラメータは、DBMS_SERVICE サービス属性の SQL トランザクション・プロファイル・パラメータに対応します。 ノート: ● SQL 翻訳機能を使用するには、事前にすべてのサーバー側アプリケーション・オブジェクトおよびデータを Oracle Database に移行しておく必要があります。 ● SQL トランザクション・プロファイルを表示するには、srvctl config service コマンドを使用します。
-preferred preferred_list	管理者管理データベースの場合にサービスを実行する優先インスタンスのリスト。 優先インスタンスのリストは、使用可能なインスタンスのリストと相互に排他である必要があります。 ノート: このパラメータは、Oracle RAC でのみ、管理者管理データベースに対してのみ使用できます。
-available available_list	管理者管理データベースの場合にサービスをフェイルオーバーする使用可能なインスタンスのリスト。 使用可能なインスタンスのリストは、優先インスタンスのリストと相互に排他である必要があります。 ノート: このパラメータは、Oracle RAC でのみ、管理者管理

パラメータ	説明
	データベースに対してのみ使用できます。
-failback {YES | NO}	優先インスタンスのリストを使い切った後に、サービスが使用可能なインスタンスにフェイルオーバーした場合、このパラメータが YES に設定されていると、優先インスタンスが使用可能になったときに、サービスがそのインスタンスに自動的にフェイルバックします。
-serverpool server_pool	ポリシー管理データベースの場合に使用されるサーバー・プールの名前。 ノート: このパラメータは、Oracle RAC でのみ、ポリシー管理データベースに対してのみ使用できます。
-cardinality {UNIFORM | SINGLETON}	サービスのカーディナリティ。UNIFORM(サーバー・プールのすべてのインスタンスに提供される)または SINGLETON(インスタンス 1 つずつで実行する)。 ノート: ● このパラメータは、Oracle RAC でのみ、ポリシー管理データベースに対してのみ使用できます。 ● ポリシー管理 Oracle RAC One Node データベースでは、すべてのサービスが SINGLETON である必要があります。
-netnum network_number	このパラメータを使用して、このサービスが提供されるネットワークを特定します。サービスは、指定したネットワークからの VIP に依存するように構成されます。 ノート: ● このパラメータを省略すると、データベース構成からデフォルトが取得されます。このデフォルトは、srvctl add database または srvctl modify database を使用して指定しますが、このときに、そのデータベースのサービスのデフォルト・ネットワークを指定する-defaultnetwork パラメータを指定します。 ● このパラメータは、Oracle RAC および Oracle RAC One Node データベース構成でのみ使用できます。

パラメータ	説明
-tafpolicy {BASIC | NONE}	TAF ポリシーの指定(管理者管理データベースのみ)。
-role "[PRIMARY][,PHYSICAL_STANDBY] [,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]"	サービス・ロール。1 つ以上のロールをカンマ区切りリストで指定できます。 このオプションは、指定したサービス・ロールの 1 つに Oracle Data Guard データベース・ロールが一致した場合に、データベースのオープン時に、サービスが自動的に起動されるように指定する場合に使用します。 手動で開始する SRVCTL の使用は、サービス・ロールに影響を受けません。 ノート: -role パラメータは、データベースの起動時にのみ、Oracle Data Guard Broker によって使用されます。手動でのサービスの起動ではすべて、ユーザーが起動するサービスの名前を指定する必要があります。
-policy {AUTOMATIC | MANUAL}	サービス管理ポリシー。 AUTOMATIC(デフォルト)の場合、サービスは、データベースの再起動時(計画された再起動(SRVCTL を使用)または障害発生後の再起動)に自動的に起動されます。ただし、サービス・ロールは自動再起動の対象にもなります(-role パラメータ)。 MANUAL の場合、データベースの計画された再起動(SRVCTL を使用)時にサービスが自動的に再起動されることはありません。MANUAL に設定しても、Oracle Clusterware は、実行中のサービスを監視し、障害が発生すると再起動されます。 ノート: CRSCTL を使用して Oracle Clusterware を停止および起動すると、サービスは障害発生時と同様に再起動されます。
-notification {TRUE | FALSE}	OCI 接続に対して高速アプリケーション通知(FAN)を有効化します。
-rfpool pool_name	リーダー・ファーム・サーバー・プールの名前を指定します。
-dtp {TRUE | FALSE}	このサービスの分散トランザクション処理を有効化するかどうかを示します。このサービスは、ポリシー管理データベースでは singleton サービス、管理者管理データベースの単一ノードでは優先サービス

パラメータ	説明
	になります。 ノート: このパラメータは、Oracle RAC でのみ使用できます。
-clbgoal {SHORT | LONG}	接続ロード・バランシングの目標。ランタイム・ロード・バランシングを使用する場合、または統合された接続プールを使用する場合は、このパラメータの値として SHORT を使用します。サービスのノードごとにセッション数でバランスを取る長時間の接続(バッチ・ジョブなど)の場合は、このパラメータの値として LONG を使用します。
-rlbgoal {NONE | SERVICE_TIME | THROUGHPUT}	ランタイム・ロード・バランシングの目標(ロード・バランシング・アドバイザ)。応答時間に基づいて接続のバランスを取るには、このパラメータを SERVICE_TIME に設定します。スループットに基づいて接続のバランスを取るには、このパラメータを THROUGHPUT に設定します。
-failovertype {NONE | SESSION | SELECT | TRANSACTION | AUTO}	フェイルオーバー・タイプを設定します。 Java のアプリケーション・コンティニュイティを有効化するには、このパラメータを TRANSACTION に設定します。透過的アプリケーション・コンティニュイティを有効化するには、このパラメータを AUTO に設定します。 TAF for OCI を有効化するには、このパラメータを SELECT または SESSION に設定します。 ノート: -failovertype を TRANSACTION に設定した場合は、-commit_outcome を TRUE に設定する必要があります。
-failovermethod {NONE | BASIC}	TAF フェイルオーバー・メソッド(下位互換性維持のためのみ)。 ファイルオーバー・タイプ(-failovertype)を NONE 以外の値に設定した場合は、このパラメータで BASIC を選択する必要があります。 ノート: このパラメータは、Oracle RAC でのみ使用できます。
-failoverretry failover_retries	アプリケーション・コンティニュイティおよび TAF の場合は、事象が発生した後に接続を試行する回数が、このパラメータによって決定されます。

パラメータ	説明
-drain_timeout timeout	リソースの排出が完了するまでの許容時間を秒数で指定します。有効な値は、空の文字列("")、0、または正の整数です。デフォルト値は空の文字列です(このパラメータが設定されていないことを表します)。0 に設定されている場合は、即座に排出が発生します。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。サービスに設定すると、この値は、コマンドライン値が設定されていない場合に使用されます。
-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}	サービスを停止する際のモードを指定します。サービスに設定すると、この値は、コマンドラインに値が設定されていない場合に使用されます。 IMMEDIATE: サービスの停止前にセッションの排出を許可します。 TRANSACTIONAL: -drain_timeout パラメータで指定した時間までセッションの排出を許可します。サービスは、制限時間に達したとき、および残っているセッションが終了したときに停止されます。 NONE を指定すると、セッションは終了されます。 デフォルトはサービスの設定から取得されます(サービスに指定されている場合)。それ以外の場合のデフォルトは、NONE です。 ノート: -stopoption パラメータは、-force パラメータと同時に使用する必要があります。
-failover_restore {NONE|LEVEL1|AUTO}	アプリケーション・コンティニュイティの場合は、-failover_restore パラメータを設定すると、リプレイ前にセッションの状態がリストアされます。アプリケーション・コンティニュイティを使用する ODP.NET および Java には LEVEL1 を使用して初期状態をリストアします。 透過的アプリケーション・コンティニュイティを有効にしてセッション状態をリストアするには、このパラメータを AUTO に設定します。 TAF またはアプリケーション・コンティニュイティを使用する OCI アプリケーションの場合は、-failover_restore を LEVEL1 に設定して現在の状態をリストアします。現在の状態が初期状態と異なる場合は、TAF コールバックが必要になります。この制限は、OCI に

パラメータ	説明
	のみ適用されます。
-failoverdelay failover_delay	アプリケーション・コンティニュイティおよび TAF の場合は、このパラメータで、フェイルオーバーにおける、各事象の再接続試行間の時間遅延(秒数)を指定します。
-gsmflags gsm_flags	ローカリティおよびリージョン・フェイルオーバーの値を設定します。
-pdb pluggable_database	プラガブル・データベース(PDB)の名前。 ノート: PDB プロパティは、サービスを作成または変更するときに指定できます。PDB プロパティによって、指定された PDB にサービスが関連付けられます。サービスの PDB プロパティを参照するには、ALL_SERVICES データ・ディクショナリ・ビューを問い合せるか、または SRVCTL ユーティリティを使用している場合は srvctl config service コマンドを実行します。 PDB を指定してサービスを作成または変更した場合、PDB が存在するかどうかは SRVCTL によってチェックされません。このコマンドを実行する前に、PDB が存在することを確認する必要があります。
-global {TRUE | FALSE}	グローバル・データ・サービスのサービスかどうかを指定します。 ノート: このパラメータは、グローバル・データ・サービスでのみ使用できます。
-maxlag maximum_lag_time	最大レプリケーション遅延時間(秒数)。負でない整数である必要があります。デフォルト値は ANY です。
-commit_outcome {TRUE | FALSE}	トランザクション・ガードを有効化します。TRUE に設定すると、トランザクションのセッションがリカバリ可能な停止により失敗したときに、トランザクションのコミット結果にアクセスできます。
-retention retention_time	-commit_outcome を TRUE に設定した場合は、データベースにコミット結果を保持する時間(秒数)が、このパラメータによって決定されます。
-replay_init_time replay_initialization_time	アプリケーション・コンティニュイティの場合、このパラメータは、リクエストの 1 番目の操作の元の実行時間と、再接続の成功後にリプレイが開始される状態になる時間との違い(秒単位)を指定しま

パラメータ	説明
	す。アプリケーション・コンティニュイティは指定した時間が経過するまでリプレイされません。このパラメータは、長い時間が経過した後にシステムがリカバリされたときに、トランザクションが意図せず実行されることを回避するためのものです。デフォルトは 5 分(300)です。最大値は 24 時間(86400)です。-failover_type パラメータが TRANSACTION に設定されていない場合、このパラメータは使用できません。
-session_state {STATIC | DYNAMIC | AUTO}	アプリケーション・コンティニュイティの場合、このパラメータは、非トランザクション・セッション状態がリクエスト内のアプリケーションによって変更される方法を示します。セッション状態の例としては、NLS 設定、オプティマイザのプリファレンス、イベントの設定、PL/SQL グローバル変数、一時表があります。透過的アプリケーション・コンティニュイティの場合、session_state は常に AUTO に設定されます。セッション状態は自動的に追跡されます。 このパラメータが考慮されるのは、-failovertype が TRANSACTION (アプリケーション・コンティニュイティの場合)または AUTO (透過的アプリケーション・コンティニュイティ場合)に設定されている場合のみです。 ● failover_type が TRANSACTION に設定されている場合、session_state を値 DYNAMIC にすることをお薦めします。 ● failover_type が AUTO に設定されている場合、session_state はデフォルトで AUTO になります。 ● failover_type が TRANSACTION または AUTO 以外の値に設定されている場合、session_state の値は設定されません。 リクエストの開始後に非トランザクション値が変更された場合、このパラメータを DYNAMIC または AUTO のいずれかに設定します。ほとんどのアプリケーションで DYNAMIC または AUTO モードを使用する必要があります。
-pqservice pq_service	パラレル問合せサービス名を指定します。
-pqpool pq_pool	パラレル問合せサーバー・プールを指定します。
-update {-preferred new_preferred_instance | -available new_available_instance}	サービス構成に、新しい優先インスタンスまたは使用可能インスタンスを追加します。-preferred では、サービスの優先インスタンス

パラメータ	説明
	のリストに追加するインスタンスの名前を指定します。-available では、サービスの使用可能インスタンスのリストに追加するインスタンスの名前を指定します。
-css_critical {yes | no}	このパラメータを YES に設定することにより、サービスに重みを追加できます。クラスタ内のノードに障害が発生した場合、Oracle Clusterware は最も重みの小さいノードを削除して、クリティカルなサービスを使用可能な状態に保ちます。 ノート: このパラメータは、管理者管理ノードでのみ使用できます。ノードがポリシー管理ノードになった場合、その時点でこのパラメータは適用されなくなります。
-verbose	冗長出力を表示します。
-force	ネットワークにリスナーが構成されていない場合にも、追加操作を強制します。

使用上のノート

このコマンドは、Oracle RAC One NodeデータベースおよびStandard Edition高可用性データベースの配置パラメータを受け入れません。

例

gl.example.comサービスをmy_racデータベースに追加し、OCI接続に対して高速アプリケーション通知を有効化し、フェイルオーバー・メソッドをBASIC、接続時ロード・バランシングの目標をLONG、フェイルオーバー・タイプをSELECT、フェイルオーバー再試行回数を180回、フェイルオーバー遅延を5秒に設定するには、この例の構文を使用します。

```
$ srvctl add service -db my_rac -service gl.example.com -notification TRUE -failovermethod BASIC -
failovertype SELECT -failoverretry 180 -failoverdelay 5 -clbgoal LONG
```

指定したサービスをデータベースに追加し、優先インスタンスと使用可能インスタンスを持ち、TAFに対応するように設定するには、この例の構文を使用します。

```
$ srvctl add service -db crm -service sales -preferred crm01,crm02 -available crm03 -tafpolicy BASIC
```

srvctl config service

サービスの構成を表示します。

構文

```
srvctl config service {-db db_unique_name [-service service_name]
   | -serverpool pool_name [-db db_unique_name]} [-verbose]
```

パラメータ

表A-79 srvctl config serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前。
-service service_name	オプションで、サービスの名前を指定できます。 このパラメータを使用しない場合、SRVCTL によって、データベースに構成されているすべてのサービスの構成情報が表示されます。
-serverpool pool_name	または、このパラメータを使用して、サービス構成を表示するサーバー・プールの名前を指定できます。オプションで、サーバー・プールが存在する特定のデータベースも指定できます。
-verbose	冗長出力を表示します。

使用上のノート

srvctl config serviceコマンドは、srvctl add | modify serviceコマンドを使用して、エディションに指定された文字列値を正確に表示します。エディションを大文字で指定した場合、srvctl config serviceは大文字を表示します。二重引用符(″″)で囲まれている場合、コマンドは二重引用符を表示します。それ以外の場合、コマンドは空の文字列を表示します。

例

このコマンドによって、ポリシー管理データベースに関する次のような情報が返されます。

```
$ srvctl config service -db crm -service webapps
Service name: webapps
Service is enabled
Server pool: sales
Cardinality: SINGLETON
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: false
Failover type: NONE
Failover method: NONE
TAF failover retries: 0
TAF failover delay: 0
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: NONE
TAF policy specification: NONE
Service is enabled on nodes:
Service is disabled on nodes:
Edition: ″my Edition″
```

このコマンドによって、管理者管理データベースに関する次のような情報が返されます。

```
$ srvctl config service -db crm -service webapps
Service name: webapps
Service is enabled
Server pool: sales
Cardinality: 1
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: false
Failover type: NONE
Failover method: NONE
TAF failover retries: 0
TAF failover delay: 0
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: NONE
TAF policy specification: NONE
Preferred instances: crm_1
Available instances:
Edition: "my Edition"
```

管理者管理Oracle RAC One Nodeデータベースのサービス構成では、1つのインスタンスが優先として表示されます。

srvctl disable service

サービスを無効化します。

サービス全体を無効化すると、すべてのインスタンスに適用され、各インスタンスが無効化されます。サービス全体がすでに無効化されている場合、サービス全体に対してこのコマンドを実行すると、エラーが戻されます。したがって、各インスタンスに対してサービス・インジケータを操作する場合、使用できないサービス操作もあります。

構文

```
srvctl disable service -db db_unique_name -services "service_name_list"
   [-instance instance_name | -node node_name]
```

パラメータ

表A-80 srvctl disable serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	サービスを無効化するデータベースの一意の名前を指定します。
-services "service_name_list"	無効化する複数のサービス名を二重引用符("")で囲んだカンマ区切りリストか、または単一のサービス名を指定します。
-instance instance_name	必要に応じて、サービスを無効化するインスタンスの名前を指定できます。 ノート:

パラメータ	説明
	● このパラメータは、管理者管理データベースに使用します。 ● このパラメータは、Oracle Clusterware および Oracle RAC でのみ使用できます。
-node node_name	-instance パラメータを使用するかわりに、このパラメータを使用して、サービスを無効化するノードの名前を指定できます。 ノート: ● このパラメータは、ポリシー管理データベースに使用します。 ● このパラメータは、Oracle Clusterware および Oracle RAC でのみ使用できます。

例

次の例は、CRMデータベースの2つのサービスをグローバルに無効化します。

```
$ srvctl disable service -db crm -service "crm,marketing"
```

次の例では、CRM1インスタンスで実行中のCRMデータベースのサービスの1つを無効化します。その結果、データベースでは引き続きサービスを使用可能ですが、インスタンスは1つ減ります。

```
$ srvctl disable service -db crm -service crm -instance crm1
```

srvctl enable service

Oracle Clusterwareのサービスを有効化します。

サービス全体の有効化は、各インスタンスでサービスを有効化することによって、すべてのインスタンスに対してサービスを有効化することになります。サービス全体がすでに有効化されている場合、このコマンドを実行すると、すべてのインスタンスが対象となって有効化されるのではなく、エラーが戻されます。したがって、各インスタンスに対してサービス・インジケータを操作する場合、使用できないサービス操作もあります。

構文

```
srvctl enable service -db db_unique_name -service "service_name_list"
   [-instance instance_name | -node node_name]
```

パラメータ

表A-81 srvctl enable serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	サービスを有効化するデータベースの一意の名前を指定します。
-service "service_name_list"	有効化する単一のサービス名、または複数のサービス名を二重引用符("")で囲んだカンマ区切りリストを指定します。
-instance instance_name	必要に応じて、このパラメータを使用して、サービスを実行するデータベース・インスタンスの名前を指定できます。 ノート: ● このパラメータは、管理者管理データベースに使用します。 ● このパラメータは、Oracle Clusterware および Oracle RAC でのみ使用できます。
-node node_name	-instance パラメータを使用するかわりに、このパラメータを使用して、サービスを有効化するノードの名前を指定できます。 ノート: ● このパラメータは、ポリシー管理データベースに使用します。 ● このパラメータは、Oracle Clusterware および Oracle RAC でのみ使用できます。

例

次の例は、サービスをグローバルに有効化します。

```
$ srvctl enable service -db crm -service crm
```

次の例は、優先インスタンスを使用するサービスを有効化します。

```
$srvctl enable service -db crm -service crm -instance crm1
```

srvctl modify service

サービス構成を変更します。

このコマンドでは、サービスに対して次のようなオンライン変更がサポートされています。

- インスタンス間のサービス・メンバーの移動
- DBMS_SERVICEのサービス属性(フェイルオーバーの遅延、ランタイム・ロード・バランシングの目標など)のオンライン変更

- 新しい優先インスタンスまたは使用可能インスタンスの追加
- サービスの優先インスタンスまたは使用可能インスタンスの削除

警告:

構成変更は必要最小限にすること、およびオンライン・サービス変更の進行中は他のサービス操作を実行しないことをお薦めします。

構文およびパラメータ

実行するタスクに応じて、次のいずれかの形式のsrvctl modify serviceコマンドを指定の構文で使用します。

インスタンス間でサービスを移動するには:

```
srvctl modify service -db db_unique_name -service service_name
      -oldinst old_instance_name -newinst new_instance_name [-force]
```

ノート:

このコマンドの形式は Oracle Clusterware でのみ使用可能です。

表A-82 srvctl modify serviceのパラメータ - サービスの移動

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。
-service service_name	サービス名を指定します。サービス名を指定しないと、すべてのサービスが SRVCTL によって移動されます。
-oldinst old_instance_name	サービスの移動元のインスタンス名を指定します。
-newinst new_instance_name	サービスの移動先のインスタンス名を指定します。
-force	変更操作を強制し、必要に応じて一部のノードでサービスを停止します。

使用可能インスタンスをサービスの優先インスタンスに変更するには:

```
srvctl modify service -db db_unique_name -service service_name
      -available avail_inst_name [-failback {YES|NO}] -toprefer [-force]
```

ノート:

この形式のコマンドは、Oracle Clusterware でのみ使用可能で、Oracle RAC One Node データベースの配置パラメータを受け入れません。このコマンドでは、サービスが移動または切断されることはなく、サービス属性が変

更されるだけです。

表A-83 srvctl modify serviceのパラメータ - 優先インスタンスへの変更

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。
-service service_name	変更するサービスの名前を指定します。
-available available_inst_name	変更する使用可能インスタンスの名前を指定します。
-failback {YES|NO}	優先インスタンスのリストを使い切った後に、サービスが使用可能なインスタンスにフェイルオーバーした場合、このパラメータがYES に設定されていると、優先インスタンスが使用可能になったときに、サービスがそのインスタンスに自動的にフェイルバックします。
-toprefer	このパラメータは、インスタンスのステータスを優先に変更する場合に指定します。
-force	変更操作を強制します。計画操作の場合、FAN と統合された Oracle 接続プールを使用するのが最適です。FAN 計画イベントを使用すると、ユーザーによる介入なしで Oracle プールから要求が排出されるようになります。インスタンスに接続されているセッションは切断されません。

複数インスタンスの使用可能および優先ステータスを変更するには:

```
srvctl modify service -db db_unique_name -service service_name
     -modifyconfig -preferred "preferred_list" [-available "available_list"]
     [-force]
```

ノート:

この形式のコマンドは、Oracle Clusterware でのみ使用可能で、Oracle RAC One Node データベースの配置パラメータを受け入れません。このコマンドでは、サービスが移動または切断されることはなく、サービス属性が変更されるだけです。インスタンスに接続されているセッションは切断されません。

表A-84 srvctl modify serviceのパラメータ - 複数インスタンスのステータスの変更

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。

パラメータ	説明
-service service_name	変更するサービスの名前を指定します。
-modifyconfig	このパラメータは、このサービスに指定されたインスタンスのみを使用するように SRVCTL に指示します(すでにサービスに割り当てられていて指定されていないインスタンスは削除されます)。
-preferred "preferred_instance_list"	二重引用符("")で囲まれた優先インスタンスのカンマ区切りリストを指定します。
-available "available_instance_list"	二重引用符("")で囲まれた使用可能インスタンスのカンマ区切りリストを指定します。
-force	変更操作を強制します。計画操作の場合、FAN と統合された Oracle 接続プールを使用するのが最適です。FAN 計画イベントを使用すると、ユーザーによる介入なしで接続プールから要求が排出されるようになります。

他のサービス属性またはOracle Clusterwareのサービスを変更するには:

```
srvctl modify service -db db_unique_name -service service_name [-eval]
 [-serverpool pool_name] [-cardinality {UNIFORM|SINGLETON}]
 [-drain_timeout timeout] [-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}]
 [-pqservice pqsvc_name] [-pqpool pq_pool_list]
 [-pdb pluggable_database] [-tafpolicy {BASIC|NONE}]
 [-edition edition_name] [-role "[PRIMARY][,PHYSICAL_STANDBY]
 [,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]"] [-notification {TRUE|FALSE}]
 [-dtp {TRUE|FALSE}] [-clbgoal {SHORT|LONG}] [-rlbgoal {NONE|SERVICE_TIME|THROUGHPUT}]
 [-failovertype {NONE|SESSION|SELECT|TRANSACTION|AUTO}] [-failovermethod {NONE|BASIC}]
 [-failover_restore [NONE|LEVEL1|AUTO]] [-failoverretry failover_retries]
 [-failoverdelay failover_delay] [-policy {AUTOMATIC|MANUAL}]
 [-sql_translation_profile profile_name] [-commit_outcome {TRUE|FALSE}]
 [-retention retention_time] [-replay_init_time replay_initiation_time]
 [-session_state {STATIC|DYNAMIC|AUTO}] [-global_override] [-verbose] [-force]
```

表A-85 srvctl modify serviceのパラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。
-service service_name	変更するサービスの名前を指定します。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮

パラメータ	説明
	定的に評価できます。 ノート: このパラメータは、ポリシー管理サービスでのみ使用できます。
-serverpool pool_name	ポリシー管理データベースの場合に使用されるサーバー・プールの名前。 ノート: このパラメータは、Oracle RAC でのみ、ポリシー管理データベースに対してのみ使用できます。
-cardinality {UNIFORM|SINGLETON}	サービスのカーディナリティを指定します。UNIFORM(サーバー・プールのすべてのインスタンスに提供される)または SINGLETON(インスタンス 1 つずつで実行する)。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-drain_timeout timeout	リソースの排出が完了するまでの許容時間を秒数で指定します。有効な値は、空の文字列("")、0、または正の整数です。デフォルト値は空の文字列です(このパラメータが設定されていないことを表します)。0 に設定されている場合は、即座に排出が発生します。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。サービスに設定すると、この値は、コマンドライン値が設定されていない場合に使用されます。
-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}	サービスを停止する際のモードを指定します。このパラメータをサービスに設定すると、この値は、コマンドラインに値が設定されていない場合に使用されます。 IMMEDIATE: サービスの停止前にセッションの排出を許可します。 TRANSACTIONAL: -drain_timeout パラメータで指定した時間までセッションの排出を許可します。サービスは、制限時間に達したとき、および残っているセッションが終了したときに停止されます。 NONE を指定すると、セッションは終了されます。

パラメータ	説明
	デフォルトはサービスの設定から取得されます(サービスに指定されている場合)。それ以外の場合のデフォルトは、NONE です。 ノート: -stopoption パラメータは、-force パラメータと同時に使用する必要があります。
-pqservice pqsvc_name	パラレル問合せサービス名のカンマ区切りリストを指定します。
-pqpool pq_pool_list	パラレル問合せサーバー・プール名のカンマ区切りリストを指定します。
-pdb pluggable_database	プラガブル・データベース(PDB)の名前を指定します。 ノート: PDB プロパティは、サービスを作成または変更するときに指定できます。PDB プロパティによって、指定された PDB にサービスが関連付けられます。サービスの PDB プロパティを参照するには、ALL_SERVICES データ・ディクショナリ・ビューを問い合せるか、または SRVCTL ユーティリティを使用している場合は srvctl config service コマンドを実行します。 PDB を指定してサービスを作成または変更した場合、その PDB が存在するかどうかは SRVCTL によってチェックされません。このコマンドを実行する前に、PDB が存在することを確認する必要があります。
-tafpolicy {BASIC|NONE}	トランザクション・フェイルオーバー(TAF)ポリシーを指定します(管理者管理データベースの場合のみ)。
-edition edition_name	サービスの初期セッション・エディション。 サービスにエディションを指定すると、そのサービスを指定するそれ以降のすべて接続で、初期セッション・エディションとしてこのエディションが使用されます。ただし、セッション接続で異なるエディションを指定した場合は、そのセッション接続で指定したエディションが初期セッション・エディションとして使用されます。 SRVCTL は、指定されたエディション名を検証しません。接続中、接続ユーザーは指定されたエディションの USE 権限を持っている必要があります。そのエディションが存在しないか、接続ユーザーが指定されたエディションの USE 権限を持たない場合は、エラーが発生します。

パラメータ	説明
-role "[PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]"	サービスが自動的に開始する必要があるデータベース・モード。1つ以上のロールをカンマ区切りリストで指定できます。 ノート: -role パラメータは、データベースの起動時にのみ、Oracle Data Guard Broker によって使用されます。手動でのサービスの起動ではすべて、ユーザーが起動するサービスの名前を指定する必要があります。
-notification {TRUE|FALSE}	TRUE の値を指定すると、Oracle Call Interface (OCI)の高速アプリケーション通知(FAN)が有効になります。
-dtp {TRUE|FALSE}	TRUE を指定すると、このサービスの分散トランザクション処理が有効になります。これにより、XA アフィニティ用に、一度に 1 つのみのインスタンスでサービスが提供されるようになります。
-clbgoal {SHORT|LONG}	このパラメータは接続時ロード・バランシングの目標を設定する場合に使用します。ランタイム・ロード・バランシングを使用する場合は SHORT に設定し、バッチ・ジョブや以前の SQL*Forms スタイルなどの長時間の接続の場合は LONG に設定します。
-rlbgoal {NONE|SERVICE_TIME|THROUGHPUT}	このパラメータは、実行時ロード・バランシングの目標を設定する場合に使用します。応答時間に基づいて接続のバランスを取るには、このパラメータを SERVICE_TIME に設定します。スループットに基づいて接続のバランスを取るには、このパラメータを THROUGHPUT に設定します。
-failovertype {NONE|SESSION|SELECT|TRANSACTION|AUTO}	このパラメータは、フェイルオーバー・タイプを設定する場合に使用します。 アプリケーション・コンティニュイティを有効化するには、このパラメータを TRANSACTION に設定します。透過的アプリケーション・コンティニュイティを有効化するには、このパラメータを AUTO に設定します。 TAF を有効化するには、このパラメータを SELECT または SESSION に設定します。
-failovermethod {NONE|BASIC}	TAF フェイルオーバー・メソッドを指定します(下位互換性維持のためのみ)。
-failover_restore [NONE|LEVEL1|AUTO]	アプリケーション・コンティニュイティの場合は、-failover_restore

パラメータ	説明
	パラメータを設定すると、リプレイ前にセッションの状態がリストアされます。アプリケーション・コンティニュイティを使用する ODP.NET および Java には LEVEL1 を使用して初期状態をリストアします。 透過的アプリケーション・コンティニュイティを有効にしてセッション状態をリストアするには、このパラメータを AUTO に設定します。 TAF またはアプリケーション・コンティニュイティを使用する OCI アプリケーションの場合は、-failover_restore を LEVEL1 に設定して現在の状態をリストアします。現在の状態が初期状態と異なる場合は、TAF コールバックが必要になります。この制限は、OCI にのみ適用されます。
-failoverretry failover_retries	アプリケーション・コンティニュイティおよび TAF の場合は、インシデント後の接続試行回数を指定します。
-failoverdelay failover_delay	アプリケーション・コンティニュイティおよび TAF の場合は、フェイルオーバー時の各インシデントの再接続試行間の遅延(秒数)を指定します。
-policy {AUTOMATIC|MANUAL}	サービス管理ポリシーを指定します。
-sql_translation_profile profile_name	Oracle Database 以外から Oracle Database にアプリケーションを移行した後でサービスを変更する場合は、このパラメータを使用して、そのサービスに対する SQL 翻訳プロファイルを指定します。 SQL 翻訳プロファイルを NULL 値に設定する場合は、-p フラグの後に空の文字列を入力する必要があります。 ノート: SQL 翻訳機能を使用するには、事前にすべてのサーバー側アプリケーション・オブジェクトおよびデータを Oracle Database に移行しておく必要があります。
-commit_outcome {TRUE|FALSE}	トランザクション・ガードを有効化します。TRUE に設定すると、トランザクションのセッションがリカバリ可能な停止により失敗したときに、トランザクションのコミット結果にアクセスできます。
-retention retention_time	トランザクション・ガード(-commit_outcome パラメータを TRUE に設定)の場合は、このパラメータで、コミット結果をデータベースに

パラメータ	説明
	保持する時間(秒数)を決定します。
-replay_init_time replay_initiation_time	アプリケーション・コンティニュイティの場合は、このパラメータで、元のリクエスト開始されたときからの時間(秒数)を指定します。アプリケーション・コンティニュイティは指定した時間が経過するまでリプレイされません。この属性は、長い時間が経過した後でシステムがリカバリしたときにリクエストの意図しないリプレイを回避します。デフォルト値は 300 (5 分)です。
-session_state {STATIC|DYNAMIC|AUTO}	アプリケーション・コンティニュイティの場合、このパラメータは、非トランザクション・セッション状態がリクエスト内のアプリケーションによって変更される方法を示します。セッション状態の例としては、NLS 設定、オプティマイザのプリファレンス、イベントの設定、PL/SQL グローバル変数、一時表があります。透過的アプリケーション・コンティニュイティの場合、session_state は常に AUTO に設定されます。セッション状態は自動的に追跡されます。 このパラメータが考慮されるのは、-failovertype が TRANSACTION (アプリケーション・コンティニュイティの場合)または AUTO (透過的アプリケーション・コンティニュイティ場合)に設定されている場合のみです。 ● failover_type が TRANSACTION に設定されている場合、session_state を値 DYNAMIC にすることをお薦めします。 ● failover_type が AUTO に設定されている場合、session_state はデフォルトで AUTO になります。 ● failover_type が TRANSACTION または AUTO 以外の値に設定されている場合、session_state の値は設定されません。 ほとんどのアプリケーションには値 AUTO または DYNAMIC をお薦めします。使用する値が不明な場合またはアプリケーションをカスタマイズできる場合は、DYNAMIC を使用します。
-global_override	グローバル・サービス属性を変更するオーバーライド値。 このパラメータは、-role、-policy、-notification、-failovertype、-failovermethod、-failoverdelay、-failoverretry および-edition の各パラメータとともに使用します。

パラメータ	説明
-verbose	冗長出力を表示します。
-force	変更操作を強制し、必要に応じて一部のノードでサービスを停止します。

使用上のノート

- サービス属性(フェイルオーバーの遅延、ランタイム・ロード・バランシングの目標など)をオンラインで変更した場合、変更が有効になるのは、サービスが次回(再)起動されたときです。
- 新しい優先インスタンスまたは使用可能インスタンスが追加されるようにサービス構成を変更した場合でも、既存サービスの稼働状態に影響はありません。ただし、新しく追加されたインスタンスは、srvctl start serviceコマンドが発行されるまで、自動的にはサービスを提供しません。
- サービスに対して使用可能なインスタンスがあり、優先インスタンスまたは使用可能インスタンスが削除されるようにサービス構成を変更した場合、サービスの稼働状態に予測できない変化が発生することがあります。
    - 新しいサービス構成に従って、一部のインスタンスでサービスが停止、削除されます。
    - サービスは、サービス構成から削除されるインスタンスで稼働している場合があります。
    - そのようなサービスは、新しいサービス構成内の次に使用可能なインスタンスに再配置されます。

前述の状況のため、オンライン・サービスを変更した場合、インスタンスが削除されていなくても、ユーザーは一時的にサービスを利用できないことがあります。または、サービスから削除されるインスタンスで、サービスを一時的に利用できないことがあります。

例

次の例は、あるインスタンスから別のインスタンスにサービス・メンバーを移動します。

```
$ srvctl modify service -db crm -service crm -oldinst crm1 -newinst crm2
```

次の例は、使用可能インスタンスを優先インスタンスに変更します。

```
$ srvctl modify service -db crm -service crm -available crm1 -toprefer
```

次のコマンドでは、優先インスタンスおよび使用可能インスタンスが交換されます。

```
$ srvctl modify service -db crm -service crm -modifyconfig -preferred "crm1" ¥
-available "crm2"
```

関連項目

- Oracle Data Guard Broker
- Oracle Database SQL翻訳および移行ガイド

srvctl predict service

サービス障害の結果を予測します。

構文

```
srvctl predict service -db db_unique_name -service service_name [-verbose]
```

パラメータ

表A-86 srvctl predict serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	確認対象のサービスが動作するデータベースの一意の名前を指定します。
-service service_name	確認する単一のサービス名、または複数のサービス名を二重引用符(“”)で囲んだカンマ区切りリストを指定します。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

srvctl relocate service

指定した1つのインスタンスから指定した別のインスタンスに指定したサービス名を一時的に再配置します。

このコマンドは、同時に1つのソース・インスタンスと1つのターゲット・インスタンスでのみ機能し、1つのソース・インスタンスから1つのターゲット・インスタンスに1つのサービスまたはすべてのサービスを再配置します。

構文

```
srvctl relocate service -db db_unique_name [-service service_name
  | -pdb pluggable_database] {-oldinst old_inst_name
  [-newinst new_inst_name] | -currentnode source_node [-targetnode target_node]} [-drain_timeout
timeout]
  [-wait YES | NO] [-pq] [-force [-noreplay]] [-eval] [-verbose]
```

パラメータ

表A-87 srvctl relocate serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	再配置対象のサービスが動作するデータベースの一意の名前を指定します。
-service service_name	再配置するサービスの名前を指定します。サービスを指定しないと、再配置可能なすべてのサービスが再配置されます。再配置できないものは、所定の場所に残されます。
-pdb pluggable_database	このパラメータは、特定のプラガブル・データベースで実行しているサービスを再配置する場合に使用します。

パラメータ	説明
-oldinst old_inst_name	サービスの再配置元のインスタンス名を指定します。
-newinst new_inst_name	サービスの再配置先のインスタンス名を指定します。このパラメータは省略可能です。インスタンスを指定しないと、Oracle Clusterware によって新しいインスタンスが選択されます。 ノート: 管理者管理データベースを使用している場合は、-oldinst および-newinst パラメータを使用する必要があり、ターゲット・インスタンスはサービスの優先リストまたは使用可能リストにある必要があります。
-currentnode source_node	サービスが現在実行中のノードの名前
-targetnode target_node	サービスが再配置されるノードの名前。ターゲット・ノードを指定しないと、Oracle Clusterware によって新しい場所が選択されます。 ノート: ポリシー管理データベースを使用している場合は、-currentnode および-targetnode パラメータを使用する必要があります。
-drain_timeout timeout	リソースの排出が完了するまでの許容時間を秒数で指定します。有効な値は、空の文字列("")、0、または正の整数です。デフォルト値は空の文字列です(このパラメータが設定されていないことを表します)。0 に設定されている場合は、即座に排出が発生します。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。サービスに設定すると、この値は、コマンドライン値が設定されていない場合に使用されます。
-wait YES | NO	YES を選択すると、サービスの再配置元ノードでサービスの排出が完了するまで待機します。
-stopoption option	サービスを停止する際のモードを指定します。サービスに設定すると、この値は、コマンドラインに値が設定されていない場合に使用されます。 IMMEDIATE: サービスの停止前にセッションの排出を許可しま

パラメータ	説明
	す。 TRANSACTIONAL: -drain_timeout パラメータで指定した時間までセッションの排出を許可します。サービスは、制限時間に達したとき、および残っているセッションが終了したときに停止されます。 NONE を指定すると、セッションは終了されます。 デフォルトはサービスの設定から取得されます(サービスに指定されている場合)。それ以外の場合のデフォルトは、NONE です。 ノート: -stopoption パラメータは、-force パラメータと同時に使用する必要があります。
-pq	パラレル問合せサービスでアクションを実行
-force [-noreplay]	サービス操作の停止中または再配置中にすべてのセッションを切断します。 サービスの停止操作または再配置操作中にセッションが終了された後で、進行中のトランザクションをアプリケーション・コンティニュイティがリプレイしないようにする場合は、必要に応じて、-noreplay パラメータを指定できます。 -noreplay パラメータは-force とともに使用するように制限されていませんが、サービスがすべてのセッションを切断するように強制した後で、進行中のトランザクションをリプレイしないようにする場合は、-force に-noreplay が必要になります。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮定的に評価できます。
-verbose	詳細な出力。

例

データベース・インスタンスcrm1で実行されるcrmサービスの特定のサービス・メンバーをデータベース・インスタンスcrm3に一時的に再配置するには:

```
$ srvctl relocate service -db crm -service crm -oldinst crm1 -newinst crm3
```

関連項目

- SQL*Plusユーザーズ・ガイドおよびリファレンス

srvctl remove service

サービスの構成を削除します。

構文

```
srvctl remove service -db db_unique_name -service service_name
   [-instance instance_name] [-global_override]
```

パラメータ

表A-88 srvctl remove serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	削除するサービスが動作するデータベースの一意の名前を指定します。
-service service_name	削除するサービスの名前を指定します。
-instance instance_name	必要に応じて、管理者管理データベースのインスタンス名を指定できます。 ノート: このパラメータは、Oracle Clusterware でのみ使用できます。
-global_override	必要に応じて、このパラメータを使用して、グローバル・サービスに対して機能する値を上書きできます。このパラメータは、グローバルでないサービスに対しては無視されます。

例

次の例では、crmというクラスタ・データベースのすべてのインスタンスからsalesサービスを削除します。

```
$ srvctl remove service -db crm -service sales
```

次の例は、crmクラスタ・データベースの特定のインスタンスからsalesサービスを削除します。

```
$ srvctl remove service -db crm -service sales -instance crm02
```

srvctl start service

データベース、プラガブル・データベース、またはインスタンスで1つまたは複数のサービスを起動します。

構文

```
srvctl start service [-db db_unique_name] [-service "services_list"
  [-pq] | -pdb pluggable_database | -serverpool pool_name]
```

```
  [-node node_name | -instance instance_name]
  [-global_override] [-startoption start_options] [-eval] [-verbose]
```

パラメータ

パラメータ	説明
-db db_unique_name	データベースの一意の名前を指定します。
-service "service_list"	1 つのサービス名を指定するか、二重引用符("")で囲まれたサービス名のカンマ区切りリストを指定します。 このパラメータを指定しない場合は、SRVCTL によって指定したデータベースのすべてのサービスが起動されます。 ノート: すべての手動でのサービスの起動では、起動するサービスの名前をユーザーが指定する必要があります。
-pq	このパラメータは、パラレル問合せサービスの開始アクションを制限する場合に指定します。
-pdb pluggable_database	プラガブル・データベースの名前を指定します。サービスの開始をプラガブル・データベースの特定のオブジェクトに制限する場合は、必要に応じて、ノードの名前またはインスタンスの名前を指定できます。
-serverpool pool_name	-pq パラメータを使用するかわりに、開始するサービスが含まれているサーバー・プールの名前を指定することもできます。このパラメータは、ポリシー管理データベースに使用します。
-node node_name	開始するサービスが存在しているノードの名前を指定します。このパラメータは、ポリシー管理データベースに使用します。
-instance instance_name	開始するサービスが存在しているインスタンスの名前を指定します。このパラメータは、管理者管理データベースに使用します。
-global_override	グローバル・サービスに対して機能するオーバーライド値。このパラメータは、グローバル・サービスに対してのみ使用します。非グローバル・サービスに対して指定しても、このパラメータは無視されます。
-startoption start_options	サービスの起動時にデータベース・インスタンスの起動が必要

パラメータ	説明
	な場合に使用する起動オプションを指定します。オプションには、OPEN、MOUNT および NOMOUNT があります。 ノート: 起動オプションに複数の語を指定する場合(read only、read write など)、語をスペースで区切り、二重引用符(″″)で囲みます。たとえば″read only″とします。
-verbose	冗長出力を表示します。

使用上のノート

- すでに実行中のサービスを開始しようとすると、srvctl start serviceコマンドは失敗します。
- すでに最大数(優先インスタンスの数)のインスタンスでサービスが稼働している場合に、サービスをそのインスタンスで起動しようとすると、srvctl start serviceコマンドは失敗します。
- srvctl modify serviceコマンドとsrvctl relocate serviceコマンドを使用すると、インスタンスでのサービスの移動またはサービスのステータス変更を実行できます。

例

次の例では、特定のデータベース上のサービスをすべて開始します。

```
$ srvctl start database -db myDB
```

次の例では、どのプラガブル・データベースにサービスが存在しているかにかかわらず、サービスのリストを開始します(後者の例では、オプションでパラレル問合せサービスに制限しています)。

```
$ srvctl start database -db myDB -service "myServ01,myServ02"
$ srvctl start database -db myDB -service "myServ01,myServ02" -pq
```

次の例では、特定のサーバー・プール内のサービスをすべて開始します。

```
$ srvctl start database -db myDB -serverpool myServerPool
```

次の例では、特定のプラガブル・データベースのサービスをすべて開始します。後者の2つの例では、オプションで、それぞれ単一のノードまたは単一のインスタンスに制限しています。

```
$ srvctl start service -db myDB -pdb myPDB1
$ srvctl start service -db myDB -pdb myPDB1 -node myRACNode01
$ srvctl start service -db myDB -pdb myPDB1 -instance myDB01
```

次の例では、特定のインスタンス(すべてのプラガブル・データベース)で特定のデータベースのサービスをすべて開始します。

```
$ srvctl start service -db myDB -instance myDB01
```

次の例では、特定のノード(すべてのプラガブル・データベース)で特定のデータベースのサービスをすべて開始します。

```
$ srvctl start service -db myDB -node myRACNode01
```

次の例では、特定のノードまたは特定のインスタンスでサービスのリストを開始します。

```
$ srvctl start service -db myDB -service "myService01,myService02" -node myRACNode01
$ srvctl start service -db myDB -service "myService01,myService02" -instance myDB01
```

srvctl status service

サービスのステータスを表示します。

Oracle RAC One Nodeデータベースに関しては、オンライン・データベース再配置が進行中の場合、再配置がアクティブか失敗かに関係なく、このコマンドはソース・ノードと宛先ノードおよび再配置のステータスを表示します。

構文

```
srvctl status service -db db_unique_name [-service "service_name_list"]
   [-force] [-verbose]
```

パラメータ

必要に応じて、このパラメータを使用して、無効化されたアプリケーションを含めることができます。

表A-89 srvctl status serviceコマンドのパラメータ

パラメータ	説明
-db db_unique_name	ステータスの確認対象のサービスが動作するデータベースの一意の名前を指定します。
-service "service_name_list"	必要に応じて、ステータスを確認するサービス名のカンマ区切りリストを指定できます。 このパラメータを使用しない場合は、指定したデータベースのすべてのサービスのステータスが表示されます。
-force	オプションで、このパラメータを使用すると無効化されたアプリケーションを含めることができます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

srvctl stop service

クラスタ・データベース全体でグローバルに、または指定したインスタンスで、1つ以上のサービスを停止します。

構文

クラスタ内の特定のノードのサービスを停止するには:

```
srvctl stop service -node node_name [-stopoption IMMEDIATE|TRANSACTIONAL|NONE]
```

```
   [-drain_timeout timeout] [-wait {YES | NO}] [-force] [-noreplay]
   [-global_override] [-verbose]
```

データベースのサービスを停止するには:

```
srvctl stop service -db db_unique_name [-pq] [-rf] [-pdb pluggable_database |
    -service "service_list" [-eval]] [-node node_name | -instance instance_name |
    -serverpool pool_name] [-stopoption IMMEDIATE|TRANSACTIONAL|NONE]
   [-drain_timeout timeout] [-wait {YES | NO}] [-force [-noreplay]
   [-global_override] [-verbose]
```

パラメータ

ノート:

Oracle Grid Infrastructure 21c以降、ポリシー管理データベースは非推奨です。

表A-90 srvctl stop serviceコマンドのパラメータ

パラメータ	説明
-node node_name	オプションで、サービスを停止するノードの名前を指定できます。このパラメータは、特定のノード上のすべてのサービスを停止する場合、-db パラメータを指定せずに使用します。-db パラメータを使用すると、そのデータベースの指定したノードのサービスのみが停止されます。
-db db_unique_name	データベースの一意の名前を指定します。
-pdb pluggable_database	また、このパラメータを使用して、特定のプラガブル・データベースで実行中のサービスを停止します。
-service "service_list"	停止する特定の 1 つのサービス、または二重引用符("")で囲まれたサービス名のリストを指定します。 サービス名のリストを指定しないと、SRVCTL はデータベース、または特定のインスタンスのすべてのサービスを停止します。
-pq	このパラメータは、パラレル問合せサービスの停止アクションを制限する場合に指定します。
-instance instance_name	オプションで、サービスを停止するインスタンスの名前を指定できます。
-serverpool pool_name	オプションで、停止するサービスが含まれているサーバー・プールの

パラメータ	説明
	名前を指定できます。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮定的に評価できます。
-stopoption IMMEDIATE|TRANSACTIONAL|NONE	サービスを停止する方法を指定します。この属性がすでにサービスに設定されている場合、コマンドに-stopoption パラメータを含めないと、その値がデフォルト値として使用されます。それ以外の場合のデフォルトは、NONE です。 ● IMMEDIATE: サービスの停止前にセッションの排出を許可します。排出されないセッションは、-drain_timeout で指定した時間制限に達すると終了します。 ● TRANSACTIONAL を指定した場合、セッションはコミット後すぐに終了します。-drain_timeout で指定した時間制限に達して、残りのセッションが終了すると、サービスは停止します。 ● NONE を指定すると、セッションは終了されます。 ノート: -stopoption パラメータは、-force パラメータと同時に使用する必要があります。
-drain_timeout timeout	リソースの排出が完了するまでの許容時間を秒数で指定します。有効な値は、空の文字列("")、0、または正の整数です。デフォルト値は空の文字列です(このパラメータが設定されていないことを表します)。0 に設定されている場合は、即座に排出が発生します。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。サービスに設定すると、この値は、コマンドライン値が設定されていない場合に使用されます。
-wait {YES | NO}	YES を選択すると、サービスを停止するノードでサービスの排出が完了するまで待機します。
-force [-noreplay]	SRVCTL によってサービスを強制停止します。これにより、指定された停止オプション(IMMEDIATE または TRANSACTIONAL)を使用してすべてのセッションが切断されます。サービスを使用中のセッションは再接続の後、別のインスタンスに接続する必要がありま

パラメータ	説明
	す。 ノート: ● -force パラメータを指定しないと、このサービスにすでに接続しているセッションは接続状態を保ちますが、新しいセッションをサービスに確立することはできません。 ● セッションが終了された後で、進行中のトランザクションをアプリケーション・コンティニュイティがリプレイしないようにする場合は、オプションで-noreplay パラメータを指定できます。 -noreplay パラメータは、-force とともに使用するように制限されていません。ただし、サービスを強制的に停止した後で、進行中のトランザクションをリプレイしない場合は、-force に-noreplay が必要です。
-global_override	グローバル・サービスに対して機能するオーバーライド値。 SRVCTL は、サービスがグローバル・サービスでない場合、このパラメータを無視します。
-verbose	このパラメータは、詳細出力を表示する場合に使用します。

例

次のコマンド例では、IMMEDIATE方式を使用してcrm1インスタンスのcrmデータベースのcrmeast PDBで実行されているサービスが停止され、別のノードにサービスが転送されるのに60秒割り当てます。

```
$ srvctl stop service -db crm -pdb crmeast -instance crm1 -drain_timeout 60 -force
- stopoption immediate -verbose
```

次のコマンド例では、各サービスに指定されているデフォルトの停止オプションを使用し、すべてのセッションがそのノードから排出されるまで待機することで、Oracle Clusterwareによって管理されるノードnode1で実行されているすべてのサービスを停止します。

```
$ srvctl stop service -node node1 -wait yes
```

srvctl add srvpool

Oracle Databaseをホストするように構成されているサーバー・プールをクラスタに追加します。

構文

```
srvctl add srvpool -serverpool server_pool_name [-eval]
```

```
    [-importance importance] [-min min_size] [-max max_size]
    [-servers "node_list" | -category server_category] [-force] [-verbose]
```

パラメータ

表A-91 srvctl add srvpoolコマンドのパラメータ

パラメータ	説明
-serverpool server_pool_name	サーバー・プールの名前。
-eval	このパラメータを使用すると、コマンドがシステムに及ぼす影響を仮定的に評価できます。
-importance importance	サーバー・プールの重要度(デフォルト値は 0 です)。
-min min_size	サーバー・プールの最小サイズ(デフォルト値は 0 です)。
-max max_size	サーバー・プールの最大サイズ。デフォルト値は-1 であり、これは、サイズが無制限であることを意味します。
-servers "node_list"	二重引用符("")で囲まれた候補ノード名のカンマ区切りリスト。サーバー・プールには候補リストのノードのみが含まれますが、候補リストのすべてのノードがサーバー・プールに含まれるとはかぎりません。 ノート: Oracle Database 12c では、-category パラメータの値に従って、サーバーがサーバー・プールに割り当てられます。
-category server_category	サーバー・プールに対して使用するサーバー・カテゴリ(""は空のカテゴリ値を表します)。
-force	他のサーバー・プールのリソースを停止する必要がある場合でも、サーバー・プールを追加します。
-verbose	冗長出力を表示します。

使用上のノート

- SRVCTLによって、サーバー・プールの名前の先頭に「ora. 」が追加されます。
- このコマンドはOracle Clusterwareでのみ使用可能です。

例

次のコマンドでは、SP1という名前のサーバー・プールを追加します(サーバー・プールの重要度を1、サーバー・プール内の最小ノード数を3、サーバー・プール内の最大ノード数を7に設定しています)。

```
srvctl add srvpool -serverpool SP1 -importance 1 -min 3 -max 7
```

srvctl config srvpool

クラスタ内の特定のサーバー・プールの構成情報を表示します。この構成情報には、名前、最小サイズ、最大サイズ、重要度、およびサーバー名のリスト(該当する場合)が含まれます。

構文

```
srvctl config srvpool [-serverpool pool_name]
```

パラメータ

このコマンドで使用できるパラメータは、構成情報を表示するサーバー・プールの名前を指定する-serverpool pool_nameのみです。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

次に、このコマンドの例を示します。

```
$ srvctl config srvpool -serverpool dbpool
```

srvctl modify srvpool

クラスタのサーバー・プールを変更します。

最小サイズ、最大サイズおよび重要度の数値を増やした場合に、サイズ変更により他のサーバー・プールの最小サイズおよび重要度が相対的に低くなった場合は、このサーバー・プールの新しいサイズが確保されるように、このサーバー・プールへのサーバーの再割当てがCRSデーモンにより試行されます。

構文

```
srvctl modify srvpool -serverpool pool_name [-eval] [-importance importance]
   [-min min_size] [-max max_size] [-servers "server_list"]
   [-category "server_category"] [-verbose] [-force]
```

パラメータ

表A-92 srvctl modify srvpoolコマンドのパラメータ

パラメータ	説明
-serverpool pool_name	変更するサーバー・プールの名前を指定します。
-eval	オプションで、このパラメータを使用するとコマンドがシステムに及ぼす影響を仮定的に評価できます。

パラメータ	説明
-importance importance	必要に応じて、サーバー・プールの重要度を変更できます。
-min min_size	必要に応じて、サーバー・プールの最小サイズを変更できます。デフォルト値は 0 (ゼロ)です。
-max max_size	必要に応じて、サーバー・プールの最大サイズを変更できます。-1 の値はサーバー・プールの最大サイズを UNLIMITED に設定します。
-servers "server_list"	必要に応じて、候補サーバー名を二重引用符("")で囲んだカンマ区切りリストを指定できます。 ノート: Oracle Database 12c では、-category パラメータの値に従って、サーバーがサーバー・プールに割り当てられます。
-category "server_category"	必要に応じて、二重引用符("")で囲んだサーバー・カテゴリ、または""で表される空のカテゴリ値を変更できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。
-force	必要に応じて、このパラメータを使用して、一部のリソースが停止している場合でも操作を強制できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例は、サーバー・プールsrvpool1の重要度ランクを0、最小サイズを2、最大サイズを4に変更します。

```
$ srvctl modify srvpool -serverpool srvpool1 -importance 0 -min 2 -max 4
```

srvctl remove srvpool

特定のサーバー・プールを削除します。

このサーバー・プールに依存するデータベースまたはサービスが存在する場合は、サーバー・プールの削除操作が正常に実行されるように、まずそれらのリソースがサーバー・プールから削除されます。

構文

```
srvctl remove srvpool -serverpool pool_name [-eval] [-verbose]
```

パラメータ

表A-93 srvctl remove srvpoolコマンドのパラメータ

パラメータ	説明
-serverpool pool_name	削除するサーバー・プールの名前を指定します。
-eval	必要に応じて、このパラメータを使用して、システムに変更を加えることなく、サーバー・プールを削除した場合の影響を評価できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 指定したサーバー・プールを正常に削除できた場合は、サーバー・プールの最小サイズ、最大サイズおよび重要度に応じて、CRSデーモンによりサーバーが他のサーバー・プールに割り当てられます。CRSデーモンにより、これらのサーバーが空きサーバー・プールに戻される場合もあります。

例

次の例では、システムからサーバー・プールを削除します。

```
$ srvctl remove srvpool -serverpool srvpool1
```

srvctl status srvpool

サーバー・プール名、サーバー・プール内のサーバーの数、そして必要に応じてサーバー・プール内のサーバーの名前を表示します。

構文

```
srvctl status srvpool [-serverpool pool_name] [-detail]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 必要に応じて、ステータスを確認するサーバー・プールの名前を指定できます。このパラメータを使用した場合は、そのサーバー・プールの名前とその中のサーバーの数(および必要に応じてサーバー名)が出力されます。
- -detailパラメータを使用しただけで、-serverpoolパラメータで特定のサーバー・プールを指定しなかった場合は、各サーバー・プールに現在割り当てられているサーバーの名前が出力されます。

srvctl add vip

仮想IPアドレス(VIP)をノードに追加します。

構文

```
srvctl add vip -node node_name -address {VIP_name|ip}/netmask[/if1[|if2|...]]
      -netnum network_number [-skip] [-verbose]
```

パラメータ

表A-94 srvctl add vipコマンドのパラメータ

パラメータ	説明
-node node_name	VIP を追加するノードの名前。
-address {VIP_name|ip}/netmask [/if1[|if2|...]]	指定したノードに従来の VIP ノード・アプリケーションを作成します。 1 つの VIP_name またはアドレスを、IPv4 ネットマスクまたは IPv6 接頭辞長とともに指定できます。
-netnum network_number	VIP を取得するネットワーク番号。デフォルトのネットワーク番号は 1 です。
-skip	VIP アドレスの到達可能性の確認をスキップするには、このパラメータを指定します。
-verbose	詳細出力

ノート:

使用上のノート

- 同一ノードの同一ネット番号(サブネットまたはインタフェース・ペア)に複数のVIPを持つことはできません。
- このコマンドはOracle Clusterwareでのみ使用可能です。

例

次に、このコマンドの例を示します。

```
# srvctl add network -netnum 2 -subnet 192.168.16.0/255.255.255.0
# srvctl add vip -node node7 -address 192.168.16.17/255.255.255.0 -netnum 2
```

1番目のコマンドはネットワーク番号2を作成し、2番目のコマンドはこのネットワークにVIPを追加します。ネットワーク番号は、他のSRVCTLコマンドの-netnumパラメータの後に指定できます。

srvctl config vip

ユーザーVIP以外で、クラスタのすべてのネットワークにおけるすべてのVIPを表示します。

構文

```
srvctl config vip {-node node_name | -vip vip_name}
```

パラメータ

表A-95 srvctl config vipコマンドのパラメータ

パラメータ	説明
-node node_name	ノード名を指定します。
-vip vip_name	あるいは、VIP 名を指定することもできます。

使用上のノート

このコマンドはOracle Clusterwareでのみ使用可能です。

例

このコマンドによって、次のような出力が返されます。

```
$ srvctl config vip -node crmnode1
VIP exists: ipv4, ipv6, network number 1, hosting node adc2100252
```

srvctl disable vip

特定のVIPを無効化します。

構文

```
srvctl disable vip -vip vip_name [-verbose]
```

使用上のノート

- このコマンドはOracle Clusterwareでのみ使用可能です。
- 無効化するVIPの名前を指定します。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

例

次のコマンドは、VIPを無効化します。

```
$ srvctl disable vip -vip vip1 -verbose
```

srvctl enable vip

特定のVIPを有効化します。

構文

```
srvctl enable vip -vip vip_name [-verbose]
```

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 有効化するVIPの名前を指定します。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

例

次の例では、crm1-vipというVIPを有効化します。

```
$ srvctl enable vip -vip crm1-vip -verbose
```

srvctl getenv vip

特定のVIPの環境変数の値を取得します。

構文

```
srvctl getenv vip -vip vip_name [-envs "name_list"] [-verbose]
```

パラメータ

表A-96 srvctl getenv vipコマンドのパラメータ

パラメータ	説明
-vip vip_name	環境変数の値を取得する VIP の名前を指定します。
-envs "name_list"	必要に応じて、特定の環境変数の名前のカンマ区切りリストを指定できます。このパラメータを使用しない場合は、VIP に関連付けられているすべての環境変数の値が表示されます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例は、指定されたVIPのすべての環境変数をリストします。

```
$ srvctl getenv vip -vip node1-vip
```

srvctl modify vip

IPアドレス・タイプを変更しますが、これを使用してIPアドレスのみも変更できます。

構文

```
srvctl modify vip -node node_name -address {VIP_name|ip}/netmask[/if1[|if2|...]]
   [-netnum network_number] [-verbose]
```

パラメータ

表A-97 srvctl modify vipコマンドのパラメータ

パラメータ	説明
-node node_name	VIP を変更するノードの名前を指定します。
-address {VIP_name|ip}/netmask[/if1[|if2|...]]	既存の VIP の構成を変更するには、このパラメータを使用します。VIP に IPv4 アドレスがあり、指定するアドレスが IPv6 で、IP アドレス・タイプが both に設定され、ネットワーク・タイプが static に設定されている場合、SRVCTL によって、IPv6 アドレスがそのリソースの既存の IPv4 アドレスに追加されます。 1 つの VIP_name または IP アドレスを、IPv4 ネットマスクまたは IPv6 接頭辞長とともに指定できます。
-netnum network_number	必要に応じて、VIP を取得するネットワーク番号を指定できます。このパラメータを使用しない場合は、nodeapps VIP の取得元と同じデフォルト・ネットワークから VIP が取得されます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 同一ノードの同一ネット番号(サブネットまたはインタフェース・ペア)に複数のVIPを持つことはできません。

例

次の例は、IPv4アドレスをVIPに追加します(まだ存在しない場合)。VIPにIPv4アドレスが存在する場合は、新しいネットワーク指定に置き換えられます。

```
# srvctl modify vip -node node7 -address 192.168.16.17/255.255.255.0 -netnum 2
```

srvctl predict vip

VIP障害の結果を予測します。

構文

```
srvctl predict vip [-vip vip_name] [-verbose]
```

使用上のノート

- 必要に応じて、障害の結果を評価するVIPの名前を指定できます。
- オプションで、-verboseパラメータを使用すると詳細な出力を表示できます。

srvctl relocate vip

現行のホスティング・ノードからクラスタ内の別のノードに特定のVIPを再配置します。

構文

```
srvctl relocate vip -vip vip_name [-node node_name] [-force] [-verbose]
```

パラメータ

表A-98 srvctl relocate vipコマンドのパラメータ

パラメータ	説明
-vip vip_name	再配置する VIP の名前を指定します。
-node node_name	必要に応じて、VIP の再配置先のターゲット・ノードの名前を指定できます。
-force	必要に応じて、このパラメータを使用して、依存性に関係なく VIP の再配置を強制できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

例

次の例では、クラスタ内の別のノードにVIPを再配置します。

```
$ srvctl relocate vip -vip vip1 -node node3
```

srvctl remove vip

特定のVIPを削除します。

構文

```
srvctl remove vip -vip "vip_name_list" [-force] [-noprompt] [-verbose]
```

パラメータ

表A-99 srvctl remove vipコマンドのパラメータ

パラメータ	説明
-vip "vip_name_list"	削除するVIP名を二重引用符("")で囲んだカンマ区切りリストを指定します。
-force	必要に応じて、このパラメータを使用して、依存性に関係なくVIPを削除できます。
-noprompt	必要に応じて、このパラメータを使用してプロンプトを抑止できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、システムから複数のVIPを削除します。

```
$ srvctl remove vip -vip "vip1,vip2,vip3" -force -noprompt -verbose
```

srvctl setenv vip

クラスタVIP環境構成を管理します。

構文

```
srvctl setenv vip -vip vip_name {-envs "name=val[,...]" | -env "name=val"}
   [-verbose]
```

パラメータ

表A-100 srvctl setenv vipコマンドのパラメータ

パラメータ	説明
-vip vip_name	環境変数を設定するVIPの名前を指定します。
-envs "name=val[,...]"	設定する環境変数の名前/値ペアのカンマ区切りリストを、

パラメータ	説明
	二重引用符(“”)で囲んで指定します。
-env ″name=val″	環境変数のリストのかわりに、このパラメータを使用して、カンマやその他の特殊文字を含んだ値を二重引用符(“”)で囲んで単一の環境変数に設定できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例は、クラスタVIPの言語環境構成を設定します。

```
$ srvctl setenv vip -vip crm1-vip -env ″LANG=en″
```

srvctl start vip

特定のVIPまたは特定のノード上のVIPを起動します。

構文

```
srvctl start vip {-node node_name | -vip vip_name} [-verbose]
```

パラメータ

表A-101 srvctl start vipコマンドのパラメータ

パラメータ	説明
-node node_name	起動する VIP が存在するノードの名前を指定します。
-vip vip_name	ノードを指定するかわりに、起動する VIP を指定できます。
-verbose	必要に応じて、このパラメータを使用して、詳細な出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、特定のVIPを起動します。

```
$ srvctl start vip -vip crm1-vip -verbose
```

srvctl status vip

特定のVIPまたは特定のノード上のVIPのステータスを表示します。

構文

```
srvctl status vip {-node node_name | -vip vip_name} [-verbose]
```

パラメータ

表A-102 srvctl status vipコマンドのパラメータ

パラメータ	説明
-node node_name	ステータスを確認する VIP が存在するノードの名前を指定します。
-vip vip_name	ノードを指定するかわりに、ステータスを確認する VIP を指定できます。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

srvctl stop vip

特定のノードの特定のVIPまたはすべてのVIP(フェイルオーバーによって再配置されたVIPを含む)を停止します。

構文

```
srvctl stop vip {-node node_name | -vip vip_name} [-force] [-relocate] [-verbose]
```

パラメータ

表A-103 srvctl stop vipコマンドのパラメータ

パラメータ	説明
-node node_name	停止する VIP が存在するノードの名前を指定します。このパラメータを使用すると、フェイルオーバーされた VIP を含め、特定のノード上のすべての VIP が停止されます。
-vip vip_name	ノードを指定するかわりに、停止する VIP を指定できます。

パラメータ	説明
-force	必要に応じて、このパラメータを使用して、依存性に関係なくVIP を停止できます。
-relocate	必要に応じて、このパラメータを使用して、VIP を再配置できます。 ノート: -node node_name パラメータは、-relocate パラメータと一緒に使用する必要があります。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

使用上のノート

このコマンドは、Oracle Clusterwareでのみ使用できます。

例

次の例では、フェイルオーバーされたVIPを含め、mynode1上のすべてのVIPを停止します。

```
$ srvctl stop vip -node mynode1 -verbose
```

srvctl unsetenv vip

指定したクラスタVIPの環境構成の設定を解除します。

構文

```
srvctl unsetenv vip -vip "vip_name_list" -envs "name_list" [-verbose]
```

パラメータ

表A-104 srvctl unsetenv vipコマンドのパラメータ

パラメータ	説明
-vip "vip_name_list"	二重引用符(“”)で囲まれた VIP 名のカンマ区切りリストを指定します。
-envs "name_list"	設定を解除する環境変数名のカンマ区切りリストを、二重引用符(“”)で囲んで指定します。
-verbose	オプションで、このパラメータを使用すると詳細出力を表示できます。

例

次の例は、 クラスタVIPのCLASSPATH環境変数の設定を解除します。

```
$ srvctl unsetenv vip -vip "crm2-vip" -envs "CLASSPATH"
```

srvctl config volume

特定のボリュームまたはすべてのボリュームの構成を表示します。

構文

```
srvctl config volume [-volume volume_name] [-diskgroup disk_group_name]
   [-device volume_device]
```

パラメータ

表A-105 srvctl config volumeコマンドのパラメータ

パラメータ	説明
-volume volume_name	構成を表示するボリュームの名前を指定します。
-diskgroup disk_group_name	構成を表示するボリュームが存在するディスク・グループの名前を指定します。
-device volume_device	構成を表示するボリューム・デバイスへのパスを指定します。

使用上のノート

- どのオプション・パラメータも指定しない場合、SRVCTLによってすべてのボリュームの構成情報が表示されます。
- -volumeパラメータのみを指定した場合は、ディスク・グループに関係なく、その名前を持つすべてのボリュームの構成が表示されます。
- -diskgroupパラメータのみを指定した場合は、指定したディスク・グループに存在するボリュームの構成情報が表示されます。
- -deviceパラメータのみを指定した場合は、そのデバイス指定子に一致するボリュームの構成情報が表示されます。
- -diskgroupパラメータと-deviceパラメータを指定した場合は、指定したディスク・グループに存在するボリューム・デバイスの構成情報が表示されます。
- このコマンドはOracle Clusterwareでのみ使用可能です。

例

このコマンドによって、次のような情報が返されます。

```
$ srvctl config volume -device /dev/asm/volume1-123
Diskgroup Name: DG1
Volume Name   : VOL1
Volume Device : /dev/asm/volume1-123
Volume is enabled.
Volume is enabled on nodes:
```

```
Volume is disabled on nodes:
```

どのパラメータも指定しない場合、SRVCTLによって次のようなすべてのボリュームの構成情報が返されます。

```
$ srvctl config volume
Diskgroup name: DG1
Volume name: VOL1
Volume device: /dev/asm/volume1-123
Volume is enabled.
Volume is enabled on nodes:
Volume is disabled on nodes:
Diskgroup name: DG1
Volume name: VOL2
Volume device: /dev/asm/volume2-456
Volume is enabled.
Volume is enabled on nodes:
Volume is disabled on nodes:
```

srvctl disable volume

特定のボリュームまたはすべてのボリュームのOracle Clusterware管理を無効化します。

このコマンドは、ボリュームのOracle Clusterwareリソースで動作することによって、そのボリューム・デバイスを停止できます。このコマンドはボリューム・デバイスを停止するのではありません。

構文

```
srvctl disable volume {-volume volume_name -diskgroup disk_group_name |
    -device volume_device}
```

パラメータ

表A-106 srvctl disable volumeコマンドのパラメータ

パラメータ	説明
-volume volume_name	無効化するボリュームの名前を指定します。
-diskgroup disk_group_name	無効化するボリュームが存在するディスク・グループの名前を指定します。
-device volume_device	-diskgroup パラメータを使用するかわりに、無効化するボリューム・デバイスへのパスを指定できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 無効化する特定のボリュームを指定する必要があります。特定のディスク・グループまたは特定のボリューム・デバイスに存在するボリュームを指定できます。

例

次の例は、DATAというディスク・グループに存在するVOLUME1というボリュームを無効化します。

```
$ srvctl disable volume -volume VOLUME1 -diskgroup DATA
```

srvctl enable volume

特定のボリュームまたはすべてのボリュームのOracle Clusterware管理を有効化します。

このコマンドは、ボリュームのOracle Clusterwareリソースで動作することによって、そのボリューム・デバイスを起動できます。このコマンドは、ボリューム・デバイスを起動せず、SQLコマンドALTER DISKGROUP ENABLE VOLUMEまたはASMCMDコマンドvolenableとは異なります(これらの2つのコマンドはボリューム・デバイスをオンラインにし、実行時状態では、ボリューム・デバイスをアクセス可能にするためです)。

構文

```
srvctl enable volume {-volume volume_name -diskgroup disk_group_name |
   -device volume_device}
```

パラメータ

表A-107 srvctl enable volumeコマンドのパラメータ

パラメータ	説明
-volume volume_name	有効化するボリュームの名前を指定します。
-diskgroup disk_group_name	有効化するボリュームが存在するディスク・グループの名前を指定します。
-device volume_device	-diskgroup パラメータを使用するかわりに、有効化するボリューム・デバイスへのパスを指定できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- 有効化する特定のボリュームを指定する必要があります。特定のディスク・グループまたは特定のボリューム・デバイスに存在するボリュームを指定できます。

例

次の例は、DATAというディスク・グループに存在するVOLUME1というボリュームを有効化します。

```
$ srvctl enable volume -volume VOLUME1 -diskgroup DATA
```

srvctl remove volume

特定のボリュームを削除します。

構文

このコマンドは、次のいずれかの構文モデルで使用します。

```
srvctl remove volume -volume volume_name -diskgroup disk_group_name [-force]
srvctl remove volume -device volume_device [-force]
```

パラメータ

表A-108 srvctl remove volumeコマンドのパラメータ

パラメータ	説明
-volume volume_name	削除するボリュームの名前を指定します。
-diskgroup disk_group_name	削除するボリュームが存在するディスク・グループの名前を指定します。
-device volume_device	削除するボリュームが存在するファイル・システム・リソースへのパスを指定します。
-force	このパラメータを使用して、実行中のボリュームも削除できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- ボリュームは、Oracle ASMで作成されます。
- 削除する特定のボリュームを指定する必要があります。特定のディスク・グループまたは特定のボリューム・デバイスに存在するボリュームを指定できます。

例

次の例は、DATAというディスク・グループに存在するVOLUME1というボリュームを削除します。

```
$ srvctl remove volume -volume VOLUME1 -diskgroup DATA
```

関連項目

- Oracle Automatic Storage Management管理者ガイド

srvctl start volume

特定の有効ボリュームを起動します。

構文

```
srvctl start volume {-volume volume_name -diskgroup disk_group_name |
    -device volume_device} [-node node_list]
```

パラメータ

表A-109 srvctl start volumeコマンドのパラメータ

パラメータ	説明
-volume volume_name	起動するボリュームの名前を指定します。
-diskgroup disk_group_name	起動するボリュームが存在するディスク・グループの名前を指定します。
-device volume_device	-diskgroup パラメータを使用するかわりに、起動するボリューム・デバイスへのパスを指定できます。
-node node_list	必要に応じて、起動するボリュームが存在するノード名を二重引用符(″″)で囲んだカンマ区切りリストを指定できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- srvctl start volumeコマンドでは、ボリューム・サービスは作成されません。ボリュームがすでに存在し、ボリューム・リソースが有効化されている場合は、SRVCTLによってそのボリュームの起動が試行されます。ボリュームが存在していても、リソースが無効化されている場合は、srvctl start volumeによりエラーが戻されます。

例

次の例は、DATAというディスク・グループに存在するVOLUME1というボリュームを起動します。

```
$ srvctl start volume -volume VOLUME1 -diskgroup DATA
```

srvctl status volume

特定のボリュームまたはすべてのボリュームのステータスを表示します。

構文

```
srvctl status volume [-device volume_device] [-volume volume_name]
   [-diskgroup disk_group_name] [-node "node_list"]
```

パラメータ

表A-110 srvctl status volumeコマンドのパラメータ

パラメータ	説明
-device volume_device	必要に応じて、ステータスを表示するボリューム・デバイスへのパスを指定できます。
-volume volume_name	必要に応じて、ステータスを表示するボリュームの名前を指定

パラメータ	説明
	できます。
-diskgroup disk_group_name	必要に応じて、ステータスを表示するボリュームが存在するディスク・グループの名前を指定できます。
-node "node_list"	必要に応じて、ステータスを表示するボリュームが存在するノード名を二重引用符(""")で囲んだカンマ区切りリストを指定できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- どのオプション・パラメータも指定しない場合、SRVCTLによってすべてのボリュームのステータスが表示されます。
- -volumeパラメータのみを指定した場合、指定したボリュームのステータスが表示されます。
- -diskgroupパラメータのみを指定した場合、指定したディスク・グループに存在するボリュームのステータスが表示されます。
- -deviceパラメータのみを指定した場合、指定したボリューム・デバイスのステータスが表示されます。
- -diskgroupパラメータと-deviceパラメータを指定した場合、指定したディスク・グループ内のボリューム・デバイスのステータスが表示されます。
- -nodeパラメータを指定した場合、指定したノードに存在するボリュームのステータスが表示されます。

例

このコマンドを実行すると、次のような情報が表示されます。

```
$ srvctl status volume -volume vol1
Volume vol1 of diskgroup diskgrp1 for device volume_device_path1 is enabled
Volume vol1 of diskgroup diskgrp1 for device volume_device_path1 is running
```

前述の例では、-nodeパラメータを指定していないため、すべてのノードに対してステータス問合せが行われます。

```
$ srvctl status volume
Volume vol1 of diskgroup diskgrp for device volume_device_path1 is enabled
Volume vol1 of diskgroup diskgrp for device volume_device_path1 is running
Volume vol2 of diskgroup diskgrp for device volume_device_path2 is enabled
Volume vol2 of diskgroup diskgrp for device volume_device_path2 is running
```

前述の例では、パラメータを指定していないため、SRVCTLによって登録されているすべてのボリュームのステータスが表示されます。

srvctl stop volume

実行中の特定のボリュームを停止します。

構文

```
srvctl stop volume {-volume volume_name -diskgroup disk_group_name |
    -device volume_device} [-node "node_list"]
```

パラメータ

表A-111 srvctl stop volumeコマンドのパラメータ

パラメータ	説明
-volume volume_name	停止するボリュームの名前を指定します。
-diskgroup disk_group_name	停止するボリュームが存在するディスク・グループの名前を指定します。
-device volume_device	-diskgroup パラメータを使用するかわりに、停止するボリューム・デバイスへのパスを指定できます。
-node "node_list"	必要に応じて、停止するボリュームが存在するノード名を二重引用符(""")で囲んだカンマ区切りリストを指定できます。

使用上のノート

- このコマンドは、Oracle Clusterwareでのみ使用できます。
- srvctl stop volumeコマンドでは、ボリュームの停止(無効化)は試行されますが、リソースが無効化されたり、Oracle ASMからボリュームが削除されることはありません。

例

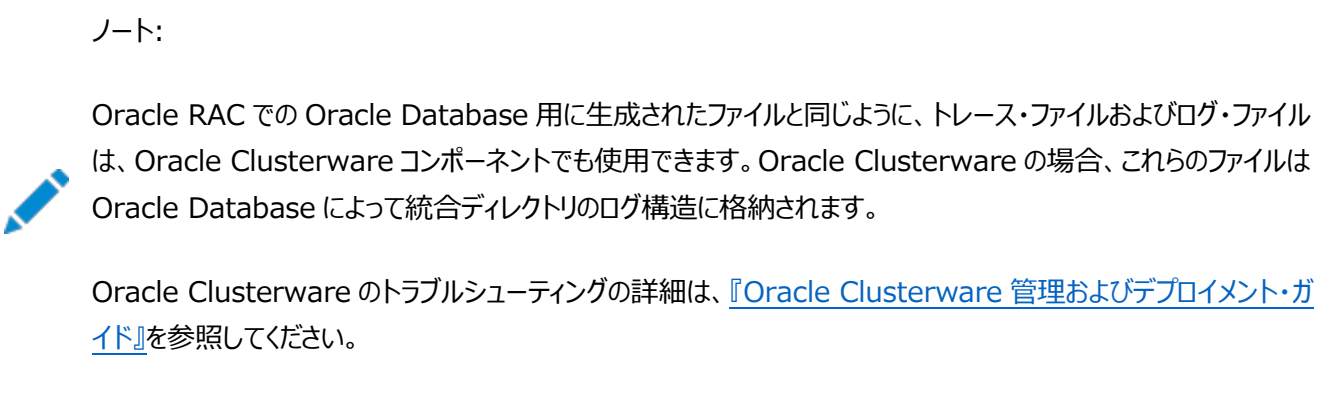
次の例は、DATAというディスク・グループに存在するVOLUME1というボリュームを停止します。

```
$ srvctl stop volume -volume VOLUME1 -diskgroup DATA
```

B Oracle RACのトラブルシューティング

この付録では、トレース・ファイルおよびログ・ファイルを使用して、Oracle Real Application Clusters(Oracle RAC)コンポーネントの問題を診断する方法について説明します。この項には次のトピックが含まれます:

- エラー分析に必要なファイルの場所
- Oracle RACでの診断データの管理
- Oracle RACでのインスタンス固有のアラート・ファイルの使用
- Oracle RACでのJavaベースのツールとユーティリティに関するトレースの有効化
- 停止保留問題の解決
- Oracle RACインスタンスでプライベート・ネットワークが使用されているかどうかの判別方法



ノート:

Oracle RAC での Oracle Database 用に生成されたファイルと同じように、トレース・ファイルおよびログ・ファイルは、Oracle Clusterware コンポーネントでも使用できます。Oracle Clusterware の場合、これらのファイルは Oracle Database によって統合ディレクトリのログ構造に格納されます。

Oracle Clusterware のトラブルシューティングの詳細は、『Oracle Clusterware 管理およびデプロイメント・ガイド』を参照してください。

エラー分析に必要なファイルの場所

Oracle Databaseは、Oracle RAC環境で発生する重要なイベントに関する情報をトレース・ファイルに記録します。

Oracle RACのトレース・ファイルは、非クラスタのOracle Databaseの場合と同じです。すべてのインスタンスに対してトレース・ファイルを定期的に監視およびバックアップし、将来のトラブルシューティングのための情報を保持します。

ORA-600エラーに関する情報は、各インスタンスのalert_SID.logファイルにあります。ここでSIDには、インスタンスの識別子を指定します。

バックグラウンド・プロセスおよびサーバー・プロセス用のアラート・ログ・ファイルとすべてのトレース・ファイルは、自動診断リポジトリに書き込まれます(自動診断リポジトリの位置は、DIAGNOSTIC_DEST初期化パラメータで指定できます)。たとえば:

```
$ORACLE_BASE/diag/rdbms/$DBNAME/$SID_NAME/trace
```

Oracle Databaseは、各バックグラウンド・スレッドに対して別々のトレース・ファイルを作成します。Oracle RACのバックグラウンド・スレッドでは、トレース・ファイルを使用して、データベース操作およびデータベース・エラーが記録されます。これらのトレース・ログは、トラブルシューティングに有効であり、また、Oracleサポート・サービスは、これらのトレース・ログを使用して、クラスタ・データベース構成の問題をより効率的にデバッグできます。トレース・ファイル名はオペレーティング・システム固有ですが、通常、各ファイルにはそれを書き込むLGWRやRECOなどのプロセスの名前が含まれます。Linux、UNIXおよびWindowsシステムでは、バックグラウンド・プロセスのトレース・ファイル名は、SID_process_name_process_identifier.trcになります。

DIAGNOSTIC_DEST初期化パラメータを設定した場合は、トレース・ファイルはユーザー・プロセスに対しても作成されます。ユーザー・プロセス・トレース・ファイルの名前は、SID_ora_process_identifier/thread_identifier.trcの形式です(LinuxおよびUNIXシステムの場合、process_identifierはプロセス識別子(PID)を表す5桁の数値で、Windowsシステムの場合、thread_identifierはスレッド識別子です)。

関連項目

- Oracle Clusterwareのトラブルシューティング
- データベースの監視

Oracle RACでの診断データの管理

Oracle RACインスタンス全体にわたる問題は、最も診断が難しいタイプの問題である可能性があります。

たとえば、複数のインスタンスからトレース・ファイルを相互に関連付け、マージする必要がある場合があります。Oracle Database 12c リリース2 (12.2)には、診断データの収集および管理用の高度な障害診断インフラストラクチャが含まれており、データベース診断データの格納に自動診断リポジトリ(ADR)ファイル・ベースのリポジトリが使用されます。共有ディスクにADRベースを作成すると、同じOracle RACデータベースのすべてのインスタンスのADRホームを同じADRベースに配置できます。共有記憶域を使用すると、次の処理を実行できます。

- ADRCIコマンドライン・ツールを使用して、すべてのインスタンスの診断を相互に関連付けることができます。

  ADRCIは、ADR内の診断データを確認し、Oracleサポートへの送信用に発生事象と問題の情報をzipファイルに圧縮できるコマンドライン・ツールです。診断データには、発生事象と問題の説明、トレース・ファイル、ダンプ、状態モニター・レポート、アラート・ログ・エントリなどが含まれます。

- データ・リカバリ・アドバイザを使用して、破損したデータ・ブロック、破損または欠落したファイルおよびその他のデータ障害を修復できます。

  データ・リカバリ・アドバイザは、永続データ障害の自動的診断、修復オプションの提示およびユーザーの要求に応じた問題の修復を行う、Oracle Databaseインフラストラクチャです。

関連項目

- ADRCI: ADRコマンド・インタプリタ
- 問題の診断と解決

Oracle RACでのインスタンス固有のアラート・ファイルの使用

Oracle RACデータベースの各インスタンスには、1つのアラート・ファイルがあります。

各インスタンスのアラート・ファイル(alert_SID.log)には、データベース操作中に発生したエラー・メッセージおよび例外に関する重要な情報が含まれています。インスタンスを起動するたびに、アラート・ファイルの末尾に情報が追加されます。すべての処理スレッドをインスタンスのアラート・ファイルに書き込むことができます。

alert_SID.logファイルは、DIAGNOSTIC_DEST初期化パラメータで指定されたディレクトリにあります。

Oracle RACでのJavaベースのツールとユーティリティに関するトレースの有効化

Oracle RACで使用可能なJavaベースのすべてのツールとユーティリティは、ツールまたはユーティリティと同じ名前のスクリプトを処理してコールします。

クラスタ検証ユーティリティ(CVU)、Oracle Database Configuration Assistant (Oracle DBCA)、Net Configuration Assistant (NETCA)、サーバー制御ユーティリティ(SRVCTL)などがあります。たとえば、Oracle DBCAを実行するには、コマンドdbcaを入力します。

Oracle Databaseでは、デフォルトでOracle DBCAおよびDatabase Upgrade Assistant(DBUA)のトレースが有効になっています。CVUおよびSRVCTLの場合は、SRVM_TRACE環境変数をTRUEに設定すると、Oracle Databaseでトレースを生成できます。トレースはログ・ファイルに書き込まれます。たとえば、Oracle DBCAとOracle DBUAの場合、トレースはそれぞれOracle_base/cfgtoollogs/dbcaとOracle_base/cfgtoollogs/dbua内のログ・ファイルに書き込まれます。

停止保留問題の解決

SHUTDOWN IMMEDIATEが保留され、停止リクエストを繰り返してもOracle Databaseが迅速に応答しなくなる場合があります。

これは、Oracle Clusterwareで現行の停止リクエストが処理中のため発生する場合があります。そのような場合は、以後の停止リクエストに対し、SQL*Plusを使用してSHUTDOWN ABORTを発行します。

Oracle RACインスタンスでプライベート・ネットワークが使用されているかどうかの判別方法

このトピックでは、Oracle RACインスタンスでプライベート・ネットワークが使用されているかどうかを手動で判別する方法について説明します。

ただし、このタスクを実行する場合は、Oracle Enterprise Manager Cloud Controlのグラフィカル・ユーザー・インタフェース(GUI)を使用してインターコネクトを確認することをお薦めします。

ほとんどのネットワーク・プロトコルで、oradebug ipcコマンドを発行し、データベースで使用されているインターコネクトを確認できます。たとえば:

```
oradebug setmypid
oradebug ipc
```

これらのコマンドは、DIAGNOSTIC_DEST初期化パラメータによって指定された場所にトレース・ファイルをダンプします。出力は次のようになる場合があります。

```
SSKGXPT 0x1a2932c flags SSKGXPT_READPENDING     info for network 0
        socket no 10    IP 172.16.193.1         UDP 43749
        sflags SSKGXPT_WRITESSKGXPT_UP  info for network 1
        socket no 0     IP 0.0.0.0      UDP 0...
```

この例では、データベースがユーザー・データグラム・プロトコル(UDP)プロトコルでIP 172.16.193.1を使用していることを確認

できます。oradebug tracefile_nameコマンドを発行して、出力が書き込まれるトレースの場所を出力することもできます。

また、V$CLUSTER_INTERCONNECTSビューを問い合せて、プライベート・インターコネクトに関する情報を確認できます。たとえば:

```
SQL> SELECT * FROM V$CLUSTER_INTERCONNECTS;
NAME  IP_ADDRESS          IS_  SOURCE
----- ------------------------ ---  --------------------------------
eth0  138.2.236.114         NO  Oracle Cluster Repository
```

用語集

自動ワークロード・リポジトリ(AWR)

すべてのOracle Databaseに存在する組込みリポジトリ。Oracle Databaseは、すべての重要な統計およびワークロード情報のスナップショットを定期的に生成し、AWRに格納します。

管理者管理データベース

実行可能なサーバーやデータベース内のどこでサービスが実行可能かを特に定義したデータベースです。

キャッシュ一貫性

任意のキャッシュを介してメモリー位置を読み取っても、別のキャッシュを介してその位置に書き込まれた最新のデータが戻されるように、複数のキャッシュ内のデータを同期化することです。キャッシュ整合性とも呼ばれます。

キャッシュ・フュージョン

ブロックを保持しているインスタンスのメモリー・キャッシュから要求側インスタンスのメモリー・キャッシュにブロックを直接コピーする、Oracle RACのディスク不要のキャッシュ一貫性メカニズム。

カーディナリティ

通常操作中に実行するデータベース・インスタンスの数です。

CDB

マルチテナント・コンテナ・データベース(CDB)とは、0 (ゼロ)、1つまたは数多くのユーザー作成のプラガブル・データベース(PDB)を含むOracle Databaseです。すべてのOracle Databaseは、CDBか非CDBのいずれかです。

クライアント・クラスタ

サーバー・クラスタで名前を通知するクラスタ。

クラスタ

エンド・ユーザーおよびアプリケーションからは1つのサーバーとして認識される、相互に接続された複数のコンピュータまたはサーバー。

クラスタ構成ポリシー

システムで定義されている各サーバー・プールに対して、定義が1つのみ含まれるドキュメント。

クラスタ構成ポリシー・セット

クラスタ内に構成されるすべてのサーバー・プール名が定義され、1つ以上の構成ポリシーが含まれるドキュメント。

クラスタ・データベース

Oracle RACデータベースの総称。

クラスタ・ファイル・システム

高パフォーマンスのサービスをクライアントに提供するために連携するサーバーのクラスタによる分散ファイル・システム。クラスタ・ファイル・システム・ソフトウェアはストレージ・クラスタ・コンポーネントへの分散リクエストを処理します。

クラスタ・レディ・サービス・デーモン(CRSD)

高可用性リカバリおよびOCRのメンテナンスのような管理操作を実行する主なOracle Clusterwareのプロセスです。また、アプリケーション・リソースを管理し、rootユーザー(Mac OS Xベースのシステムではadminグループのユーザー)として実行され、障害発生時に自動的に再起動します。

Cluster Synchronization Services(CSS)

クラスタ全体のメンバーシップの共通ビューを提供して、各ノードのメンバーシップの状態を検出および追跡するOracle Clusterwareコンポーネント。CSSはプロセスの状態(特にデータベース・インスタンスの状態)も監視します。バックグラウンド・プロセスであるグローバル・エンキュー・サービス・モニター(LMON)が、クラスタ・データベース環境の状態を監視し、CSSへの登録および登録解除を行います。「OCSSD」も参照してください。

クラスタ時刻同期化サービス

クラスタ内のすべてのノードの内部クロックがすべて同期化されるようにする時刻同期メカニズム。

クラスタ検証ユーティリティ(CVU)

共有ストレージ・デバイス、ネットワーク構成、システム要件、Oracle Clusterware、グループ、ユーザーなどのOracle RACの様々なコンポーネントを検証するツール。

コミット結果

トランザクションがコミットされた後に、Oracle Databaseからクライアントに送られるメッセージ。これらのメッセージには永続性がありません。

データベース・プール

データベース・クラウド内の一連のデータベースであり、一意のグローバル・サービス・セットを提供し、特定の管理ドメインに属しま

す。クラウド・データベースを複数のプールにパーティション化すると、サービス管理が簡素化され、かつ、各プールを異なる管理者が管理できることによって、セキュリティが向上します。

分散トランザクション処理(DTP)

分散トランザクションの枠組み。外部的に調整されたトランザクションであるXAタイプと、内部的に調整されたトランザクションである分散SQLタイプ(Oracleのデータベース・リンク)の両方が含まれます。

動的ネットワーク

IPv4にDHCPを、またはIPv6にステートレス自動構成(autoconfig)を使用するネットワーク。

イベント・マネージャ(EVM)

Oracle Clusterwareイベントをパブリッシュするバックグラウンド・プロセス。イベントの発生時に、EVMは指定されたコールアウト・ディレクトリをスキャンし、そのディレクトリ内のすべてのスクリプトを実行します。

イベント・マネージャ・デーモン(EVMD)

コールアウトを管理するracgevtプロセスを開始する、LinuxまたはUNIXのイベント・マネージャ・デーモン。

障害グループ

障害グループはディスク・グループのディスクのサブセットです。これらのディスクはハードウェアを共有するため、同時に障害が発生する可能性があります。障害グループは、データのミラー・コピーを格納するために使用されます。

高速アプリケーション通知(FAN)

アプリケーションでは、FANを使用して、迅速な障害の検出、障害発生後の接続プールの分散の均等化、および障害が発生したコンポーネントの修復時の接続プールの分散の再均等化を行うことができます。FAN通知プロセスでは、クラスタ・サーバーが使用不可になるか、またはネットワーク・インタフェースに障害が発生した場合にOracle Databaseが発行するシステム・イベントが使用されます。

高速接続フェイルオーバー

JDBC、OCIまたはODP.NETを使用するクライアントのようなFAN統合クライアントに、高可用性を提供します。高速接続フェイルオーバーを使用するようにクライアントを構成すると、クライアントは自動的にFANイベントをサブスクライブし、データベースのUPイベントおよびDOWNイベントに対処できます。それに対応して、Oracle Databaseは、要求されたデータベース・サービスを提供するアクティブ・インスタンスにクライアントを接続します。

ファイル・システム

ファイル・システムとは、ディスクへの構造化アクセスを提供するソフトウェア・コンポーネントです。ファイル・システムは、ファイルなど

のオブジェクトをアプリケーション・プログラムに提供します。一般に、ファイルへのアクセスは、アプリケーション・プログラムがファイル・アクセスに使用するオペレーティング・システム・コール(Open/CloseやRead/Writeなど)を定義している標準APIで指定されます。ファイル・システムは、通常はオペレーティング・システムのコンポーネントとして提供されますが、独立したソフトウェア・コンポーネントとして提供されることもあります。

ディスク書込みの強制実行

Oracle RACでは、ある特定のデータ・ブロックを変更できるのは一時点で1つのインスタンスのみです。あるインスタンスが必要としているデータ・ブロックが、別のインスタンスによって変更された場合、そのブロックに対して発行されたリクエストのタイプによっては、ディスク書込みの強制実行が必要になります。

General Parallel File System

General Parallel File System (GPFS)は、IBM社の共有ディスク・ファイル・システム製品。同機種または異機種クラスタ内のすべてのノードからのデータ・アクセスを提供します。

グローバル・キャッシュ・サービス(GCS)

キャッシュ・フュージョンを実装するプロセス。グローバル・ロール内のブロックのブロック・モードを保持します。インスタンス間のブロックの転送を担います。グローバル・キャッシュ・サービスでは、グローバル・キャッシュ・サービス・プロセス(LMSn)、グローバル・エンキュー・サービス・デーモン(LMD)などの、様々なバックグラウンド・プロセスが使用されます。

グローバル・キャッシュ・サービス・プロセス(LMSn)

リモート・メッセージを管理するプロセス。Oracle RACでは最大で10のグローバル・キャッシュ・サービス・プロセスが提供されます。

グローバル・キャッシュ・サービス(GCS)リソース

複数のOracle RACインスタンスのバッファ・キャッシュ内のデータ・ブロックへのアクセスを調整し、キャッシュ一貫性を提供するグローバル・リソース。

グローバル・データベース名

データベースを他のデータベースから一意に識別する完全な名前。グローバル・データベース名の書式は database_name.database_domainです—たとえば: OP.EXAMPLE.COM。

グローバル動的パフォーマンス・ビュー(GV$)

Oracle RACクラスタ内のすべてのオープン・インスタンスに関する情報を格納する動的パフォーマンス・ビューです。(ローカル・インスタンスには限りません。)これに対して、標準の動的パフォーマンス・ビュー(V$)は、ローカル・インスタンスに関する情報のみを格納します。

グローバル・エンキュー・サービス(GES)

グローバルに共有されるエンキューを調整するサービス。

グローバル・エンキュー・サービス・デーモン(LMD)

リソースへのリクエストを管理してブロックへのアクセスを制御する、リソース・エージェント・プロセス。LMDはデッドロックの検出およびリモート・リソース要求の処理も行います。リモート・リソース・リクエストとは、別のインスタンスから発行されたリクエストです。

グローバル・エンキュー・サービス・モニター(LMON)

クラスタ全体を監視してグローバル・リソースを管理するバックグラウンドのLMONプロセス。LMONはインスタンスの完全破損および障害が発生したインスタンスに関連付けられたリカバリを管理します。特に、LMONはグローバル・リソースに関連付けられたリカバリの部分を処理します。LMONによって提供されるサービスは、クラスタ・グループ・サービスとも呼ばれます。

グローバル・サービス・デーモン(GSD)

SRVCTLからのリクエストを受信して、起動、停止などの管理ジョブ・タスクを実行するコンポーネント。コマンドは各ノードでローカルに実行され、その結果はSRVCTLに戻されます。GSDはデフォルトでノードにインストールされています。

グリッドのプラグ・アンド・プレイ・デーモン(GPNPD)

このプロセスを使用すると、グリッド・プラグ・アンド・プレイ・プロファイルにアクセスできます。また、クラスタのノード間でプロファイルの更新が調整され、すべてのノードで最新のプロファイルが保持されます。

High Availability Cluster Multi-Processing

IBM AIXベースの高可用性クラスタ・ソフトウェア製品。HACMPは高可用性(HA)とクラスタ・マルチプロセッシング(CMP)の2つの主なコンポーネントで構成されます。

高可用性

冗長コンポーネントを搭載したシステムであり、ハードウェアまたはソフトウェア障害が発生した場合でも一貫性があり中断されないサービスを提供します。ある程度の冗長性を含みます。

インスタンス

Oracle RACデータベースの場合、クラスタ内の各ノードには、通常、データベースを参照する実行中のOracleソフトウェアのインスタンスが1つ存在します。データベースが起動されると、Oracle Databaseによってシステム・グローバル領域(SGA)と呼ばれるメモリー領域が割り当てられ、1つ以上のOracle Databaseプロセスが起動されます。このSGAとOracle Databaseプロセスの組合せはインスタンスと呼ばれます。各インスタンスには、一意のOracleシステム識別子(SID)、インスタンス名、ロールバック・セグメントおよびスレッドIDが割り当てられます。

インスタンス・ケージング

フォアグラウンド・プロセス用にインスタンスが同時に使用できるCPUの数を制限するために初期化パラメータを使用する方法。

インスタンス・メンバーシップ・リカバリ

Oracle RACで、すべてのクラスタ・メンバーが機能しているか、またはアクティブであることを保証するために使用される方法です。インスタンス・メンバーシップ・リカバリによってメンバーシップがポーリングおよび調整されます。制御ファイルでハートビートを示していないメンバー、または定期的なアクティビティ問合せメッセージに応答しないメンバーは、停止しているとみなされます。

インスタンス名

クラスタで共通のサービス名が共有されている場合に、特定のインスタンスを一意に識別するために使用されるインスタンスの名前。インスタンス名は、インスタンス初期化ファイル(initsid.ora)のINSTANCE_NAMEパラメータによって識別されます。Oracleシステム識別子(SID)と同じです。

インスタンス番号

データ・ブロックのエクステントを特定のインスタンスと関連付ける番号。インスタンス番号を使用すると、インスタンスを起動し、そのインスタンスに割り当てられたエクステントが挿入および更新に使用されるようにできます。これによって、そのインスタンスが他のインスタンスに割り当てられた領域を使用していないことを確認できます。

インターコネクト

ノード間の通信リンク。

キーストア

透過的データ暗号化キーを格納するコンテナです。以前のリリースでは、これはウォレットと呼ばれていました。

論理ボリューム・マネージャ(LVM)

オンラインでのディスク記憶域管理に使用される、LinuxまたはUNIXのサブシステムの総称。

プロセス間通信(IPC)

オペレーティング・システム依存の高速転送コンポーネント。IPCは異なるノード上のインスタンス間でメッセージを転送します。インターコネクトとも呼ばれます。

マスター・ブート・レコード(MBR)

コンピュータの起動時に実行されるプログラム。通常、MBRはローカル・ハード・ディスクの最初のセクターに存在します。パーティション表を調査し、システムの起動に使用するパーティションを決定して、起動プロセスを開始します。その後、MBRプログラムは

起動パーティションのブート・セクターに制御を移し、起動プロセスを続行させます。

メモリー不足

サーバー上の使用可能なメモリー量が限られていることを示す状態。

メトリック

累積統計の変更率です。

マルチテナント・コンテナ・データベース

「CDB」を参照してください。

可変

コールされるたびに結果を変更できる非deterministic関数。リプレイ時に関数の結果が変更されると、可変関数によってリプレイは拒否されることがあります。キー値で使用されるsequence.nextvalおよびSYSDATEを検討してください。主キーがこれらの関数コールからの値で構築され、後の外部キーまたは他のバインドで使用される場合は、リプレイで同じ関数結果が返される必要があります。アプリケーション・コンティニュイティでは、付与されたOracle関数コールに対してリプレイ時に可変値置換を提供して、不透明なバインド/変数一貫性を実現します。

ネットワーク接続ストレージ(NAS)

ネットワーク経由でサーバーに接続された記憶域。

ネットワーク・タイム・プロトコル(NTP)

TCP/IPの最上位に構築されたインターネット標準プロトコル。このプロトコルは、ネットワーク内のコンピュータ時計の時刻をミリ秒単位で正確に同期化します。

ネットワーク・インタフェース・カード(NIC)

コンピュータをネットワークに接続するために、コンピュータに挿入するカード。

ノード

ノードは、Oracle RACおよびOracle Clusterwareソフトウェアがインストールされているコンピュータ・システムです。

Object Link Manager

シンボリック・リンクを論理ドライブにマッピングし、それらをOLMのグラフィカル・ユーザー・インタフェースに表示するOracleインタフェース。

OCSSD

クラスタ同期サービス(CSS)・デーモンを管理するLinuxまたはUNIXのプロセスです。クラスタ・ノード・メンバーシップを管理し、oracleユーザーとして実行するプロセスであり、このプロセスが失敗した場合はクラスタが再起動されます。

Oracle Cluster File System

Oracleでは、Linux用のOCFS2およびOracle ASM Cluster File System (Oracle ACFS)という2つのファイル・システムを提供しています。Oracle ACFSは独自のファイル・システムですが、Linux用のOCFS2のソースは、GNUのGeneral Public License (GPL)に基づいて使用できます。2つのファイル・システムに互換性はありません。

Oracle Cluster Registry (OCR)

クラスタ・ノード・リストに関する情報およびインスタンスからノードへのマッピング情報を管理するOracle RAC構成情報のリポジトリ。OCRは、カスタマイズされたアプリケーション用のOracle Clusterwareリソース・プロファイルに関する情報も管理します。

Oracle Clusterware

ノードのメンバーシップ、グループ・サービス、グローバル・リソース管理、高可用性機能などのクラスタ・データベースの処理を管理するOracle提供のクラスタウェア。

Oracle拡張クラスタ

サイトと呼ばれる複数の場所に配置されているノードで構成されるクラスタです。

Oracle Flex Cluster

ハブ・ノードおよびサポートされている他のノードで構成された大きなクラスタ。ハブ・ノードは、現在のメンバーシップ・アルゴリズムを使用してクラスタを形成し、他のノードはメンバーシップのために単一のハブ・ノードに接続します。

Oracle Grid Infrastructure

エンタープライズ・グリッド・アーキテクチャ用のインフラストラクチャを提供するソフトウェア。クラスタの場合は、Oracle ClusterwareとOracle Automatic Storage Management(Oracle ASM)が含まれます。スタンドアロン・サーバーの場合は、Oracle RestartとOracle ASMが含まれます。Oracle Databaseでは、これらのインフラストラクチャ製品が組み合されてOracle Grid Infrastructureホーム(Grid_home)と呼ばれる1つのソフトウェア・インストールになっています。

Oracleグリッド・ネーミング・サービス・デーモン(GNSD)

Oracleグリッド・ネーミング・サービスは、クラスタmDNSと外部DNSサーバー間のゲートウェイ。gnsdプロセスは、クラスタ内で名前解決を実行します。

Oracle高可用性サービス・デーモン(OHASD)

このプロセスは、クラスタ操作を円滑化するプロセスで構成される、Oracle Clusterwareスタックの下位部分を常駐させます。

Oracle Interface Configurationツール(OIFCFG)

非クラスタのOracle DatabaseとOracle RACデータベースの両方で使用されるコマンドライン・ツールです。このツールを使用すると、コンポーネントへのネットワーク・インタフェースの割当ておよび割当て解除を行ったり、特定のネットワーク・インタフェースを使用するようにコンポーネントに指定したり、コンポーネントの構成情報を取得することができます。Oracle Universal Installerも、OIFCFGを使用して、使用可能なインタフェースの識別および表示を行います。

Oracle Managed Files

いくつかの初期化パラメータに基づいて、制御ファイル、REDOログ・ファイル、データファイルなどのデータベース・ファイルの名前の指定、場所の設定、作成、削除を自動化するサービスです。Oracle Managed Filesは、VxFSやODMなどのホスト・オペレーティング・システムでサポートされている従来のファイル・システムに加えて使用できます。データベース管理の多くの側面について独自の方針を作成する必要をなくすことによって、それらの詳細を簡略化できます。

Oracle Notification Service

すべてのFANイベントに関する情報を通信する、パブリッシュおよびサブスクライブ・サービス。

Oracle Universal Installer

Oracle Clusterware、Oracleリレーショナル・データベース・ソフトウェアおよびOracle RACソフトウェアをインストールするためのツールです。Oracle Universal Installerを使用してDatabase Configuration Assistant(DBCA)を起動することもできます。

Oracle XA

Oracle Database以外のトランザクション・マネージャでグローバル・トランザクションを調整できるようにする外部インタフェース。

PDB

マルチテナント・コンテナ・データベース(CDB)で、Oracle Netクライアントに非CDBとして表示されるスキーマ、スキーマ・オブジェクトおよび非スキーマ・オブジェクトのポータブル・コレクション。

プラガブル・データベース

「PDB」を参照してください。

ポリシー管理データベース

クラスタ・リソースとして定義したデータベースです。データベースの管理は、データベースがどのサーバーで実行可能か、予想されるワークロードのサポートにデータベースのインスタンスがいくつ必要かといった、リソースの構成方法によって定義されます。

RAWデバイス

ファイル・システムがまだ設定されていないディスク・ドライブ。ディスクの共有が可能であるため、Oracle RACに使用されます。「RAWパーティション」も参照してください。

RAWパーティション

最小限のアクセス・レベルでアクセスされる物理ディスクの部分。拡張パーティションが作成され、論理パーティションがフォーマットされずに拡張パーティションに割り当てられた場合に作成されます。フォーマットが完了したパーティションは、クックド・パーティションと呼ばれます。「RAWデバイス」も参照してください。

リカバリ可能なエラー

実行中のアプリケーション・セッション・ロジックとは関係なく、外部システムの障害が原因で発生するエラーのクラス。リカバリ可能なエラーは、フォアグラウンド、ネットワーク、ノード、記憶域、データベースの計画済停止および計画外停止に続いて発生するエラーです。アプリケーションは、最後に発行された操作のステータスを把握しないままの状態で残される可能性があるエラー・コードを受信します。

Recovery Manager (RMAN)

データファイル、制御ファイルおよびアーカイブREDOログ・ファイルをバックアップ、コピー、リストアおよびリカバリできるツール。Oracleサーバーに含まれており、個別にインストールする必要はありません。RMANは、オペレーティング・システム(OS)のプロンプトからコマンドライン・ユーティリティとして実行するか、またはGUIベースのOracle Enterprise ManagerのBackup Managerを使用して実行できます。

リージョン

互いに近くに存在すると考えられるデータベース・クライアントおよびサーバーが含まれる論理的な境界です。

リクエスト

アプリケーションから送信される作業単位。リクエストは通常、単一データベース接続での単一WebリクエストのSQL、PL/SQLおよびその他のデータベース・コールに対応し、一般に接続プールからのデータベース接続のチェックアウトとチェックインのために作成されたコールによって区別されます。

リクエスト境界

リクエスト境界は、アプリケーションまたはアプリケーション・サーバーがデータベース接続プールから接続を流用および返却する場所をマークします。

結果キャッシュ

結果キャッシュはSGAまたはクライアント・アプリケーション・メモリー内のメモリー領域で、データベースの問合せまたは問合せブロックの結果を再利用するために格納します。キャッシュされた行は、失効しないかぎり文およびセッション間で共有されます。

ランタイム接続ロード・バランシング

Oracle Databaseは、要求されたアプリケーションに対して現在のワークロードに応じて最適なサービスを提供する接続プールに基づいて、インテリジェントなサービス接続決定を行うことができます。JDBC、ODP.NETおよびOCIクライアントはロード・バランシング・アドバイザと統合され、これらのいずれかのクライアント環境を使用して、ランタイム接続ロード・バランシングを使用できます。

スケーラビリティ

Oracle RACアプリケーションにノードを追加し、大幅なスケールアップおよびスピードアップを実現する機能です。

セキュア・シェル(SSH)

ネットワーク上のリモート・コンピュータにログインするためのプログラム。SSHを使用すると、リモート・システム上でコマンドを実行し、ファイルをあるシステムから別のシステムに移動できます。SSHでは厳密認証が使用され、セキュアでないチャネル上での通信を保護します。

サーバー制御(SRVCTL)ユーティリティ(SRVCTL)

Server Management(SRVM)は、Oracle Enterprise ManagerをOracle RACで操作するために必要なコンポーネントを構成します。Intelligent Agent、グローバル・サービス・デーモン、SRVCTLなどのSRVMコンポーネントを使用すると、オープンなクライアント/サーバー・アーキテクチャを介して異機種間環境で実行されているクラスタ・データベースを、Oracle Enterprise Managerを使用して管理できます。

サーバー

Oracleソフトウェアがインストールされていないコンピュータ・システムです。

サーバー・クラスタ

共有GNSサーバーが実行されているクラスタ。

サーバー・グループ

アプリケーション、データベースまたはその両方をホスティングするグループに論理的に分割されているクラスタ内のノードです。サーバー・グループはその他のサーバー・グループのメンバーになることができます。

サービス・レベル

システムのパフォーマンスの尺度。

サービス

Oracle RACデータベースで定義可能なエンティティ。サービスによって、データベースのワークロードをグループ化し、サービスを提供するために割り当てられた最適なインスタンスに作業をルーティングできます。

セッション状態一貫性

COMMITが実行された後にそのトランザクションの状態が変更された場合、セッションが失われていると、トランザクションをリプレイしてその状態を再確立することはできません。アプリケーション・コンティニュイティを構成する場合、アプリケーションは、初期設定後のセッション状態が動的であるか静的であるか、および要求内の過去のCOMMIT操作を継続するのが適切であるかどうかに応じて分類されます。

- 動的: (デフォルト)セッション状態の変更が初期化で完全にカプセル化されていない場合、およびフェイルオーバー時にコールバックで完全に取得できない場合、そのセッションは動的な状態です。要求内の最初のトランザクションがコミットされると、次の要求が開始されるまでフェイルオーバーは内部的に無効化されます。これは、ほとんどのアプリケーションが要求に使用するデフォルト・モードです。
- 静的: (要求での特殊設定) NLS設定やPL/SQLパッケージの状態など、すべてのセッション状態の変更を初期化コールバックで繰り返すことができる場合、そのセッションは静的な状態です。この設定は、要求内のセッション状態を変更しないデータベース診断アプリケーションのみに使用されます。コールバックにより再確立できない非トランザクション状態変更が要求内にある場合は、静的モードを設定しないでください。不明な場合は、動的モードを使用します。

Shared Everything

すべてのインスタンスによってすべてのデータへのアクセスが共有されるデータベースのアーキテクチャ。

単一クライアント・アクセス名(SCAN)

Oracle Database 11gのデータベース・クライアントは、SCANを使用してデータベースに接続します。SCANは、パブリック・クライアント接続を処理するクラスタ内の複数のリスナーに対応する、複数のIPアドレスに解決できます。

シングルトン・サービス

一度に1つのインスタンス上のみで実行されるサービス。サービスの分散トランザクション処理(DTP)プロパティを定義することによって、サービスを強制的にSINGLETONサービスにすることができます。

スプリット・ブレイン・シンドローム

複数のインスタンスによってクラスタ・データベースの制御が試行される状態。たとえば、2ノード環境で、一方のインスタンスによって更新の管理が試行され、もう一方のインスタンスによって同時に更新の管理が試行される状態です。

SQL翻訳プロファイル

SQL翻訳プロファイルは、Oracle以外のデータベースのSQL文をOracleに翻訳する方法、およびOracleエラー・コードとANSI SQLSTATESを他のベンダーの等価のものに翻訳する方法を指示する、データベース・スキーマ・オブジェクトです。

システム識別子

Oracleシステム識別子(SID)は、実行中のOracleソフトウェアの特定のインスタンスを識別します。Oracle RACデータベースの場合、クラスタ内の各ノードにはデータベースを参照するインスタンスが存在します。

透過的アプリケーション・フェイルオーバー(TAF)

Oracle RACやOracle RAC Guardなどの高可用性環境を対象としたランタイム・フェイルオーバーで、TAFとは、アプリケーションからサービスへの接続のフェイルオーバーおよび再確立を指します。これにより、クライアント・アプリケーションは接続障害の発生時にデータベースに自動的に再接続でき、実行中だったSELECT文を再開することも可能です。この再接続は、Oracle Call Interfaceライブラリ内から自動的に実行されます。

仮想インターネット・プロトコル(VIP)

特定の単一サーバーまたはネットワーク・インタフェース・カード(NIC)に割り当てられるのではなく、単一サーバー、複数のドメイン名または複数のサーバーに存在している複数のアプリケーションに割り当てられるIPアドレス。

ボリューム・マネージャ

ディスクの断片の集合をボリュームにマップする処理を管理するソフトウェア・コンポーネント。

投票ディスク

ノードのメンバーシップに関する情報を管理するファイル。

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

- config srvpool A.9.115
- config vip A.9.120
- config volume A.9.133
- convert database A.9.3
- disable database A.9.4
- disable diskgroup A.9.19
- disable instance A.9.30
- disable listener A.9.40
- disable nodeapps A.9.59
- disable ons A.9.71
- disable scan A.9.80
- disable scan_listener A.9.91
- disable service A.9.105
- disable vip A.9.121
- disable volume A.9.134
- downgrade database A.9.5
- enable database A.9.6
- enable diskgroup A.9.20
- enable instance A.9.31
- enable listener A.9.41
- enable nodeapps A.9.60
- enable ons A.9.72
- enable scan A.9.81
- enable scan_listener A.9.92
- enable service A.9.106
- enable vip A.9.122
- enable volume A.9.135
- -eval parameter A.1
- getenv database A.9.7
- getenv listener A.9.42
- getenv nodeapps A.9.61, A.9.123
- ヘルプ A.5
- modify database A.9.8
- modify instance A.9.32
- modify listener A.9.43
- modify network A.9.54
- modify nodeapps A.9.62
- modify ons A.9.73
- modify scan A.9.82
- modify scan_listener A.9.93
- modify service A.9.107
- modify srvpool A.9.116
- modify vip A.9.124

- predict database A.9.9
- predict diskgroup A.9.21
- predict listener A.9.44
- predict network A.9.55
- predict scan A.9.83
- predict scan_listener A.9.94
- predict service A.9.108
- predict vip A.9.125
- relocate database A.9.10
- relocate scan A.9.84
- relocate scan_listener A.9.95
- relocate server A.9.101
- relocate service A.9.109
- relocate vip A.9.126
- remove database A.9.11
- remove diskgroup A.9.22
- remove instance A.9.33
- remove listener A.9.45
- remove network A.9.56
- remove nodeapps A.9.63
- remove ons A.9.74
- remove scan A.9.85
- remove scan_listener A.9.96
- remove service A.9.110
- remove srvpool A.9.117
- remove vip A.9.127
- remove volume A.9.136
- setenv database A.9.12, A.9.128
- setenv listener A.9.46
- setenv nodeapps A.9.64
- srvctl setenv 3.1.1.1
- start database 3.2.1, A.9.13
- start diskgroup A.9.23
- start home A.9.26
- start instance 3.2.1, A.9.34
- start listener A.9.47
- start nodeapps A.9.65
- start ons A.9.75
- start scan A.9.86
- start scan_listener A.9.97
- start service A.9.111
- start vip A.9.129
- start volume A.9.137

T

U

V

W

X

- XAアフィニティ 5.4.2
- XAトランザクション 5.4.1
    - Oracle RACインスタンスにまたがる 5.4.1
    - サービスの使用 5.4.3
- XAトランザクション 13.3.4